

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA TOÁN - CƠ - TIN HỌC**



## **COOKING ASSISTANT CHATBOT**

Nhóm sinh viên thực hiện:

Nguyễn Thị Hòa	:	23001521
Đào Thị Ngọc Bích	:	23001501
Đinh Thị Kiều Na	:	23001537
Dương Diễm Quỳnh	:	23001555
Lưu Thị Thủy Tiên	:	23001563

Mã học phần: MAT3508  
Học kỳ I, Năm học 2025–2026

# Thông tin Dự án

**Học phần:** MAT3508 – Nhập môn Trí tuệ Nhân tạo

**Học kỳ:** Học kỳ 1, Năm học 2025-2026

**Trường:** VNU-HUS (Đại học Quốc gia Hà Nội, Trường Đại học Khoa học Tự nhiên)

**Tên dự án:** Cooking Assistant Chatbot

**Ngày nộp:** 30/11/2025

**Báo cáo PDF:** Liên kết tới báo cáo PDF trong kho GitHub

**Slide thuyết trình:** Liên kết tới slide thuyết trình trong kho GitHub

**Kho GitHub:** <https://github.com/IamHoa05/cooking-assistant-chatbot>

## Thành viên nhóm

Họ tên	Mã sinh viên	Tên GitHub	Đóng góp
Nguyễn Thị Hòa	23001521	IamHoa05	Trưởng nhóm Backend Xây dựng NLP
Đào Thị Ngọc Bích	23001501	daobich14	Frontend Tiền xử lý dữ liệu
Đinh Thị Kiều Na	23001537	kieuna2005	Frontend Xây dựng Embedding
Dương Diễm Quỳnh	23001555	ddquynh	Backend Xây dựng LLM
Lưu Thị Thủy Tiên	23001563	ttien2312	Backend Xây dựng NLP

# Danh sách hình vẽ

2.1	Nguyên lý hoạt động của Embedding . . . . .	7
2.2	Quy trình xử lý văn bản của kiến trúc Transformer . . . . .	9
2.3	Pipeline xác định intent . . . . .	11
3.1	Phân phối similary món ăn . . . . .	26
3.2	Phân phối similary nguyên liệu . . . . .	27

# Danh sách bảng

2.1	Thông số kỹ thuật trong triển khai mô hình BGE-M3 . . . . .	10
2.2	So sánh thông số kỹ thuật giữa BGE-M3 và PhoBERT . . . . .	11
2.3	Các thư viện tokenization phổ biến trong dự án . . . . .	15
2.4	So sánh các mô hình LLaMA 3.1 8B instant, LLaMA Local và GPT-4, GPT-3.5 . . . . .	21
3.1	Kết quả phân loại intent . . . . .	28
3.2	Kết quả trích xuất slot . . . . .	28
3.3	Hiệu suất truy vấn của các FAISS Index . . . . .	29
3.4	Độ chính xác của các FAISS Index . . . . .	30

# Mục lục

<b>1</b>	<b>Giới thiệu</b>	<b>1</b>
<b>2</b>	<b>Phương pháp &amp; Triển khai</b>	<b>2</b>
2.1	Thu thập và làm sạch dữ liệu . . . . .	2
2.1.1	Thu thập dữ liệu . . . . .	2
2.1.2	Tiền xử lý dữ liệu . . . . .	4
2.2	Xây dựng Embedding . . . . .	7
2.2.1	Giới thiệu chung về embedding . . . . .	7
2.2.2	Lựa chọn mô hình embedding . . . . .	8
2.2.3	Triển khai embedding cho dữ liệu . . . . .	10
2.2.4	So sánh BGE-M3 với PhoBERT . . . . .	10
2.3	Xây dựng NLP . . . . .	11
2.3.1	Xác định intent và trích xuất thông tin (Slot Extraction) . . . . .	11
2.3.2	Xây dựng hệ thống tìm kiếm với FAISS . . . . .	17
2.4	Xây dựng LLM . . . . .	20
2.4.1	Lựa chọn mô hình . . . . .	20
2.4.2	Thực hiện mô hình . . . . .	22
<b>3</b>	<b>Kết quả &amp; Phân tích</b>	<b>24</b>
3.1	Đánh giá mô hình Embedding . . . . .	24
3.1.1	Tổng quan . . . . .	24
3.1.2	Phương pháp đánh giá . . . . .	24
3.1.3	Kết quả đánh giá . . . . .	25
3.2	Hiệu suất xác định Intent & Slot Extraction . . . . .	28
3.2.1	Xác định intent . . . . .	28
3.2.2	Slot Extraction . . . . .	28
3.2.3	Phân tích . . . . .	29
3.3	Hiệu suất tìm kiếm với FAISS . . . . .	29
3.3.1	Thông tin Index . . . . .	29
3.3.2	Hiệu suất truy vấn . . . . .	29
3.3.3	Độ chính xác . . . . .	29

3.3.4	Phân tích . . . . .	30
3.4	Hiệu suất và độ chính xác của LLM . . . . .	30
3.4.1	Hiệu suất . . . . .	30
3.4.2	Độ chính xác . . . . .	30
<b>4</b>	<b>Kết luận</b>	<b>32</b>

# Lời mở đầu

Trong kỷ nguyên chuyển đổi số hiện nay, trí tuệ nhân tạo (Artificial Intelligence – AI) và các kỹ thuật xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) đã trở thành nền tảng quan trọng trong việc phát triển các hệ thống tương tác người – máy. Bên cạnh các lĩnh vực phổ biến như chăm sóc khách hàng, giáo dục hay y tế, việc ứng dụng AI vào hỗ trợ nấu ăn đang thu hút sự quan tâm nhờ khả năng đáp ứng nhu cầu thực tiễn, mang tính cá nhân hóa và hỗ trợ tức thời cho người dùng.

Xuất phát từ nhu cầu tối ưu hóa trải nghiệm nấu ăn, đặc biệt trong bối cảnh người dùng ngày càng ưu tiên sự nhanh chóng và tiện lợi, nhóm thực hiện đề tài “Cooking Assistant Chatbot”. Hệ thống được xây dựng nhằm hỗ trợ người dùng tra cứu công thức nấu ăn, gợi ý món ăn dựa trên nguyên liệu sẵn có, hướng dẫn thao tác chế biến. Lựa chọn ứng dụng mô hình ngôn ngữ lớn (Large Language Models – LLMs), kết hợp thuật toán tìm kiếm, phân loại ý định (intent classification) và trích xuất thông tin, giúp chatbot có khả năng tương tác tự nhiên và đưa ra kết quả phù hợp với yêu cầu từ người dùng.

Báo cáo này trình bày toàn bộ quy trình triển khai hệ thống, bao gồm thiết kế kiến trúc, xây dựng tập dữ liệu, lựa chọn mô hình, phát triển thuật toán đề xuất món ăn và tích hợp backend – frontend. Đồng thời, báo cáo cũng đánh giá hiệu quả hệ thống thông qua các tiêu chí về độ chính xác, khả năng phản hồi và mức độ phù hợp với nhu cầu người dùng. Những kết quả đạt được không chỉ minh chứng cho tiềm năng ứng dụng AI trong lĩnh vực ẩm thực thế giới nói chung và ẩm thực Việt Nam nói riêng, mà còn mở ra nhiều hướng phát triển trong tương lai, đặc biệt trong việc nâng cao tính cá nhân hóa và tự động hóa quy trình nấu ăn trong cuộc sống hằng ngày.

Do hạn chế về kiến thức và thời gian, bài báo cáo này có thể còn nhiều thiếu sót. Chúng em rất mong nhận được sự góp ý và phản hồi để hoàn thiện hơn trong những lần báo cáo sau.

Chúng em xin chân thành cảm ơn!

Hà Nội, ngày 30 tháng 11 năm 2025

# Chương 1

## Giới thiệu

Thông qua áp dụng các kiến thức thuộc lĩnh vực Trí tuệ nhân tạo (đặc biệt trong mảng Xử lý ngôn ngữ tự nhiên), trợ lý ảo thông minh CookGPT ra đời với mục đích hỗ trợ người dùng Việt Nam trở thành "master chef" một cách nhanh chóng, tiện lợi và thú vị.

Dự án Cooking Assistant Chatbot tập trung vào các lĩnh vực sau:

- **Ngôn ngữ:** Sử dụng tiếng Việt, đảm bảo các hướng dẫn và gợi ý về món ăn phù hợp với người dùng địa phương.
- **Chức năng:** Hệ thống cung cấp gợi ý về món ăn, mô tả món ăn, hướng dẫn nấu ăn chi tiết, phân loại món theo độ khó, thời gian chế biến và khẩu phần. Hiện tại, hệ thống chưa hỗ trợ tạo món ăn mới hoặc fine-tune mô hình trực tiếp.
- **Ưu điểm của dự án:**
  - Tích hợp các chức năng phong phú: phân loại món ăn theo độ khó, thời gian, khẩu phần, gợi ý món ăn từ nguyên liệu có sẵn.
  - Giúp người dùng tiết kiệm thời gian tra cứu và tổng hợp từ nhiều nguồn: đưa ra danh sách kết quả phù hợp yêu cầu của người dùng với độ chính xác cao trong thời gian ngắn.
  - Cung cấp các trải nghiệm tương tác vui vẻ, sinh động cho người dùng qua việc dùng giọng văn đậm chất Gen Z: thoải mái, gần gũi, hài hước và dễ hiểu.

Với các mục tiêu và phạm vi trên, CookGPT hướng tới việc trở thành một trợ lý ẩm thực tiện ích, thân thiện, dễ sử dụng và luôn mang lại trải nghiệm ấn tượng cho người dùng trong suốt quá trình nghiên cứu, chế biến và thử nghiệm món ăn.

# Chương 2

## Phương pháp & Triển khai

### 2.1 Thu thập và làm sạch dữ liệu

#### 2.1.1 Thu thập dữ liệu

Dữ liệu được nhóm thu thập từ trang web "Món ngon mỗi ngày" (monngonmoin-gay.com), với mục tiêu xây dựng một kho dữ liệu phong phú gồm hơn 2000 công thức nấu ăn. Dự án này sử dụng phương pháp Web Crawling tiên tiến với công cụ Selenium WebDriver để tự động hóa quá trình truy cập, tương tác và trích xuất thông tin chi tiết từ các trang web động (JavaScript-rendered pages).

Mục đích của việc thu thập dữ liệu này không chỉ dừng lại ở việc tạo ra một tập dữ liệu thô, mà còn nhằm cung cấp nền tảng vững chắc cho các ứng dụng phân tích dữ liệu chuyên sâu như:

- **Phân tích xu hướng ẩm thực:** Xác định các nguyên liệu phổ biến, các loại món ăn được tìm kiếm nhiều nhất, và thời gian chế biến trung bình.
- **Xây dựng hệ thống gợi ý (Recommendation System):** Phát triển mô hình đề xuất món ăn dựa trên khẩu phần, độ khó, hoặc các nguyên liệu sẵn có của người dùng.
- **Nghiên cứu ngôn ngữ tự nhiên (NLP):** Phân tích cấu trúc và nội dung của các bước hướng dẫn nấu ăn để tối ưu hóa quy trình học máy.

Quá trình thu thập được thực hiện theo nguyên tắc ổn định và bảo toàn dữ liệu, cho phép tích hợp và cập nhật dữ liệu liên tục qua nhiều lần chạy mà không làm mất đi kho dữ liệu hiện có.

Trong dự án này, chúng em sử dụng công cụ Selenium WebDriver để thu thập dữ liệu (crawl) dựa trên các lý do sau:

Trang web, "Món ngon mỗi ngày", sử dụng JavaScript để tải và hiển thị nhiều nội dung quan trọng như danh sách công thức, tên món, nguyên liệu, và các bước thực hiện.

Các công cụ crawl dữ liệu truyền thống, như thư viện Requests hay BeautifulSoup, chỉ có khả năng đọc mã nguồn HTML tĩnh ban đầu (Source HTML) trước khi JavaScript chạy. Do đó, nếu không có Selenium, chúng em sẽ không thu thập được phần lớn dữ liệu cần thiết. Selenium mô phỏng một trình duyệt hoàn chỉnh (như Chrome), kích hoạt môi trường để JavaScript chạy, đảm bảo mọi nội dung đều được tải đầy đủ trước khi trích xuất.

Dự án yêu cầu khả năng tương tác với giao diện trang web, cụ thể là:

- **Xử lý Pop-up:** Mã nguồn cần đóng pop-up cookie (`span.cookieDrawer__close`) xuất hiện lần đầu để không bị che khuất nội dung chính. Selenium cho phép định vị và thực hiện hành động nhấp chuột để xử lý tương tác này.
- **Cuộn trang (Scroll):** Trang danh sách sử dụng kỹ thuật tải chậm (Lazy Loading), Selenium là công cụ cần thiết để mô phỏng hành vi cuộn xuống, buộc trang tải thêm các công thức tiếp theo.

Selenium cung cấp cơ chế `WebDriverWait` và `expected_conditions`. Cơ chế này cho phép chúng em chỉ thị cho chương trình chờ đợi một cách thông minh cho đến khi một phần tử cụ thể (ví dụ: nút đóng pop-up) xuất hiện và sẵn sàng để tương tác trong một khoảng thời gian nhất định (ví dụ: 5 giây). Điều này giúp quá trình crawl trở nên bền vững và ít bị lỗi hơn trước sự thay đổi về tốc độ tải trang hoặc thời gian phản hồi của máy chủ, thay vì chỉ dựa vào lệnh dừng cố định (`time.sleep`) không đáng tin cậy.

Quy trình thực hiện thu thập dữ liệu công thức nấu ăn của nhóm em được tiến hành một cách tuần tự và có kiểm soát, sử dụng Selenium WebDriver để xử lý các trang web động. Quy trình này bao gồm bốn bước chính:

1. **Thiết lập môi trường và khởi tạo phiên làm việc:** Đầu tiên, chúng em khởi tạo một phiên trình duyệt Chrome mới thông qua `webdriver.Chrome()`. Đồng thời, các tham số quan trọng cho quá trình crawl được xác định, bao gồm URL gốc, phạm vi trang cần duyệt (từ `start_page=5` đến `max_pages=20`), và số lượng công thức tối đa mỗi trang (`limit_per_page=50`).
2. **Thu thập URL và hình ảnh từ trang danh sách (List Crawl):** Chúng em thực hiện lặp qua từng trang danh sách theo phạm vi đã thiết lập. Khi truy cập trang đầu tiên, chương trình sẽ sử dụng `WebDriverWait` để phát hiện và đóng cửa sổ pop-up cookie (`span.cookieDrawer__close`), đảm bảo việc thu thập dữ liệu không bị cản trở. Sau đó, chương trình tìm kiếm các thẻ chứa liên kết công thức (`div.relative.rounded-xl a`) và trích xuất URL chi tiết (`href`) cùng với đường dẫn hình ảnh (`src`). Các link này được lưu vào danh sách `all_recipes` để chuẩn bị cho bước tiếp theo.

3. **Thu thập chi tiết dữ liệu công thức (Detail Crawl):** Sử dụng danh sách URL đã thu thập, chương trình lần lượt truy cập từng trang chi tiết công thức. Trên mỗi trang, chúng em sử dụng các CSS Selector chuyên biệt để trích xuất các trường thông tin cụ thể: Tên món (`span.title`), Nguyên liệu (`div.block-nguyenlieu li`), Các bước nấu (`#section-soche li`, `#section-thuchien p`), và các thông số Khẩu phần/Thời gian/Độ khó. Toàn bộ quá trình trích xuất được đặt trong khối `try...except` để gán giá trị mặc định ("NA") nếu một trường dữ liệu bị thiếu, đảm bảo quá trình crawl không bị gián đoạn.
4. **Hợp nhất và lưu trữ dữ liệu đầu ra:** Sau khi hoàn tất quá trình thu thập chi tiết, chương trình gọi `driver.quit()` để đóng trình duyệt. Tiếp theo, chúng em kiểm tra sự tồn tại của tệp `recipes.json`. Nếu tệp đã tồn tại, dữ liệu cũ sẽ được đọc và nối thêm dữ liệu mới thu thập được vào cuối. Cuối cùng, toàn bộ dữ liệu hợp nhất được ghi vào tệp `recipes.json` với định dạng JSON chuẩn (`ensure_ascii=False`, `indent=4`), đảm bảo dữ liệu tiếng Việt được hiển thị chính xác và dễ đọc, đồng thời duy trì tính bảo toàn dữ liệu qua các lần chạy.

### 2.1.2 Tiền xử lý dữ liệu

Sau khi hoàn tất quá trình thu thập dữ liệu thô bằng Selenium WebDriver và lưu trữ thành công vào tệp JSON, nhóm em tiến hành tiền xử lý dữ liệu (Data Preprocessing).

Tiền xử lý dữ liệu là một bước cực kỳ quan trọng, là giai đoạn bắt buộc sau khi thu thập dữ liệu thô và trước khi đưa dữ liệu vào mô hình. Tầm quan trọng của nó được thể hiện qua câu nói nổi tiếng trong khoa học dữ liệu: "Garbage In, Garbage Out" (Đầu vào rác, Đầu ra rác). Dữ liệu thô hiếm khi sạch sẽ, đầy đủ hoặc nhất quán; nếu không được xử lý, chúng sẽ dẫn đến kết quả phân tích sai lệch, mô hình kém chính xác, và lãng phí thời gian, tài nguyên.

#### Chuẩn hóa đơn vị khối lượng

Dữ liệu nguyên liệu thô được thu thập dưới dạng danh sách các chuỗi văn bản với khối lượng viết tắt nhiều và không đồng nhất (ví dụ: "Thịt heo nạc xay 500g", "Nước tương Maggi 1 M canh"). Chúng em đã xây dựng hàm `normalize_unit` để giải quyết vấn đề này:

Hàm `normalize_unit` hoạt động theo cơ chế xử lý từng bước như sau:

Đầu tiên, hàm kiểm tra kiểu dữ liệu đầu vào. Nếu tham số `text` là một danh sách (list), hàm sẽ tự gọi đệ quy để áp dụng xử lý lên từng phần tử trong danh sách đó, đảm bảo mọi mục trong danh sách đều được chuẩn hóa.

Tiếp theo, nếu đầu vào không phải là chuỗi ký tự (string), hàm sẽ trả về nguyên giá trị đó mà không thực hiện bất kỳ thay đổi nào.

Khi đầu vào là chuỗi ký tự, hàm sử dụng một bảng ánh xạ (`unit_map`) được định nghĩa sẵn để thay thế các ký hiệu đơn vị viết tắt bằng tên đầy đủ. Cụ thể, bảng này quy định các phép thay thế như: 'g' thành 'gam', 'M' thành 'muỗng', 'tr' thành 'trái', 'c' thành 'củ'. Quá trình thay thế này được thực hiện thông qua biểu thức chính quy (regex), đảm bảo rằng chỉ các ký hiệu đơn vị đứng ngay sau một giá trị số mới được thay thế, nhằm tránh việc thay đổi nhầm các ký tự trùng lặp trong tên nguyên liệu.

### Chuẩn hóa thời gian và số người ăn

Trong dữ liệu còn tồn tại đơn vị thời gian theo kiểu "1 giờ 20 phút"... Chúng em xây dựng hàm `normalize_cook_time` để chuẩn hóa thời gian sang phút.

Khi nhận đầu vào là một chuỗi mô tả thời gian như "1 giờ 20 phút", hàm sẽ xử lý qua ba bước chính.

Đầu tiên, hàm chuyển chuỗi về dạng chữ thường và loại bỏ khoảng trắng thừa để đảm bảo tính nhất quán.

Tiếp theo, hàm sử dụng biểu thức chính quy để trích xuất các thành phần giờ và phút: nó tìm kiếm các cụm từ chứa "giờ" hoặc "h" để lấy số giờ, đồng thời tìm các cụm từ chứa "phút" hoặc "ph" để lấy số phút.

Sau khi có được các giá trị này, hàm thực hiện quy đổi toàn bộ thời gian về đơn vị phút bằng cách nhân số giờ với 60 và cộng với số phút tương ứng.

Kết quả trả về là tổng số phút, cho phép so sánh và phân tích các giá trị thời gian một cách thống nhất trong toàn bộ tập dữ liệu.

### Chuẩn hóa tên nguyên liệu

Hàm `normalize_ingredient_name` được thiết kế để chuẩn hóa tên nguyên liệu bằng cách loại bỏ các thông tin không cần thiết và thống nhất cách gọi tên. Hàm hoạt động qua sáu bước chính.

Đầu tiên, hàm xóa các biểu tượng icon và chuyển toàn bộ tên nguyên liệu về chữ thường để đảm bảo tính nhất quán.

Tiếp theo, hàm loại bỏ tất cả các ký tự đặc biệt chỉ giữ lại chữ cái, số và khoảng trắng.

Bước thứ ba, hàm sử dụng một danh sách các từ mô tả trạng thái, hành động chế biến để loại bỏ các thông tin thừa như "băm", "cắt", "tươi", "khô" - ví dụ, "thịt heo cắt lát" sẽ trở thành "thịt heo".

Sau đó, hàm tiếp tục loại bỏ các tên thương hiệu cụ thể như "aji-ngon", "ajinomoto" khỏi tên nguyên liệu.

Bước thứ năm thực hiện việc gom nhóm và thay thế các tên nguyên liệu phức tạp thành tên chuẩn hóa - chẳng hạn, các biến thể của "hạt nêm ajinon" đều được chuyển thành "hạt nêm" thống nhất.

Cuối cùng, hàm dọn dẹp các khoảng trắng thừa và trả về tên nguyên liệu đã được chuẩn hóa. Quy trình này giúp tạo ra một từ điển nguyên liệu thống nhất, loại bỏ sự trùng lặp do cách gọi tên khác nhau, từ đó hỗ trợ hiệu quả cho việc phân tích dữ liệu và xây dựng hệ thống gợi ý công thức nấu ăn.

### Sửa lỗi chính tả các nguyên liệu trong trường `ingredient_names`

Hàm `apply_corrections` được sử dụng để sửa lỗi chính tả và chuẩn hóa từ vựng trong tên nguyên liệu thông qua một bảng từ điển sửa lỗi được định nghĩa sẵn.

Hàm hoạt động bằng cách tách tên nguyên liệu thành các từ đơn lẻ, sau đó kiểm tra từng từ với bảng `INGREDIENT_CORRECTIONS` - đây là một từ điển ánh xạ các từ viết sai chính tả hoặc các biến thể không chuẩn sang dạng chuẩn hóa. Ví dụ, nếu từ điển chứa ánh xạ "hànhla" → "hành lá", thì khi gặp từ "hànhla" trong tên nguyên liệu, hàm sẽ tự động thay thế thành "hành lá".

Quá trình này đảm bảo rằng các nguyên liệu có cùng bản chất nhưng được viết khác nhau do lỗi đánh máy, viết tắt không thống nhất, hoặc lỗi chính tả sẽ được chuẩn hóa về cùng một tên gọi chuẩn.

Điều này đặc biệt quan trọng trong việc làm sạch dữ liệu ẩm thực, nơi mà các lỗi chính tả và cách viết không đồng nhất có thể dẫn đến việc cùng một nguyên liệu bị phân loại thành nhiều mục khác nhau, gây khó khăn cho việc phân tích thống kê, đếm tần suất xuất hiện của nguyên liệu, và xây dựng hệ thống gợi ý công thức chính xác.

Hàm chỉ thực hiện thay thế khi tìm thấy từ trong từ điển sửa lỗi và giữ nguyên từ gốc nếu không tìm thấy hoặc giá trị thay thế là rỗng.

### Phân loại món ăn

Hàm `detect_category` được sử dụng để tự động phân loại các món ăn dựa trên tên gọi của chúng thông qua việc nhận diện các từ khóa đặc trưng.

Hàm hoạt động bằng cách so sánh tên món ăn (đã được chuyển về chữ thường) với một bảng ánh xạ các từ khóa đến thể loại tương ứng.

Cụ thể, hàm duyệt qua từng cặp từ khóa - thể loại trong bảng ánh xạ và kiểm tra xem từ khóa đó có xuất hiện trong tên món ăn hay không. Khi tìm thấy sự trùng khớp, hàm sẽ trả về thể loại tương ứng đã được viết hoa chữ cái đầu.

Ví dụ, nếu tên món có chứa từ "bún", "phở", "miến" hoặc "hủ tiếu", món ăn sẽ được phân loại là "Món nước"; nếu chứa từ "chiên" hoặc "rán" sẽ được xếp vào loại "Chiên".

Phương pháp phân loại này giúp tự động hóa việc gán nhãn thể loại cho hàng ngàn công thức nấu ăn, tạo điều kiện thuận lợi cho việc phân tích xu hướng ẩm thực, tìm kiếm và lọc món ăn theo thể loại, cũng như xây dựng hệ thống gợi ý món ăn dựa trên sở thích của người dùng.

## 2.2 Xây dựng Embedding

### 2.2.1 Giới thiệu chung về embedding

#### Embedding là gì?

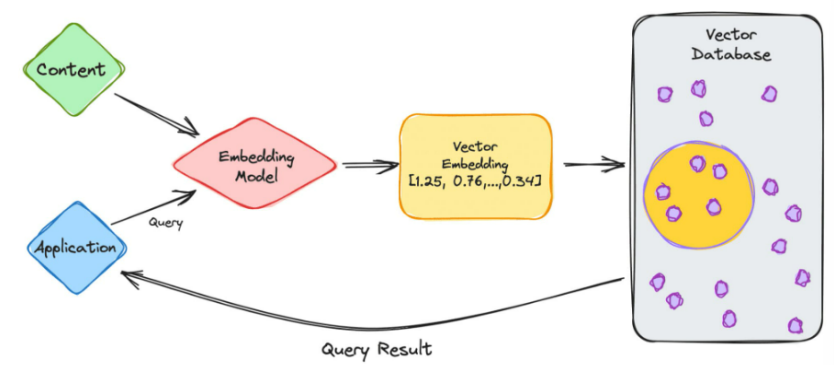
Embedding là một kỹ thuật trong lĩnh vực trí tuệ nhân tạo, đặc biệt trong học máy và xử lý ngôn ngữ tự nhiên (NLP), dùng để biểu diễn dữ liệu dạng vector trong không gian nhiều chiều. Thông qua embedding, các dữ liệu phức tạp như văn bản, hình ảnh hay âm thanh được chuyển đổi thành dạng số để máy tính có thể xử lý hiệu quả.

Embedding giúp tương quan giữa các dữ liệu được thể hiện dưới dạng khoảng cách hoặc góc trong không gian vector, từ đó hỗ trợ máy tính nhận diện, phân loại, tìm kiếm thông tin hoặc dự đoán kết quả. Ví dụ trong NLP, embedding giúp máy tính hiểu được nghĩa và mối quan hệ giữa các từ, thay vì chỉ nhìn vào ký tự hoặc chuỗi văn bản.

Trong phạm vi dự án này, embedding đóng vai trò trung tâm. Tất cả tên món ăn, nguyên liệu đều cần được chuyển đổi thành vector để hệ thống có thể tính toán độ tương đồng và đưa ra gợi ý chính xác.

#### Nguyên lý hoạt động của embedding

Embedding hoạt động dựa trên việc chuyển dữ liệu phân tán thành các vector số trong một không gian liên tục. Quá trình này diễn ra thông qua một mô hình học sâu được huấn luyện để nhận biết và mã hoá các đặc trưng quan trọng của dữ liệu. Khi học, mô hình điều chỉnh trọng số sao cho những đối tượng có tính chất hoặc ngữ nghĩa gần nhau được đặt gần nhau trong không gian véc-tơ, còn những đối tượng khác biệt sẽ xa nhau.



Hình 2.1: Nguyên lý hoạt động của Embedding

#### Các loại embedding phổ biến

Trong lĩnh vực trí tuệ nhân tạo, embedding không phải chỉ có một dạng duy nhất mà tồn tại dưới nhiều loại, mỗi loại phù hợp với loại dữ liệu và mục tiêu xử lý khác nhau.

- **Word embedding** là loại phổ biến nhất trong NLP, giúp biểu diễn các từ thành vector số để máy tính hiểu được mối quan hệ giữa các từ. Chẳng hạn như Word2Vec hay GloVe, những mô hình này giúp nhận diện từ đồng nghĩa, phân loại từ theo ngữ cảnh, và cải thiện hiệu quả các tác vụ ngôn ngữ như phân tích cảm xúc hay dịch máy.
- **Sentence embedding** và **document embedding** được dùng để biểu diễn câu hoặc cả tài liệu thành vector. Điều này cho phép các mô hình hiểu nghĩa của cả câu, đoạn văn hoặc tài liệu dài thay vì chỉ từng từ riêng lẻ. Ví dụ các mô hình như: Sentence-BERT, BGE-M3.
- **Image embedding** giúp chuyển đổi hình ảnh thành vector số. Vector này giữ các đặc trưng quan trọng của hình ảnh, cho phép máy tính nhận diện hình ảnh, tìm kiếm hình ảnh tương tự hoặc phân loại hình ảnh theo nội dung.
- **Audio embedding** được sử dụng để biểu diễn âm thanh, giọng nói hoặc nhạc thành vector số, hỗ trợ các ứng dụng nhận diện giọng nói, phân loại nhạc, hay phát hiện sự kiện âm thanh.

Mỗi loại embedding đều có ưu điểm riêng, nhưng điểm chung là chúng giúp máy tính hiểu và so sánh dữ liệu một cách thông minh hơn, từ đó nâng cao hiệu quả của các mô hình AI. Trong dự án này, nhóm sử dụng loại sentence embedding để encode tên món ăn và nguyên liệu.

### 2.2.2 Lựa chọn mô hình embedding

Bài toán của dự án yêu cầu một mô hình embedding với các tiêu chí sau:

- Hỗ trợ tiếng Việt tốt.
- Cho kết quả tương đồng ngữ nghĩa chính xác.
- Áp dụng được cho các câu ngắn như tên món ăn và nguyên liệu.

Sau khi tham khảo các mô hình hiện có, nhóm đã lựa chọn mô hình embedding BGE-M3 (BAAI General Embedding – Multi-Functionality, Multi-Linguality, Multi-Granularity).

#### Mô hình embedding BGE-M3

BGE-M3 là mô hình sentence embedding đa chức năng được xây dựng trên kiến trúc Transformer. Mô hình nổi bật với các đặc điểm sau:

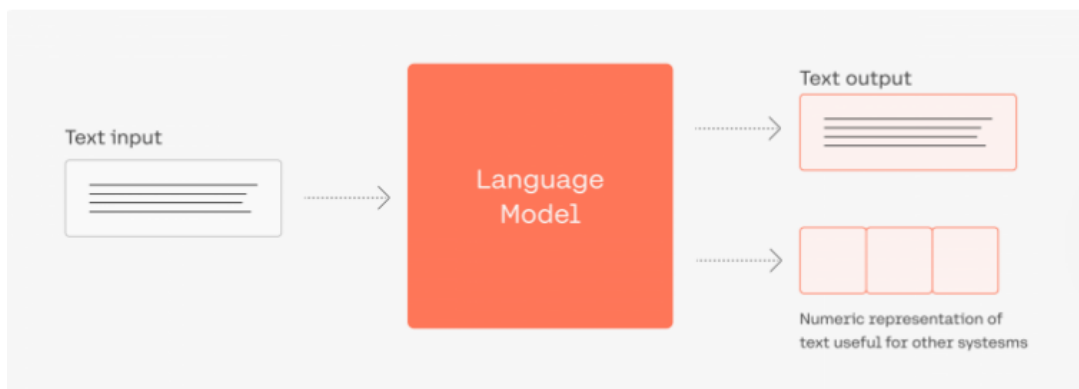
- **Đa ngôn ngữ:** Hỗ trợ hơn 100 ngôn ngữ, đạt kết quả state-of-the-art trong các nhiệm vụ truy xuất đa ngôn ngữ và xuyên ngôn ngữ.

- **Đa chức năng:** Có thể thực hiện đồng thời ba loại truy xuất phổ biến:
  - Dense retrieval (tìm kiếm vector dày đặc)
  - Sparse retrieval (tìm kiếm vector thưa)
  - Multi-vector retrieval (tìm kiếm đa vector)
- **Độ dài văn bản:** Xử lý đầu vào với độ dài khác nhau, từ câu ngắn đến tài liệu dài lên đến 8192 token.
- **Kỹ thuật huấn luyện:** Sử dụng phương pháp self-knowledge distillation, trong đó các điểm số mức độ liên quan từ các chức năng truy xuất khác nhau được tích hợp làm tín hiệu giáo viên (teacher signal), nâng cao chất lượng embedding.

### Kiến trúc Transformer trong BGE-M3

Để xử lý văn bản, mô hình BGE-M3 sử dụng kiến trúc Transformer, cho phép biểu diễn câu và đoạn văn dưới dạng vector ngữ cảnh. Cụ thể, Transformer thực hiện các bước sau:

- **Tokenization:** chia văn bản thành các token và mã hóa thành vector.
- **Embedding từng token:** mỗi token được biểu diễn dưới dạng vector số.
- **Self-Attention:** dựa vào tọa độ của vector trong không gian, mô hình học máy sẽ xem xét mối quan hệ giữa các token trong câu.
- **Encoder:** bộ mã hóa Transformer biến đổi các vector nhúng của tất cả các token thành một vector ngữ cảnh biểu diễn toàn câu.



Hình 2.2: Quy trình xử lý văn bản của kiến trúc Transformer

### 2.2.3 Triển khai embedding cho dữ liệu

Quy trình xây dựng embedding cho hệ thống chatbot gợi ý món ăn được triển khai như sau:

- Mô hình sử dụng dữ liệu đã được tiền xử lý từ file `recipes_cleaned.csv`, hai trường quan trọng cần embedding là `dish_name` và `ingredient_names`. Mục tiêu là chuyển hai cột này thành vector để phục vụ tìm kiếm theo độ tương đồng ngữ nghĩa.
- Mô hình BGE-M3 sẽ được tải bằng thư viện `SentenceTransformer` thông qua việc đọc thông số từ `config.yml`.
- Mỗi món ăn sẽ được encode thành một vector duy nhất, các nguyên liệu của mỗi món sẽ được encode riêng lẻ và lưu thành một mảng nhiều vector. Mỗi công thức được biểu diễn dưới dạng vector 1024 chiều.
- Dữ liệu sau khi embedding sẽ được lưu vào DataFrame với file `recipes_embeddings.pkl`.

Thông số	Giá trị	Mô tả
Model	BAAI/bge-m3	Multilingual embedding model
Embedding Dimension	1024	Số chiều của mỗi vector
Batch Size	32	Số samples xử lý mỗi lần
Normalization	L2 norm	Chuẩn hóa độ dài vector
Data Type	float32	Kiểu dữ liệu vector
Device	Auto-detect	CUDA/CPU

Bảng 2.1: Thông số kỹ thuật trong triển khai mô hình BGE-M3

### 2.2.4 So sánh BGE-M3 với PhoBERT

BGE-M3 là mô hình embedding đa ngôn ngữ từ Beijing Academy of AI (BAAI), phát hành năm 2024, hỗ trợ hơn 100 ngôn ngữ với kiến trúc Sentence Transformer dựa trên BERT, tối ưu cho tìm kiếm và truy xuất thông tin dựa trên độ tương đồng ngữ nghĩa. PhoBERT là mô hình BERT được huấn luyện sẵn dành riêng cho tiếng Việt từ VinAI Research, phát hành năm 2020, với phiên bản base có khoảng 135 triệu tham số. BGE-M3 nhắm đến embedding đa chức năng (dense, sparse, multi-vector), trong khi PhoBERT tập trung vào các nhiệm vụ NLP tiếng Việt như POS (gán nhãn loại từ trong câu) tagging và NER (nhận diện thực thể trong văn bản).

Về kích thước mô hình: BGE-M3 có khoảng 567 triệu tham số, lớn hơn nhiều so với PhoBERT (135 triệu tham số). Điều này cho phép BGE-M3 học và lưu trữ nhiều thông tin phong phú hơn, đặc biệt hữu ích cho semantic search (tìm kiếm ngữ nghĩa) và các tác vụ đa ngôn ngữ.

Thông số	BGE-M3	PhoBERT
Số tham số	~567M	~135M (base)
Embedding dimension	1024	768
Max sequence length	8192 tokens	256 tokens
Ngôn ngữ hỗ trợ	100+	Tiếng Việt
Tokenizer	SentencePiece	Syllable-based

Bảng 2.2: So sánh thông số kỹ thuật giữa BGE-M3 và PhoBERT

Về chiều vector embedding: Vector lớn hơn (1024 chiều của BGE-M3 so với 768 chiều của PhoBERT) giúp mô hình biểu diễn thông tin chi tiết và ngữ cảnh tốt hơn. Nhờ đó, độ chính xác khi tính tương đồng ngữ nghĩa được cải thiện, tuy nhiên chi phí bộ nhớ và tính toán cũng cao hơn.

Về hiệu suất và use case: BGE-M3 vượt trội trong retrieval đa ngôn ngữ và semantic search. Trong khi đó PhoBERT mạnh về hiểu morphology tiếng Việt, nhưng để dùng cho retrieval hoặc semantic search, cần fine-tuning và không hỗ trợ cross-lingual.

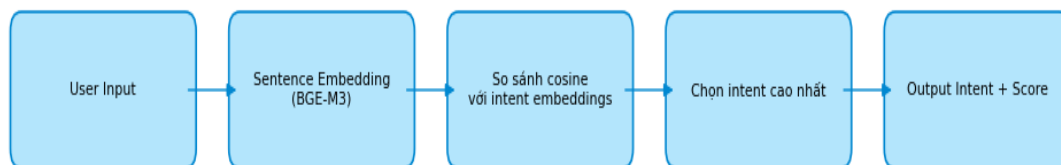
## 2.3 Xây dựng NLP

### 2.3.1 Xác định intent và trích xuất thông tin (Slot Extraction)

#### Xác định intent

##### *a. Mục tiêu*

Bước xác định intent nhằm phân loại chính xác ý định của người dùng khi tương tác với hệ thống. Cụ thể, hệ thống cần nhận biết liệu người dùng hướng tới việc tìm kiếm món ăn hay nhận hướng dẫn nấu ăn. Việc nhận diện intent chính xác tạo nền tảng để các bước tiếp theo - như trích xuất thông tin và gợi ý món ăn - hoạt động hiệu quả, từ đó nâng cao trải nghiệm người dùng và đảm bảo tính nhất quán trong phản hồi.



Hình 2.3: Pipeline xác định intent

##### *b. Thu thập và xây dựng tập dữ liệu Intent*

Để phục vụ bài toán phân loại ý định người dùng, hai nhóm intent chính được xác định gồm: `suggest_dishes` (gợi ý món ăn) và `cooking_guide` (hướng dẫn nấu ăn). Bộ dữ liệu được xây dựng theo quy trình bán tự động dựa trên mô hình ngôn ngữ và kiểm duyệt thủ công, bao gồm các bước:

1. **Xây dựng mô tả intent ban đầu.** Mỗi intent được mô tả rõ ràng về mục đích, phạm vi và cách diễn đạt đặc trưng.
2. **Sinh câu mẫu bằng mô hình ngôn ngữ.** ChatGPT được sử dụng để sinh các câu hỏi tự nhiên, đa dạng cấu trúc với prompt yêu cầu:
  - Câu ngắn, câu dài;
  - Câu có bối cảnh thực tế;
  - Câu paraphrase không trùng lặp cấu trúc;
  - Câu dễ gây nhầm lẫn giữa hai intent nhằm tăng độ bao phủ.
3. **Kiểm duyệt thủ công.** Các câu sinh ra được kiểm tra và loại bỏ:
  - Lỗi ngữ pháp, lỗi diễn đạt,
  - Câu không thuộc intent tương ứng,
  - Câu trùng lặp hoặc nhiễu.
4. **Mở rộng dữ liệu trong quá trình kiểm thử.**

Trong quá trình chạy thử hệ thống, những câu gây nhầm lẫn hoặc nhập bởi người dùng thực tế được bổ sung vào dataset. Điều này giúp mô hình thích nghi dần với dữ liệu thực và cải thiện độ chính xác.

Kết quả cuối cùng thu được bộ dữ liệu gồm khoảng 1.000 mẫu intent với mức độ đa dạng cao, đảm bảo đủ để xây dựng embedding mẫu đại diện cho từng nhóm ý định.

### ***c. Phương pháp***

Hệ thống sử dụng phương pháp **lai (hybrid)** giữa mô hình embedding ngữ nghĩa và luật dựa trên từ khóa. Mục tiêu là tận dụng khả năng hiểu ngữ nghĩa sâu của mô hình embedding, đồng thời bổ sung tính chắc chắn khi câu truy vấn chứa các từ khóa đặc trưng.

#### **1. Biểu diễn ngữ nghĩa bằng embedding**

Câu nhập của người dùng được mã hoá thành vector bằng mô hình **BGE-M3**. Mỗi intent có một tập embedding mẫu được tạo trước từ bộ dữ liệu đã xây dựng.

Độ tương đồng giữa câu nhập và các mẫu intent được đo bằng **cosine similarity**. Để giảm nhiễu, hệ thống sử dụng chiến lược *top-k mean*:

$$s_{\text{embed}} = \text{mean}(\text{top-k cosine})$$

Phương pháp này giúp đại diện tốt hơn cho toàn bộ intent thay vì phụ thuộc vào một câu mẫu duy nhất.

## 2. Thành phần rule-based từ từ khóa

Bên cạnh embedding, mỗi intent được gán một tập từ khóa mô tả hành vi ngôn ngữ phổ biến. Ví dụ:

- **cooking\_guide**: “hướng dẫn”, “cách làm”, “công thức”, “học nấu”, “dạy”, “chia sẻ” ...
- **suggest\_dishes**: “nấu món gì”, “gợi ý món”, “có nguyên liệu”, “món phù hợp”, “tìm món”, “đề xuất” ...

Nếu câu người dùng chứa các mẫu từ khóa này, hệ thống cộng thêm một điểm rule nhỏ:

$$s_{\text{rule}} \in [0, 0.1]$$

Điểm rule tuy nhỏ nhưng có tác dụng rõ rệt trong các câu ngắn hoặc câu chứa cấu trúc đặc trưng, nơi embedding có thể chưa phân biệt tốt.

## 3. Công thức kết hợp điểm

Điểm cuối cùng cho mỗi intent được tính bằng:

$$S = s_{\text{embed}} + \alpha \cdot s_{\text{rule}}$$

Trong đó  $\alpha = 0.1$  là hệ số điều chỉnh mức ảnh hưởng của rule. Intent có điểm số cao nhất được chọn làm kết quả phân loại.

### d. Triển khai

Quá trình triển khai được thực hiện trong backend theo các bước sau:

1. **Tiếp nhận câu truy vấn.** Câu người dùng được đưa vào pipeline mà không cần tiền xử lý phức tạp.
2. **Sinh embedding.** Mô hình BGE-M3 sinh ra vector biểu diễn ngữ nghĩa của câu.
3. **Tính toán top-k mean cosine.** Hệ thống so sánh embedding đầu vào với embedding mẫu của từng intent và lấy trung bình top-k.
4. **Áp dụng rule-based.** Tập từ khóa được quét bằng regex; nếu khớp thì cộng trọng số rule.
5. **Tổng hợp điểm và chọn intent.** Intent có điểm  $S$  cao nhất được trả về kèm theo bảng điểm chi tiết (phục vụ đánh giá).

Cách triển khai này đảm bảo:

- Linh hoạt khi mở rộng thêm intent mới,
- Dễ điều chỉnh bằng cách thay đổi mức trọng số hoặc danh sách từ khóa,
- Độ chính xác cao khi áp dụng vào câu truy vấn tiếng Việt tự nhiên.

### Trích xuất thông tin (Slot Extraction)

#### *a. Mục tiêu*

- Trích xuất các thông tin quan trọng từ câu nhập của người dùng, phục vụ cho việc gợi ý món ăn và hướng dẫn nấu ăn.
- Các slot chính gồm:
  - **ingredient**: Nguyên liệu (ví dụ: gà, khoai tây, cà rốt)
  - **category**: Danh mục món ăn (ví dụ: kho, xào, luộc)
  - **dish\_name**: Tên món ăn cụ thể (ví dụ: bánh mì chiên bơ tỏi)
  - **time**: Thời gian nấu (ví dụ: 45 phút, 2 giờ)
  - **difficulty**: Độ khó (dễ, trung bình, khó)
  - **serving**: Số khẩu phần (ví dụ: 4 người, 2 phần)
- Mục tiêu là nhận diện chính xác các slot, tránh nhầm lẫn substring, và xử lý câu nhập tự nhiên, dài, hoặc nhiều biến thể ngôn ngữ.

#### *b. Phương pháp*

- **Tokenization**: là quá trình chia nhỏ văn bản thành các đơn vị nhỏ hơn gọi là token. Token có thể là:
  - Word-level: chia theo từ, ví dụ: “Tôi muốn ăn cơm” → [Tôi, muốn, ăn, cơm]
  - Character-level: chia theo ký tự, ví dụ: “cơm” → [c, o, m]
  - Subword / BPE: ví dụ “cooking” → [cook, ing]
  - Sentence-level: chia câu, ví dụ “Tôi thích NLP. Nó rất thú vị.” → [Tôi thích NLP., Nó rất thú vị.]

Mục đích: chuẩn hóa dữ liệu, tách từ, loại bỏ dấu câu, token hóa cho embeddings hoặc mô hình NLP.

Thư viện phổ biến: Bảng dưới đây liệt kê các thư viện tokenization phổ biến:

**Giải thích và lý do chọn ViTokenizer:**

Thư viện	Ngôn ngữ	Cách hoạt động
spaCy	Python	NLP pipeline, nhận diện từ, số, tên riêng, chuẩn hóa văn bản
ViTokenizer	Python	Tokenize tiếng Việt theo từ, dictionary + rule-based, xử lý từ ghép chuẩn
BERT / HuggingFace	Python	Subword tokenization (BPE / WordPiece), tạo token ID cho embeddings

Bảng 2.3: Các thư viện tokenization phổ biến trong dự án

- **spaCy**: mạnh mẽ cho NLP đa ngôn ngữ, nhận diện từ, số, tên riêng, nhưng không tối ưu cho tiếng Việt.
  - **ViTokenizer**: chuyên biệt cho tiếng Việt, xử lý từ ghép, âm tiết nhiều dấu chuẩn, phù hợp với hệ thống nhận diện slot (nguyên liệu, tên món, category). Đây là lựa chọn chính trong dự án.
  - **BERT / HuggingFace Tokenizers**: hỗ trợ subword tokenization, chuẩn cho embeddings và mô hình Transformer, dùng kết hợp sau khi tokenize để tính vector đại diện.
- **N-grams**: là các tập con liên tiếp của token trong văn bản, từ 1 đến N token.
    - Giúp nhận diện các cụm từ dài và cố định trong ngôn ngữ, ví dụ: “khoai tây”, “bánh mì chiên bơ tỏi”.
    - Ưu tiên các n-grams dài hơn để tránh nhầm lẫn với các từ đơn lẻ.
    - Trong hệ thống NLP, N-grams kết hợp với danh sách từ điển giúp phát hiện chính xác các slot (nguyên liệu, tên món) ngay cả khi token đơn lẻ không đủ thông tin.
  - **Fuzzy Matching (So khớp gần đúng)**: là phương pháp so khớp token hoặc n-grams với từ điển mà không yêu cầu trùng tuyệt đối.
    - Xử lý biến thể ngôn ngữ, lỗi chính tả hoặc cách viết khác nhau.
    - Dựa trên *similarity ratio* giữa token/n-gram và từ điển; chỉ những kết quả đạt ngưỡng similarity mới được coi là match.
    - Ví dụ: “khoai tây” vẫn có thể được nhận diện là “khoai tây”.
  - **Rule-based Extraction (Trích xuất theo quy tắc)**: dùng các quy tắc, biểu thức chính quy (regex) hoặc từ khóa đặc trưng để trích xuất thông tin.
    - Phù hợp với các slot có cấu trúc chuẩn: thời gian nấu, số khẩu phần, độ khó.
    - Ví dụ:
      - \* “2 giờ 30 phút” → thời gian nấu 150 phút.

- \* “4 người” → số khẩu phần 4.

– Kết hợp với tokenization và n-grams giúp nhận diện các slot chính xác hơn.

### *c. Triển khai*

#### – Tokenization:

- \* Sử dụng **ViTokenizer** để tokenize câu nhập tiếng Việt:

```
tokens = ViTokenizer.tokenize(text).split()
```

- \* Loại bỏ stopwords và ký tự đặc biệt trước khi xử lý tiếp.

#### – N-gram generation:

- \* Tạo các n-grams từ 1 đến 6 token để nhận diện các cụm từ dài (multi-word slot).
- \* Sắp xếp giảm dần theo độ dài để ưu tiên match các cụm từ dài trước.

#### – Nguyên liệu (Ingredients):

- \* So khớp các n-grams với danh sách nguyên liệu.
- \* Dùng fuzzy matching để nhận diện biến thể và lỗi chính tả.
- \* Loại bỏ trùng lặp sau khi match.

#### – Category:

- \* Sử dụng regex với ranh giới từ để nhận diện category chính xác, tránh substring nhầm: ‘
- \* Ưu tiên match cụm từ dài trước, tránh nhầm lẫn như "kho" trong "khoai".

#### – Tên món:

- \* Dùng n-grams kết hợp fuzzy matching với threshold để nhận diện tên món gần đúng.
- \* Giúp hệ thống nhận diện tên món ngay cả khi người dùng viết sai chính tả.

#### – Thời gian, khẩu phần, độ khó:

- \* Regex nhận diện các slot đặc thù:
  - Thời gian: “2 giờ 30 phút” → 150 phút.
  - Khẩu phần: “4 người” → 4.
  - Độ khó: mapping từ khóa “dễ, trung bình, khó”.

#### – Gom slot theo intent:

- \* `suggest_dishes`: category, ingredient, time, difficulty, serving.
- \* `cooking_guide`: dish\_name.

– **Ưu điểm:**

- \* Trích xuất chính xác các slot quan trọng.
- \* Kết hợp token-level + regex giảm nhầm lẫn substring.
- \* Fuzzy matching giúp nhận diện tên món gần đúng.

– **Nhược điểm và cải tiến:**

- \* Category và nguyên liệu hiện tại dựa vào từ điển → không nhận diện từ mới.
- \* Có thể kết hợp embedding + cosine similarity để xử lý câu nhập tự nhiên hơn.
- \* Áp dụng rule ranking hoặc confidence score để ưu tiên slot đáng tin cậy nhất.

### 2.3.2 Xây dựng hệ thống tìm kiếm với FAISS

#### Tổng quan về FAISS

FAISS (Facebook AI Similarity Search) là thư viện mã nguồn mở hỗ trợ tìm kiếm tương đồng và phân cụm trên các vector dense một cách hiệu quả. Thư viện cung cấp các thuật toán tìm kiếm trên tập vector lớn, kể cả khi vượt quá bộ nhớ RAM, đồng thời cung cấp công cụ đánh giá và tinh chỉnh tham số [1].

FAISS được phát triển bằng C++ với wrapper Python đầy đủ, nhiều thuật toán được tối ưu hóa cho GPU để tăng tốc xử lý dataset lớn. Thư viện được phát triển chủ yếu bởi nhóm FAIR (Facebook AI Research) của Meta.

#### Nhu cầu sử dụng FAISS trong hệ thống tìm kiếm

Trong các hệ thống retrieval-based, tìm kiếm dựa trên từ khóa truyền thống (BM25, TF-IDF) gặp hạn chế do không hiểu ngữ nghĩa của truy vấn. Vector embedding từ các mô hình NLP/ML (như BERT, BGE, CLIP) biểu diễn dữ liệu và query trong không gian vector, giúp giải quyết vấn đề này.

FAISS cung cấp:

- Tìm kiếm nhanh các vector tương đồng (Exact/Approximate Nearest Neighbor).
- Giảm bộ nhớ nhờ kỹ thuật nén và index phân lớp (IVF, PQ).
- Hỗ trợ GPU tăng tốc với dataset lớn.

Nhờ đó, FAISS phù hợp cho semantic search, recommendation system, và retrieval-augmented generation (RAG) trong chatbot [2, 3].

### Kiến trúc và thành phần chính của FAISS

Các thành phần chính:

- **Index:** Lưu trữ và truy xuất vector; xác định chiến lược exact hay approximate.
- **Metric:** Đo khoảng cách/similarity, phổ biến:
  - **L2 Distance (Euclidean):**  $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$
  - **Inner Product (Dot Product):**  $\text{similarity}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$
- **Quantizer:** Lượng tử hóa vector trong approximate search (PQ).
- **Inverted File (IVF):** Chia không gian vector thành cell để tìm kiếm nhanh.
- **GPU module:** Tăng tốc xử lý dataset lớn.

### Các loại Index trong FAISS

#### Exact Search

- **IndexFlatIP:** Inner product.
- **IndexFlatL2:** L2 distance.
- **Ưu điểm:** Độ chính xác 100%.
- **Nhược điểm:** Tốn tính toán dataset lớn.

#### Approximate Search

- **IndexIVFFlat:** Chia cluster bằng K-means, cần training.
- **IndexIVFPQ:** IVF + Product Quantization, tiết kiệm bộ nhớ.
- **HNSW:** Đồ thị phân cấp, không cần training, hiệu suất cao CPU/GPU.

#### GPU Indexes

- **GPUFlat, GPUIVF:** Chạy trên GPU, tăng tốc truy vấn.

### Cơ chế hoạt động

1. **Embedding:** Chuyển ingredients và dish names thành các vector dense.
2. **Xây dựng Index:** Thêm vector vào FAISS Index.
3. **Truy vấn:** Mã hóa input của user, tìm top-k vectors gần nhất.
4. **Filtering & Ranking:** Xếp hạng các món ăn dựa trên similarity score và các tiêu chí bổ sung.

## Triển khai FAISS trong hệ thống

### Lựa chọn Index cho hệ thống

Hệ thống sử dụng **IndexFlatIP** vì:

- Dataset vừa phải (ingredients và dish names).
- Độ chính xác cao, inner product tương đương cosine similarity với embedding chuẩn hóa.
- Triển khai đơn giản, không cần training.

$$\text{top\_k} = \arg \max_{i \in [1, N]} (\mathbf{q} \cdot \mathbf{v}_i)$$

### Xây dựng FAISS Index

1. Mã hóa ingredients và dish names thành embedding vectors.
2. Thêm embedding vào IndexFlatIP.
3. Index lưu trữ vectors và trả về ID để mapping với dataset gốc.

### Tìm kiếm thời gian thực

1. Mã hóa input (ingredients/dish) thành vector.
2. Tìm top-k vectors gần nhất từ FAISS Index.
3. Lấy thông tin món ăn từ dataset gốc dựa trên ID.

### Kết quả cuối cùng

- Kết hợp similarity score và các điểm số khác để xếp hạng các món ăn.
- Lấy top-k món ăn có similarity score cao nhất.

## Ưu điểm và hạn chế

### Ưu điểm

- Tốc độ cao, hỗ trợ exact/approximate search.
- Hỗ trợ GPU, tăng tốc dataset lớn.
- Đa dạng index, dễ điều chỉnh.
- Dễ tích hợp với embedding từ text/hình ảnh/audio.
- Batch search tối ưu throughput.

## Hạn chế

- Không tối ưu với dữ liệu sparse.
- Một số approximate index có recall thấp hơn exact.
- Index lớn tốn nhiều RAM nếu không dùng IVF/PQ/GPU.

## Ứng dụng thực tế

- Chatbot / Search Engine: retrieval-based, RAG.
- Recommendation System: gợi ý sản phẩm/dish dựa trên embedding.
- Image Similarity: tìm ảnh tương tự bằng embedding CLIP/CNN.
- Semantic Search NLP: tìm văn bản, đoạn hội thoại gần nghĩa.
- Multimodal Search: tích hợp text, hình ảnh, audio.

## 2.4 Xây dựng LLM

### 2.4.1 Lựa chọn mô hình

*So sánh mô hình LLaMA 3.1 8B instant, mô hình local LLaMA và mô hình GPT-4/GPT-3.5:*

Tiêu chí	LLaMA 3.1 8B Instant	local LLaMA	GPT-4 / GPT-3.5
Ngôn ngữ	Chủ yếu tiếng Anh, tiếng Việt cần prompt kỹ lưỡng	Tùy mô hình: nếu pretrain/fine-tune có thể hỗ trợ tiếng Việt	Hỗ trợ đa ngôn ngữ tốt, bao gồm tiếng Việt
Hiệu năng	Rất nhanh nhờ AI accelerator của Groq	Phụ thuộc GPU/CPU; chậm nếu phần cứng yếu	Nhanh nhưng phụ thuộc API và giới hạn request
Triển khai	Dễ dàng qua LangChain + API	Khó, cần GPU mạnh và setup phức tạp	Dễ triển khai qua API, không cần hạ tầng phức tạp
Tùy chỉnh	Hạn chế, chủ yếu điều khiển qua prompt	Cao, có thể fine-tune, retrain hoặc thêm dữ liệu riêng	Hạn chế; chủ yếu dùng prompt hoặc “system message”
Chi phí	Trả phí API	Chi phí phần cứng (GPU/CPU mạnh)	Trả phí API theo gói sử dụng
Multi-turn	Hỗ trợ tốt qua LangChain	Khó hơn, cần quản lý context thủ công	Hỗ trợ tốt, duy trì context qua nhiều lượt chat
Phụ thuộc internet	Có	Không	Có
Độ tin cậy	Cao nếu dữ liệu vector chuẩn	Tùy chất lượng fine-tune	Cao, nhưng vẫn có khả năng hallucination nếu prompt mập mờ

Bảng 2.4: So sánh các mô hình LLaMA 3.1 8B instant, LLaMA Local và GPT-4, GPT-3.5

Từ bảng so sánh trên, thông qua phân tích các ưu – nhược điểm về chi phí và yêu cầu cấu hình phần cứng, mô hình phù hợp nhất là mô hình có khả năng cung cấp API miễn phí, không giới hạn và không đòi hỏi hệ thống phần cứng mạnh để triển khai. Dựa trên các tiêu chí này, mô hình ngôn ngữ lớn LLaMA 3.1 8B Instant được ưu tiên lựa chọn nhằm thực hiện nhiệm vụ sinh phản hồi tự động. Các phản hồi được tạo ra dựa trên dữ liệu đầu vào do người dùng cung cấp, sau khi đã được xử lý tại bước NLP.

Việc triển khai được thực hiện thông qua framework LangChain, giúp quản lý toàn bộ pipeline xử lý, xây dựng prompt và tích hợp dữ liệu truy xuất từ hệ cơ sở dữ liệu vector. Mô hình được gọi thông qua Groq API (nền tảng ChatGroq), tận dụng khả năng

tăng tốc xử lý vượt trội của phần cứng chuyên dụng, giúp giảm độ trễ và cải thiện tốc độ phản hồi.[4]

## 2.4.2 Thực hiện mô hình

Mô hình nhận đầu vào là dữ liệu đã được xử lý và chuẩn hóa từ bước NLP, bao gồm danh sách nguyên liệu, tên món ăn hoặc các yêu cầu liên quan đến món ăn mà người dùng cung cấp. Nếu bước truy xuất dữ liệu không tìm thấy kết quả phù hợp (kết quả rỗng), hệ thống sẽ phản hồi bằng một thông báo hài hước nhằm tăng tính tương tác với người dùng. Trong trường hợp có ít nhất một kết quả phù hợp, hệ thống sẽ xây dựng các thông điệp đầu vào (messages) cho LLM.

Hai loại thông điệp được sử dụng bao gồm:

- **SystemMessage**: LLM được vào vai một chuyên gia ẩm thực gen Z Việt Nam. Các kết quả trả về của LLM phải có giọng điệu dí dỏm, hài hước, đồng thời nội dung câu trả lời phải đảm bảo tính ngắn gọn, rõ ràng và độ chính xác cao. Việc quy định những điều trên giúp mô hình gần gũi hơn với các bạn trẻ, phần nào giúp các bạn tăng thêm hứng thú hoặc trở nên hứng thú với việc nấu ăn, tạo thói quen chăm sóc bản thân bằng nguồn dinh dưỡng đảm bảo sạch sẽ và vệ sinh hơn.
- **HumanMessage**: LLM nhận đầu vào là dữ liệu đã được xử lý từ NLP, sau đó LLM sử dụng những thông tin này để sinh phản hồi cho người dùng theo những yêu cầu và quy tắc đã quy định trong **SystemMessage**.

LLM được gọi bằng phương thức `chat.invoke(messages)`, trả về nội dung phản hồi dưới dạng văn bản. Toàn bộ quy trình được đóng gói trong khối `try-except` nhằm đảm bảo tính ổn định. Nếu không xảy ra lỗi kết nối API, mô hình sẽ sinh tự động câu trả lời chứa danh sách kết quả phù hợp với yêu cầu từ người dùng. Trong trường hợp xảy ra lỗi kết nối API, mô hình sẽ trả về danh sách dữ liệu đã được xử lý từ NLP, không trả về bất cứ phản hồi được sinh tự động nào.

Nhờ cách thiết kế các hàm xử lý chuyên biệt như `describe_dishes` và `smooth_instructions` mà hệ thống có khả năng đáp ứng đa dạng nhu cầu truy vấn của người dùng. Đồng thời, mỗi phản hồi đều mang phong cách trẻ trung, hài hước và gần gũi, tạo sự thân thiện và tăng trải nghiệm tương tác:

- **describe\_dishes**: Sinh phản hồi về mô tả món ăn.
- **smooth\_instructions**: Sinh phản hồi về hướng dẫn nấu món ăn.

Đặc biệt, việc giữ nguyên danh sách món ăn được cung cấp từ bước NLP và yêu cầu mô hình “không được tự bịa món mới” giúp đảm bảo tính chính xác và nhất quán giữa

dữ liệu NLP và phản hồi từ LLM. Điều này hạn chế hiện tượng hallucination (ảo giác AI) và nâng cao độ tin cậy của kết quả.

Tổng thể, phần triển khai LLM đóng vai trò trung tâm trong hệ thống, kết nối giữa dữ liệu đã được xử lý và giao diện hội thoại với người dùng, đồng thời mang lại trải nghiệm tự nhiên, vui tươi và hiện đại đúng với định hướng của trợ lý ảo thông minh.

[5]

# Chương 3

## Kết quả & Phân tích

### 3.1 Đánh giá mô hình Embedding

#### 3.1.1 Tổng quan

Trong dự án, mô hình cần embedding 2 trường thông tin chính là tên món ăn và nguyên liệu. Đánh giá được thực hiện trên hai tập dữ liệu chính:

- Món ăn: 2,405 tên món ăn tiếng Việt
- Nguyên liệu: 2,396 loại nguyên liệu

Mục tiêu là kiểm tra khả năng tìm kiếm ngữ nghĩa (semantic search) của mô hình với các truy vấn tiếng Việt, đặc biệt là khả năng xử lý các cụm từ ngắn, từ đơn lẻ và các biến thể của tên món ăn/nguyên liệu.

#### 3.1.2 Phương pháp đánh giá

##### Quy trình

- **Tạo embeddings:** Chuyển đổi toàn bộ tên món ăn và nguyên liệu thành vector 1024 chiều.
- **Tìm kiếm tương đồng:** Sử dụng cosine similarity để tìm các kết quả phù hợp nhất.
- **Đo lường hiệu suất:** Sử dụng các chỉ số Accuracy@1, Accuracy@5 và MRR.

##### Các chỉ số đánh giá

- **Accuracy@1:** Tỷ lệ truy vấn có kết quả đúng ở vị trí đầu tiên.
- **Accuracy@5:** Tỷ lệ truy vấn có kết quả đúng trong top 5.

- **MRR (Mean Reciprocal Rank):** Trung bình nghịch đảo của vị trí kết quả đúng đầu tiên.

### 3.1.3 Kết quả đánh giá

#### Hiệu suất tổng thể

Khi thực hiện đánh giá cho 10 món ăn và 10 nguyên liệu bất kỳ:

- **Món ăn:** thịt gà, cơm gà, phở bò, bún chả, gỏi cuốn, bánh mì, cá hồi, tôm, lẩu, salad.
- **Nguyên liệu:** trứng, hành tây, cà rốt, khoai tây, nấm, tỏi, gừng, ớt, rau, thịt bò.

Mô hình BGE-M3 cho ra kết quả như sau:

Chỉ số	Món ăn	Nguyên liệu
Accuracy@1	100% (10/10)	100% (10/10)
Accuracy@5	100% (10/10)	100% (10/10)
MRR	1.0000	1.0000

#### Phân tích món ăn

**Kết quả tổng quan:**

- **Khớp chính xác:** 7/10 truy vấn, mô hình trả về đúng món ăn với  $similarity = 1.0$  (ví dụ: *cơm gà, phở bò*).
- **Khớp một phần:** 3/10 truy vấn, kết quả top 1 là biến thể mở rộng của món tìm kiếm.

**Ví dụ minh họa:**

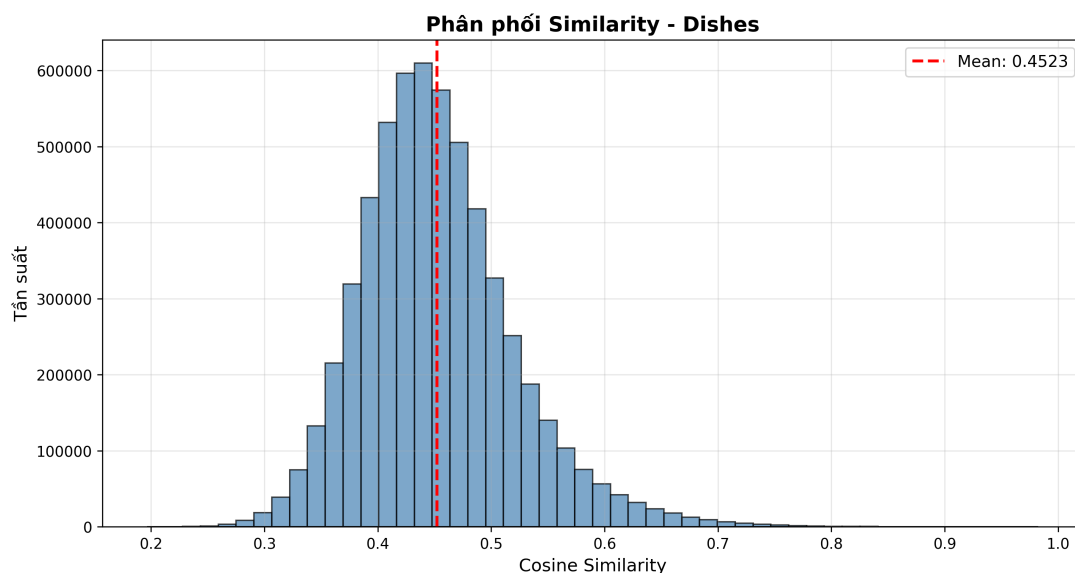
- **Truy vấn:** “thịt gà”  
**Top 1:** “thịt gà nướng” ( $similarity = 0.8638$ )  
**Nhận xét:** Hợp lý vì đây là món cụ thể chứa nguyên liệu “thịt gà”.
- **Truy vấn:** “gỏi cuốn”  
**Top 1:** “gỏi cuốn tôm” ( $similarity = 0.8975$ )  
**Nhận xét:** Top 5 đều là các biến thể của gỏi cuốn (tôm, cá hồi, tôm chiên, ...).
- **Truy vấn:** “bún chả”  
**Top 1:** “bún chả cá” ( $similarity = 0.8898$ )  
**Nhận xét:** Món “bún chả” có thể không tồn tại trong cơ sở dữ liệu, nên mô hình chọn món gần nhất về ngữ nghĩa.

Phân tích nguyên liệu**Kết quả tổng quan:**

- **Hiệu suất tốt:** 10/10 truy vấn đều trả về đúng nguyên liệu với  $similarity = 1.0$ .
- **Các truy vấn:** trứng, hành tây, cà rốt, khoai tây, nấm, tỏi, gừng, ớt, rau, thịt bò.

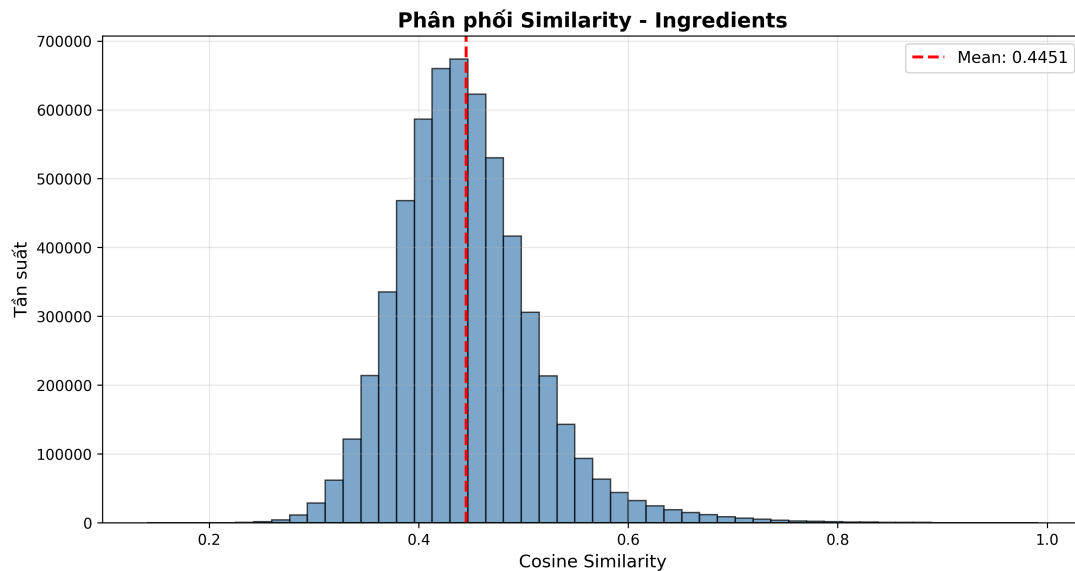
**Nhận xét:**

- Mô hình xử lý rất tốt các nguyên liệu phổ biến, đơn lẻ và không có nhiều biến thể trong tiếng Việt.
- Đây là nhóm dữ liệu ít đa dạng hơn so với món ăn, giúp mô hình đạt độ chính xác tối đa.

Phân tích phân phối similarity

Hình 3.1: Phân phối similarity món ăn

Phân phối tương đối đồng đều với độ lệch chuẩn thấp (0.0678). Giá trị trung bình (0.45) cho thấy các món ăn có sự khác biệt vừa phải trong không gian embedding. Khoảng cách rõ rệt giữa các món hoàn toàn khác biệt (min: 0.1964) và các biến thể của cùng món (max: 0.9825)



Hình 3.2: Phân phối similary nguyên liệu

Tương tự món ăn, nguyên liệu có phân phối ổn định. Độ tương đồng cao nhất (0.9914) xuất hiện ở các cặp nguyên liệu chỉ khác thứ tự từ.

### Phân tích các cặp tương đồng

- Trong top 10 cặp món ăn giống nhau nhất, các cặp có similarity > 0.94 chủ yếu là biến thể ngôn ngữ: thay đổi nhẹ từ ngữ, đảo thứ tự từ hoặc từ đồng nghĩa.
  - “cupcake trứng muối chà bông phô mai” ↔ “bánh cupcake trứng muối chà bông phô mai”
  - “gyoza mì trộn” ↔ “mì trộn gyoza”

Mô hình nắm bắt tốt các biến thể tiếng Việt và không phụ thuộc vào thứ tự từ.

- Các món ăn hoàn toàn khác nhau có similarity rất thấp, cho thấy mô hình phân biệt rõ ràng giữa các nhóm món ăn không liên quan.
- Với nguyên liệu, các cặp giống nhau nhất chủ yếu là đảo thứ tự từ, cho thấy mô hình nhận diện tốt ngay cả khi cấu trúc câu thay đổi nhưng ngữ nghĩa giữ nguyên.

### Kết luận

Qua quá trình đánh giá, mô hình BGE-M3 đã cho thấy hiệu suất tốt trên dữ liệu tiếng Việt về ẩm thực mà nhóm hiện có. Mô hình thể hiện khả năng hiểu ngữ nghĩa tốt và xử lý hiệu quả các biến thể ngôn ngữ trong tiếng Việt.

Mô hình có khả năng nhận diện mối quan hệ ngữ nghĩa giữa các món ăn và nguyên liệu, đồng thời xử lý tốt các cách diễn đạt khác nhau của cùng một khái niệm. Kết quả

đánh giá cho thấy BGE-M3 là một lựa chọn phù hợp cho việc áp dụng vào dự án chatbot gợi ý nấu ăn.

## 3.2 Hiệu suất xác định Intent & Slot Extraction

### 3.2.1 Xác định intent

Trong dự án, hệ thống chỉ cần nhận diện hai intent chính:

- **Hướng dẫn nấu ăn:** Người dùng hỏi cách chế biến một món cụ thể.
- **Gợi ý món ăn:** Người dùng đưa nguyên liệu sẵn có và thời gian nấu, khẩu phần ăn, độ khó hay thể loại và muốn hệ thống gợi ý món phù hợp.

Khi kiểm thử trên tập dữ liệu gồm 100 câu hỏi mẫu, kết quả phân loại intent như sau:

Intent	Số câu đúng / Tổng số	Tỷ lệ chính xác
Hướng dẫn nấu ăn	64 / 66	97%
Gợi ý món ăn	35 / 39	90%
<b>Trung bình</b>	<b>99 / 105</b>	<b>94%</b>

Bảng 3.1: Kết quả phân loại intent

### 3.2.2 Slot Extraction

Đối với mỗi intent, hệ thống cần trích xuất các slot quan trọng:

- Với intent **Hướng dẫn nấu ăn:** tên món ăn.
- Với intent **Gợi ý món ăn:** danh sách nguyên liệu.

Kết quả kiểm thử:

Slot	Số câu đúng / Tổng số	Tỷ lệ chính xác
Tên món ăn	100 / 105	91%
Nguyên liệu	99 / 105	90%
<b>Trung bình</b>	<b>99 / 105</b>	<b>90%</b>

Bảng 3.2: Kết quả trích xuất slot

### 3.2.3 Phân tích

Hệ thống hoạt động tốt với intent **Hướng dẫn nấu ăn** vì dữ liệu huấn luyện rõ ràng. Với intent **Gợi ý món ăn**, độ chính xác thấp hơn một chút do cách viết nguyên liệu không thống nhất (ví dụ: “cà rốt” và “carot”).

Slot extraction đạt độ chính xác cao, nhưng vẫn cần cải thiện từ điển chuẩn hoá nguyên liệu để tránh lỗi viết tắt hoặc sai chính tả.

**Kết luận:** Với chỉ hai intent, hệ thống NLP đạt độ chính xác trung bình trên 90%, đủ để chatbot hiểu đúng nhu cầu người dùng trong hầu hết trường hợp.

## 3.3 Hiệu suất tìm kiếm với FAISS

### 3.3.1 Thông tin Index

Hệ thống sử dụng hai index chính cho việc tìm kiếm:

- **Dish index:** 2,420 vectors, dimension = 1,024.
- **Ingredient names index:** 15,752 vectors, dimension = 1,024.

Nhận xét: Dataset dish nhỏ nên index đơn giản, truy vấn nhanh; dataset names lớn hơn nên cần cân nhắc tối ưu thêm.

### 3.3.2 Hiệu suất truy vấn

Index	Avg time/query (ms)	QPS
Dish	1.10	910
Names	3.85	260

Bảng 3.3: Hiệu suất truy vấn của các FAISS Index

Nhận xét:

- Dish index cho phép truy vấn gần như real-time với throughput cao.
- Names index truy vấn chậm hơn do số lượng vector lớn hơn, cần tối ưu nếu mở rộng dataset.

### 3.3.3 Độ chính xác

Nhận xét:

- Dish index đạt độ chính xác tuyệt đối, phù hợp với truy vấn theo tên món ăn.
- Names index có precision và recall thấp, cần cải thiện embedding hoặc kết hợp với re-ranking/approximate index để tăng hiệu quả.

Index	Precision@5	Recall@5	Test cases
Dish	100%	100%	100
Names	24%	24%	100

Bảng 3.4: Độ chính xác của các FAISS Index

### 3.3.4 Phân tích

- Với index nhỏ (dish), Exact search (`IndexFlatIP`) hoạt động tối ưu về tốc độ và accuracy.
- Với index lớn (names), cần cân nhắc sử dụng Approximate search (IVF, PQ) hoặc hybrid với re-ranking để cải thiện recall.
- Hệ thống hiện tại phù hợp cho retrieval món ăn, nhưng còn hạn chế khi tìm kiếm theo tên nguyên liệu phức tạp.

## 3.4 Hiệu suất và độ chính xác của LLM

### 3.4.1 Hiệu suất

Mô hình LLaMA 3.1 8B Instant cho tốc độ phản hồi rất nhanh và ổn định nhờ chạy trên hạ tầng Groq, nơi sử dụng bộ tăng tốc ngôn ngữ (LPU) tối ưu cho xử lý các tác vụ sinh văn bản.

Do toàn bộ quá trình suy luận được thực hiện qua Groq API, hệ thống không phụ thuộc vào GPU cục bộ nên độ trễ không bị ảnh hưởng bởi cấu hình phần cứng của máy chạy backend.

Trong quá trình thử nghiệm, khi gọi hàm `describe_dishes()` với danh sách chứa 3 món ăn trở lên và truy vấn liên tiếp 5 lần, thời gian phản hồi trung bình chỉ khoảng 4–6s mỗi yêu cầu, không xuất hiện độ trễ tăng dần theo số lượng truy vấn.

Bên cạnh đó, toàn bộ các lời gọi API đều được bao bọc trong khối try–except, giúp hệ thống giữ trạng thái ổn định khi xảy ra lỗi mạng và tự động chuyển sang chế độ trả về nội dung dự phòng, đảm bảo trải nghiệm mượt mà cho người dùng.

### 3.4.2 Độ chính xác

Mô hình được kiểm soát thông qua hệ thống luật trong `SystemMessage`, giúp giảm thiểu việc sinh nội dung không liên quan hoặc bịa đặt do hiện tượng ảo giác AI.

Ví dụ, khi NLP trả về danh sách món gồm: “Disum hải sản”, “Sụn heo xáo nghệ”, “Chả tôm thịt mắc khén”, mô hình chỉ được phép mô tả hoặc hỏi lại trong phạm vi ba món này.

Trong thử nghiệm, khi người dùng nhập yêu cầu "Tôi muốn nấu các món có thịt heo, nấu trong 30 phút", mô hình trả về mô tả chính xác: "Bạn có thịt heo đúng không? Tôi nghĩ bạn sẽ thích Sụn heo xào nghệ. Nguồn hương vị truyền thống của món này rất đặc trưng. Sụn được nấu mềm tỏng hỗn hợp nghệ, muối và các nguyên liệu khác, tạo nên một hương vị độc đáo, khó cưỡng."

Mô hình không tạo ra món mới như "Canh chua tôm" hoặc "Canh kimchi", thể hiện việc tuân thủ đúng luật và bám sát dữ liệu có thật. Ở hàm `smooth_instructions()`, mô hình cũng thể hiện độ chính xác tốt khi viết lại hướng dẫn nấu theo đúng giới hạn số lượng chữ và không đánh số thứ tự, kể cả khi dữ liệu gốc chứa nhiều ký tự thừa hoặc có đánh số.

Nhờ sự kết hợp giữa kiểm soát chặt chẽ của `SystemMessage`, dữ liệu NLP đầu vào chính xác và khả năng suy luận nhanh của hạ tầng Groq, mô hình duy trì được cả hiệu suất cao và độ chính xác ổn định, đáp ứng yêu cầu của một hệ thống gợi ý ẩm thực thân thiện và mang tính ứng dụng cao.

# Chương 4

## Kết luận

Hệ thống CookGPT đã chứng minh hiệu quả trong việc hỗ trợ người dùng tra cứu, gợi ý và cá nhân hóa trải nghiệm nấu ăn nhờ việc kết hợp đồng bộ các kỹ thuật tiền xử lý dữ liệu, embedding, xử lý ngôn ngữ tự nhiên và mô hình ngôn ngữ lớn. Quy trình tiền xử lý dữ liệu giúp chuẩn hóa và làm sạch tập dữ liệu công thức nấu ăn, đảm bảo thông tin được tổ chức nhất quán, có cấu trúc và giàu ngữ nghĩa. Bên cạnh đó, việc ứng dụng kỹ thuật embedding cho phép chatbot hiểu sâu hơn về mối quan hệ giữa nguyên liệu, phương pháp chế biến và khẩu vị, từ đó thực hiện tìm kiếm ngữ nghĩa chính xác và gợi ý món ăn phù hợp với yêu cầu của người dùng.

Các kỹ thuật xử lý ngôn ngữ tự nhiên kết hợp với mô hình ngôn ngữ lớn giúp chatbot có khả năng diễn giải và phản hồi linh hoạt, tự nhiên theo phong cách hội thoại của con người nói chung và gen Z nói riêng. Nhờ đó, chatbot không chỉ trả lời câu hỏi một cách linh hoạt, dí dỏm mà còn có thể hỗ trợ đưa ra thực đơn, danh sách nguyên liệu cần thiết, tối ưu hóa thời gian nấu và cá nhân hóa theo nhu cầu của từng người dùng thay vì chỉ là những câu chữ khô khan truyền thống.

Tổng thể, hệ thống thể hiện tính khả thi, hiệu quả và tính mở rộng cao. Trong tương lai, mô hình có thể được nâng cấp bằng cách tích hợp thêm dữ liệu hình ảnh món ăn ở từng giai đoạn nấu nhằm giúp người dùng có thể cải thiện chất lượng món ăn, cải thiện khả năng nhận diện nguyên liệu, tối ưu phương pháp truy xuất thông tin và bổ sung chức năng đề xuất thực đơn dựa trên chế độ ăn hoặc tình trạng sức khỏe. Ngoài ra, mong muốn phát triển mô hình hỗ trợ đa ngôn ngữ và giọng nói, phát triển thành ứng dụng di động để tăng khả năng tiếp cận với người dùng. Việc phát triển theo hướng đa phương thức và học tăng cường sẽ giúp CookGPT trở thành một trợ lý nấu ăn thông minh, toàn diện và thân thiện hơn với người dùng.

# Tài liệu tham khảo

- [1] "FAISS: Facebook AI Similarity Search," Meta AI. Accessed: Nov. 30, 2025. [Online]. Available: <https://faiss.ai/>
- [2] "FAISS: Facebook AI Similarity Search Tutorial," DataCamp. Accessed: Nov. 30, 2025. [Online]. Available: <https://www.datacamp.com/blog/faiss-facebook-ai-similarity-search>
- [3] "HuggingFace Course – FAISS," HuggingFace. Accessed: Nov. 30, 2025. [Online]. Available: <https://huggingface.co/learn/llm-course/vi/chapter5/6>
- [4] "ChatGroq – LangChain Documentation," LangChain. Accessed: Nov. 30, 2025. [Online]. Available: <https://docs.langchain.com/oss/python/integrations/chat/groq>
- [5] "ChatGroq – LangChain Messages", LangChain. Accessed: Nov.30, 2025. [Online]. Available: <https://docs.langchain.com/oss/python/langchain/messages>
- [6] "Cơ chế hoạt động của Transformer?" Hugging Face — LLM Course. Accessed: Nov. 30, 2025. [Online]. Available: <https://huggingface.co/learn/llm-course/vi/chapter1/4>
- [7] "Tất tần tật về WordEmbedding – Word Embedding là gì?" Luu.name. Accessed: Nov. 30, 2025. [Online]. Available: <https://luu.name.vn/tat-tan-tat-ve-wordembedding-word-embedding-la-gi/>
- [8] "Từ Transformer Model tới Large Language Model" — Lưu.name. Accessed: Nov. 30, 2025. [Online]. Available: <https://luu.name.vn/tu-transformer-model-toi-large-language-model/>
- [9] "bge-m3" — BAAI. Hugging Face. Accessed: Nov. 30, 2025. [Online]. Available: <https://huggingface.co/BAAI/bge-m3>
- [n-gram] -Gram Language Modelling with NLTK<https://www.geeksforgeeks.org/nlp/n-gram-language-modelling-with-nltk/>

[Vtokenizer] itokenizer<https://viblo.asia/p/phan-loai-van-ban-tu-dong-bang-machine-learning-nhu-the-nao-4P856Pa1ZY3>