Trường Đại học Khoa học Tự nhiên, ĐHQGHN
Khoa Toán – Cơ – Tin học

Cooking Assistant Chatbot

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

Nội dung chính

Tổng quan về CookGPT

- Thông qua áp dụng các kiến thức thuộc lĩnh vực AI (đặc biệt trong mảng NLP), CookGPT ra đời với mục đích hỗ trợ người dùng Việt Nam trở thành một "master chef".
- Tập trung vào các lĩnh vực:
  - Sử dụng tiếng Việt, đảm bảo các hướng dẫn và gợi ý về món ăn phù hợp với người dùng.
  - 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.

Thu thập dữ liệu

- Thu thập >2000 công thức từ trang web monngonmoingay.com sử dụng công cụ Selenium WebDriver để Crawl dữ liệu.
- Dữ liệu thu thập được sẽ được sử dụng để làm dữ liệu phục vụ việc xây dựng mô hình.

Lý do sử dụng Selenium WebDriver

- Trang web dùng JavaScript → HTML tĩnh không đủ dữ liệu
- Các công cụ truyền thống như Requests hay BeautifulSoup chỉ đọc HTML tĩnh, nên sẽ bị mất thông tin, trong khi đó selenium webdriver cho phép các tướng tác sau
- Các tương tác phổ biến:
  - Đóng pop-up cookie
  - Cuộn trang để tải thêm dữ liệu (Lazy Loading)

Các bước thu thập dữ liệu

1. Khởi tạo trình duyệt
   - driver = webdriver.Chrome() chính là lệnh khởi tạo Selenium WebDriver.
   - Sau bước khởi tạo này bạn có thể dùng driver để:
   driver.get(url) → mở trang web
   driver.find_element(...) → tìm nút, ảnh, nội dung
   driver.click() → bấm nút
   driver.quit() → đóng trình duyệt

Các bước thu thập dữ liệu

1. Crawl URL và hình ảnh
   Bước 1: Duyệt qua từng trang
   Bước 2: Xử lý popup (chỉ ở trang đầu tiên)
   Bước 3: Tìm kiếm các thẻ chứa món ăn
   Bước 4: Trích xuất thông tin từ mỗi thẻ
   Bước 5: Lưu trữ tạm thời

Các bước thu thập dữ liệu

1. Crawl chi tiết món ăn
   Duyệt qua URL đã thu thập và mở trang bằng driver.get(recipe["url"])
   Thu thập các thông tin như: tên món ăn, nguyên liệu, các bước nấu, thời gian, độ khó, tip...

Tiền xử lý dữ liệu

- Chuẩn hoá đơn vị khối lượng
- Chuẩn hoá thời gian nấu
- Chuẩn hoá tên nguyên liệu
- Sửa lỗi chính tả nguyên liệu
- Phân loại món ăn theo tên

Chuẩn hoá đơn vị (normalize_unit)

- Xử lý list bằng đệ quy
- Dùng regex thay thế đơn vị ngay sau số:
  - g $\rightarrow$ gam
  - M $\rightarrow$ muỗng
  - tr $\rightarrow$ trái
  - c $\rightarrow$ củ

Chuẩn hoá thời gian nấu

- Trích số giờ và phút bằng regex
- Quy đổi toàn bộ về phút:

$$\text{tổng phút} = 60 \times \text{giờ} + \text{phút}$$

Chuẩn hoá tên nguyên liệu

- Loại bỏ icon, đưa về chữ thường
- Xoá ký tự đặc biệt
- Loại bỏ từ mô tả trạng thái: “cắt”, “băm”, “tươi”, . . .
- Loại bỏ tên thương hiệu: Ajingon, Ajinomoto, . . .
- Gom nhóm nhiều biến thể về cùng một tên chuẩn

Sửa lỗi chính tả nguyên liệu

- Dùng bảng từ điển `INGREDIENT_CORRECTIONS`
- Tách tên thành từng từ
- Thay thế từng từ nếu tồn tại trong từ điển
- Giữ nguyên nếu không có ánh xạ
- Giảm trùng lặp do lỗi gõ, viết tắt, sai chính tả

Phân loại món ăn tự động

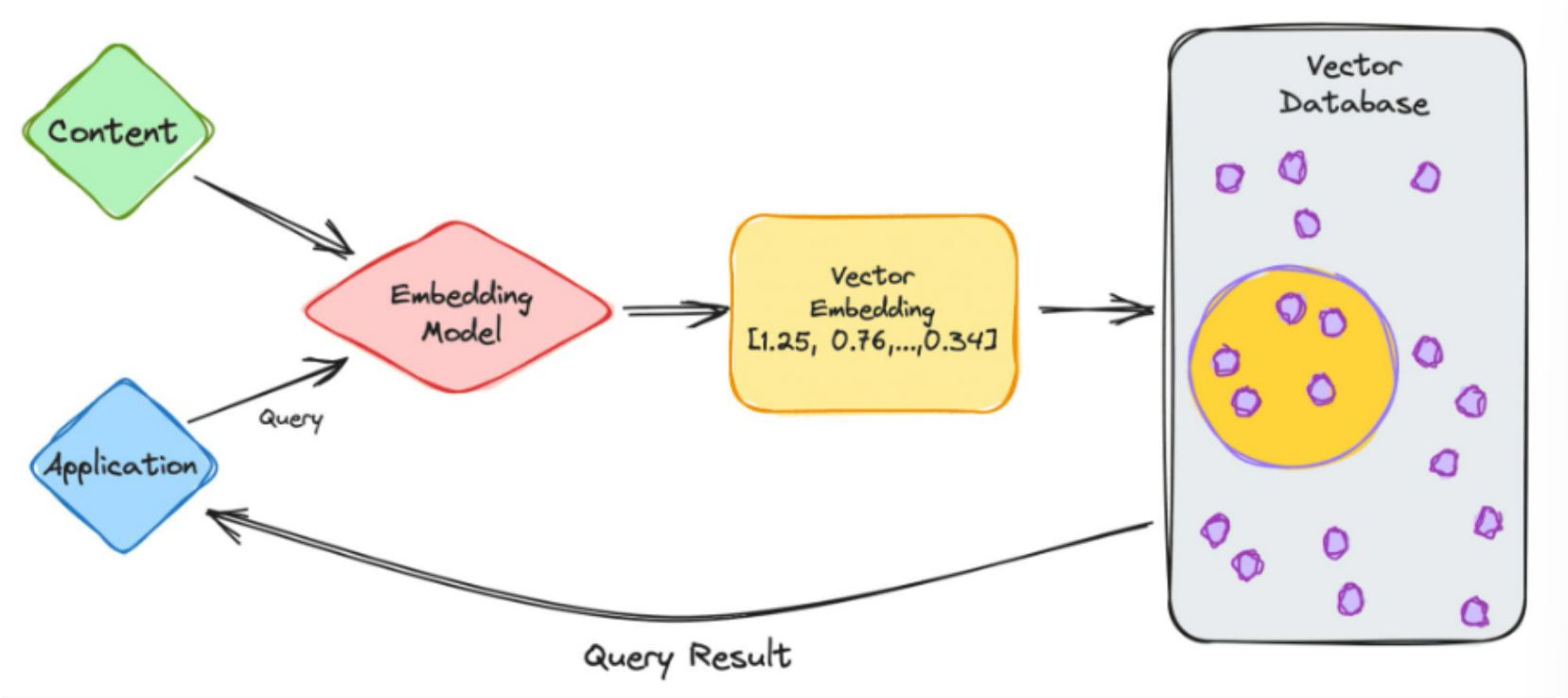
- So sánh tên món với tập từ khoá
- Ví dụ:
  - "bún", "phở" → Món nước
  - "chiên", "rán" → Chiên
- Trả về tên thể loại viết hoa chữ cái đầu

Giới thiệu chung về embedding

- Embedding là một kỹ thuật trong AI, học máy và xử lý ngôn ngữ tự nhiên dùng để biểu diễn dữ liệu dạng vector trong không gian nhiều chiều.
- Embedding thể hiện mối quan hệ và mức độ tương đồng giữa các đối tượng thông qua khoảng cách hoặc góc trong không gian vector.
- Hỗ trợ máy tính nhận diện, phân loại, tìm kiếm thông tin hoặc dự đoán hiệu quả.

Nguyên lý hoạt động của embedding

- Thông qua một mô hình học sâu được huấn luyện, dữ liệu được chuyển thành vector số trong một không gian liên tục.



Các loại embedding phổ biến

- **Word embedding**: biểu diễn các từ thành vector số: Word2Vec, GloVe. Nhận diện từ đồng nghĩa, phân loại từ theo ngữ cảnh.
- **Sentence embedding** và **document embedding**: biểu diễn câu, tài liệu thành vector. Cho phép các mô hình hiểu nghĩa của cả câu, đoạn văn hoặc tài liệu dài: Sentence-BERT, BGE-M3.
- **Image embedding**: chuyển đổi hình ảnh thành vector số. Giúp máy tính nhận diện hình ảnh, tìm kiếm hình ảnh tương tự, phân loại hình ảnh theo nội dung.
- **Audio embedding**: biểu diễn âm thanh, giọng nói, nhạc thành vector số, hỗ trợ các ứng dụng nhận diện giọng nói, phân loại nhạc, phát hiện sự kiện âm thanh.

Mô hình embedding BGE-M3

- Là mô hình sentence embedding đa chức năng được xây dựng trên kiến trúc Transformer, đặc điểm:
  - Đa ngôn ngữ: Hỗ trợ hơn 100 ngôn ngữ và ross-lingual.
  - Đa chức năng: thực hiện đồng thời 3 loại truy xuất: tìm kiếm vector dày đặc, tìm kiếm vector thưa, tìm kiếm đa vector.
  - Xử lý văn bản có độ dài khác nhau lên đến 8192 token.
  - Huấn luyện dựa trên phương pháp self-knowledge distillation.

=> Đáp ứng các yêu cầu của project: hỗ trợ tiếng Việt , cho kết quả tương đồng ngữ nghĩa chính xác, embedding các câu ngắn: tên món ăn, tên nguyên liệu.

Quy trình xử lý văn bản trong BGE-M3

BGE-M3 sử dụng kiến trúc Transformer. Cụ thể thực hiện các bước:

- **Tokenization:** chia văn bản thành các token và mã hóa thành vector.
- **Token Embedding:** mỗi token được biểu diễn dưới dạng vector số.
- **Self-Attention:** xem xét mối quan hệ và mức độ phụ thuộc giữa các token trong câu.
- **Transformer Encoder:** biến đổi các vector nhúng thành biểu diễn ngữ cảnh của toàn câu.

Triển

- Đọc dữ liệu từ `recipes_cleaned.csv`: `dish_name`, `ingredient_names`
- Load mô hình BGE-M3 từ `config.yml`
- Encode tên món ăn thành một vector 1024 chiều
- Encode nguyên liệu từng item riêng biệt thành mảng vector 1024 chiều
- Lưu kết quả vào file `recipes_embeddings.pkl`

Mục tiêu xác định intent

Xác định intent nhằm phân loại chính xác ý định của người dùng khi tương tác với hệ thống:

- Nhận biết người dùng muốn **tìm kiếm món ăn**.
- Nhận biết người dùng muốn **hướng dẫn nấu ăn**.

Việc nhận diện intent chính xác tạo nền tảng cho các bước tiếp theo như trích xuất thông tin và gợi ý món ăn.

Pipeline xác định intent

Hình: Pipeline xác định intent

Thu thập và xây dựng tập dữ liệu Intent

Hai nhóm intent chính:

- `suggest_dishes` (gợi ý món ăn)
- `cooking_guide` (hướng dẫn nấu ăn)

Quy trình xây dựng dữ liệu:

1. Mô tả intent ban đầu.
2. Sinh câu mẫu bằng mô hình ngôn ngữ (đa dạng, paraphrase, dễ gây nhầm lẫn).
3. Kiểm duyệt thủ công (loại bỏ lỗi, trùng lặp, nhiễu).
4. Mở rộng dữ liệu trong quá trình kiểm thử.

Kết quả: khoảng **1.000 mẫu intent** đa dạng.

Phương pháp Hybrid

Hệ thống kết hợp **embedding ngữ nghĩa** và **rule-based từ khóa**:

1. Biểu diễn ngữ nghĩa bằng embedding (BGE-M3).
2. Rule-based từ từ khóa đặc trưng cho từng intent.
3. Công thức kết hợp điểm:

$$S = s_{\text{embed}} + \alpha \cdot s_{\text{rule}}, \quad \alpha = 0.1$$

Cách triển khai

Các bước triển khai:

1. Tiếp nhận câu truy vấn.
2. Sinh embedding bằng BGE-M3.
3. Tính toán top-k mean cosine.
4. Áp dụng rule-based từ khóa.
5. Tổng hợp điểm và chọn intent.

Ưu điểm

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

Mục tiêu Slot Extraction

- Trích xuất thông tin quan trọng từ câu nhập của người dùng.
- Các slot chính:
  - **ingredient**: nguyên liệu (gà, khoai tây, cà rốt)
  - **category**: danh mục món ăn (kho, xào, luộc)
  - **dish_name**: tên món cụ thể (bánh mì chiên bơ tỏi)
  - **time**: thời gian nấu (45 phút, 2 giờ)
  - **difficulty**: độ khó (dễ, trung bình, khó)
  - **serving**: số khẩu phần (4 người, 2 phần)
- Mục tiêu: nhận diện chính xác, tránh nhầm substring, để làm input phục vụ quá trình tìm kiếm sau

Phương pháp: Tokenization

- Chia nhỏ văn bản thành token:
  - Word-level: "Tôi muốn ăn cơm" → [Tôi, muốn, ăn, cơm]
  - Character-level: "cơm" → [c, ơ, m]
  - Subword/BPE: "cooking" → [cook, ing]
  - Sentence-level: "Tôi thích NLP. Nó thú vị." → [Tôi thích NLP., Nó thú vị.]
- Mục đích: chuẩn hóa dữ liệu, phục vụ embeddings/NLP.

Thư viện Tokenization

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
BERT/HuggingFace	Python	Subword tokenization (BPE/WordPiece), tạo token ID cho embeddings

Lý do chọn ViTokenizer

- **spaCy**: mạnh mẽ đa ngôn ngữ nhưng không tối ưu cho tiếng Việt.
- **ViTokenizer**: chuyên biệt cho tiếng Việt, xử lý từ ghép, phù hợp nhận diện slot.
- **BERT Tokenizers**: chuẩn cho embeddings, dùng kết hợp sau khi tokenize.

Phương pháp: N-grams

- Tạo ra tập con liên tiếp của token (1 đến N).
- Ưu tiên n-grams dài để tránh nhầm lẫn.
- Kết hợp từ điển để phát hiện slot chính xác.

Phương pháp: Fuzzy Matching

- Là phương pháp so khớp gần đúng với từ điển, không yêu cầu trùng tuyệt đối.
- Xử lý biến thể ngôn ngữ, lỗi chính tả.
- Ví dụ: "khoai tay" $\rightarrow$ "khoai tây".
- Dựa trên similarity ratio, chỉ match khi vượt ngưỡng.

Phương pháp: Rule-based Extraction

- Dùng regex/từ khóa đặc trưng cho slot có cấu trúc chuẩn.
- Ví dụ:
  - "2 giờ 30 phút" $\rightarrow$ 150 phút.
  - "4 người" $\rightarrow$ số khẩu phần 4.
- Kết hợp tokenization + n-grams để tăng độ chính xác.

Cách triển khai

- Tokenization bằng ViTokenizer.
- Sinh n-grams (1–6 token).
- So khớp nguyên liệu với từ điển + fuzzy matching.
- Regex cho category, thời gian, khẩu phần, độ khó.
- Gom slot theo intent:
  - `suggest_dishes`: category, ingredient, time, difficulty, serving.
  - `cooking_guide`: dish_name.

Ưu điểm và Nhược điểm

Ưu điểm:

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

Nhược điểm:

- Category/nguyên liệu dựa vào từ điển → không nhận diện từ mới.
- Cần kết hợp embedding + cosine similarity để xử lý tự nhiên hơn.
- Cần rule ranking hoặc confidence score để ưu tiên slot đáng tin cậy.

FAISS – Facebook AI Similarity Search

- Thư viện mã nguồn mở cho **vector similarity search**.
- Hỗ trợ **Exact / Approximate Nearest Neighbor**, GPU acceleration.
- Ứng dụng: **Semantic Search**, **Recommendation**, **RAG chatbot**.

Nhu cầu sử dụng FAISS

- **Vấn đề với từ khóa truyền thống:**
  - BM25, TF-IDF không hiểu **ngữ nghĩa**.
  - Khó tìm món ăn tương tự khi query khác từ.
- **Giải pháp: Embedding + FAISS**
  - Chuyển text sang vector dense.
  - Tìm nearest neighbors nhanh.

Kiến trúc FAISS

- **Index**: lưu trữ và truy xuất vector
- **Metric**: L2 distance / Inner Product
- **Quantizer**: cho approximate search
- **IVF**: chia không gian vector
- **GPU module**: tăng tốc dataset lớn

Các loại Index trong FAISS

Exact Search

- IndexFlatIP, IndexFlatL2
- Ưu điểm: Độ chính xác 100%
- Nhược điểm: Tốn CPU, dataset lớn

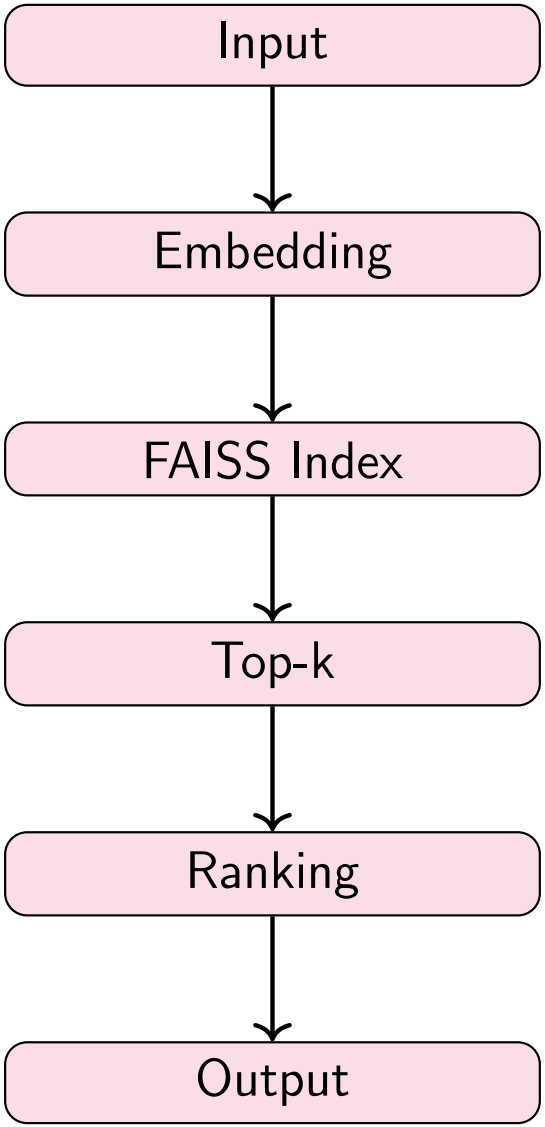
Approximate Search

- IndexIVFFlat, IndexIVFPQ, HNSW
- Ưu điểm: Nhanh, tiết kiệm bộ nhớ
- Nhược điểm: Recall thấp hơn exact

GPU Index

- GPUFlat, GPUIVF
- Ưu điểm: Tăng tốc truy vấn >< Nhược điểm: Cần GPU

Cơ chế hoạt động



Triển khai FAISS trong hệ thống

- Chọn **IndexFlatIP** cho dataset vừa phải.
- Thêm vector embedding của ingredients và dish names.
- Top-k retrieval: $\text{top\_k} = \underset{i \in [1,N]}{\arg\max}(\text{q} \cdot \text{v}_i)$
- Mapping top-k ID $\rightarrow$ dataset gốc.

Ưu điểm & Hạn chế của FAISS

Ưu điểm

- Tốc độ cao, hỗ trợ Exact/Approx
- Hỗ trợ GPU
- Dễ tích hợp với text/hình ảnh/audio
- Batch search tối ưu throughput

Hạn chế

- Không tối ưu với dữ liệu sparse
- Approximate index recall thấp hơn exact
- Index lớn tốn RAM nếu không dùng IVF/PQ/GPU

Ứng dụng thực tế của FAISS

- Chatbot / Search Engine (RAG)
- Recommendation System
- Image Similarity (CLIP/CNN)
- Semantic Search NLP
- Multimodal Search (text, hình ảnh, audio)

Hiệu suất tìm kiếm với FAISS

- **Index dish:** 2420 vectors, 1024 chiều
  - Độ trễ trung bình: 1.10 ms/query
  - Truy vấn mỗi giây(QPS): 910
  - Precision@5 / Recall@5: 100%
- **Index ingredient names:** 15752 vectors, 1024 chiều
  - Độ trễ trung bình: 3.85 ms/query
  - Truy vấn mỗi giây(QPS): 260
  - Precision@5 / Recall@5: 24%

Nhận xét:

- Index dish rất nhanh và chính xác $\rightarrow$ phù hợp với truy vấn món ăn.
- Index ingredient names chậm hơn, độ chính xác thấp hơn $\rightarrow$ cần tối ưu hoặc kết hợp với các bộ lọc khác.

Lựa chọn mô hình LLM

Mô hình	**Chi phí API**	**Phần cứng**
Llama 3.1 8B Instant	Miễn phí	Không yêu cầu
Local Llama	Miễn phí	Cấu hình mạnh
GPT-4, GPT-3.5	Mất phí	Không yêu cầu

- Sử dụng Llama 3.1 8B Instant để sinh phản hồi tự động từ dữ liệu đã xử lý ở bước NLP.
- Triển khai qua LangChain, quản lý pipeline, prompt và tích hợp dữ liệu từ cơ sở vector.
- Gọi mô hình qua Groq API, tận dụng phần cứng tăng tốc để giảm độ trễ và cải thiện tốc độ phản hồi.

Triển khai mô hình LLM

- Đầu vào: dữ liệu đã xử lý và chuẩn hóa từ bước NLP (nguyên liệu, tên món ăn và yêu cầu liên quan).
- Nếu không có kết quả phù hợp, hệ thống sẽ trả về thông báo hài hước để tăng tính tương tác.
- Nếu có kết quả, hệ thống tạo các thông điệp đầu vào cho LLM.

Triển khai mô hình LLM

- Sử dụng SystemMessage và HumanMessage để chuẩn bị tin nhắn:
  - SystemMessage: chuyên gia ẩm thực gen Z Việt Nam; giọng điệu dí dỏm, hài hước; nội dung ngắn gọn, rõ ràng và chính xác.
  - HumanMessage: sinh phản hồi cho người dùng theo những yêu cầu và quy tắc trong SystemMessage.
- Khi lỗi API xảy ra, hệ thống trả về dữ liệu NLP mà không sinh phản hồi tự động.
- Các hàm chuyên biệt describe_dishes() và smooth_instructions() giúp hệ thống đáp ứng nhiều truy vấn khác nhau.
- Việc giữ nguyên danh sách món ăn và không bịa món mới giúp hạn chế hiện tượng ảo giác AI và tăng độ tin cậy.