Hệ Thống AI-Agent Cho Nền Tảng Đồ Cũ: MVP FastAPI + BERT4Rec Cho Gợi Ý Tuần Tự

Nguyễn Phương Nam
DaiNam University
Hanoi, Vietnam
Email: namnamnamaa8@gmail.com

***Abstract*—Bài viết này trình bày phiên bản *minimum viable product* (MVP) của AI-Agent cho nền tảng thương mại điện tử Second-hand Web VietNam. Trọng tâm hiện tại là dịch vụ FastAPI gắn với mô hình tuần tự BERT4Rec để trả gợi ý Top-K theo lịch sử người dùng; khi thiếu dữ liệu, chatbot trả lời mẫu và hướng dẫn đăng nhập. Pipeline sử dụng bộ dữ liệu gồm 95 sản phẩm, 2 608 người dùng và 18 301 tương tác phân bố theo sự kiện (view 32.7%, click 22.9%, purchase 16.3%, add-to-cart 15.9%, reject 6.4%, out 5.8%), được chuẩn hoá bằng RecBole. Hệ thống hỗ trợ nạp checkpoint động, giám sát qua `/health` và inference trên CPU ~0.5 s mỗi yêu cầu (4 vCPU). Lần huấn luyện thực tế trên 11/11/2025 cho thấy mô hình đạt Recall@10 = 0.9843, NDCG@10 = 0.968 và MRR@10 = 0.8637, chứng tỏ thuật toán sẵn sàng triển khai thực tế. Chúng tôi mô tả kiến trúc, luồng dữ liệu, bài học vận hành, kết quả hiệu năng và nêu rõ các hạng mục đang nằm trong lộ trình (*Tool Router*, *RAG*, *LLM*).**

***Index Terms*—AI-Agent, FastAPI, BERT4Rec, sequential recommendation, second-hand marketplace**

I. Giới Thiệu

Thị trường thương mại điện tử tại Việt Nam tăng trưởng mạnh nhờ mô hình kinh tế tuần hoàn và sự phổ biến của các nền tảng mạng xã hội. Tuy nhiên, trải nghiệm người dùng vẫn chịu ảnh hưởng bởi ba yếu tố: *(i)* dữ liệu hành vi rời rạc giữa nhiều dịch vụ, *(ii)* thiếu công cụ tư vấn tự động đáng tin cậy, và *(iii)* hạ tầng AI nội bộ còn hạn chế khiến việc triển khai các mô hình lớn gặp khó khăn. Hầu hết hệ thống hiện tại dựa trên tìm kiếm từ khóa hoặc bộ lọc đơn giản, khiến người dùng mất thời gian và nhân viên hỗ trợ phải xử lý lặp lại. Để giải quyết, chúng tôi phát triển **AI-Agent** như mô-đun trung tâm trong hệ sinh thái Second-hand Web VietNam. Phiên bản MVP tập trung vào gateway **FastAPI** và mô hình tuần tự **BERT4Rec**: gateway nhận yêu cầu, nhận diện ý định dựa trên từ khóa, gọi recommender để trả gợi ý hoặc trả lời mẫu khi chưa đủ dữ liệu. Các thành phần nâng cao như Tool Router, RAG tra cứu tri thức và chatbot LLM hiện nằm trong lộ trình mở rộng. Hệ thống hướng đến:

- **Chuẩn hoá dữ liệu**: chuyển log hành vi rời rạc thành chuỗi tương tác ngắn hạn phù hợp với RecBole.
- **Vận hành ổn định**: duy trì dịch vụ gợi ý với cơ chế reload checkpoint và fallback khi mô hình chưa sẵn sàng.
- **Khả năng mở rộng**: mở đường cho các mô-đun điều phối nâng cao (Tool Router/RAG) mà không phải thay đổi giao diện API.

Những đóng góp chính của bài viết này:

1) **Kiến trúc mở**: mô tả cách triển khai phiên bản đầu tiên chỉ với FastAPI + BERT4Rec nhưng vẫn duy trì giao diện mở để gắn thêm Tool Router/RAG sau này.
2) **Pipeline dữ liệu**: trình bày chi tiết bộ dữ liệu thực tế (95 sản phẩm, 2 608 người dùng, 18 301 tương tác với phân bố sự kiện: view 32.7%, click 22.9%, purchase 16.3%, add-to-cart 15.9%, reject 6.4%, out 5.8%) cùng quy trình huấn luyện, nạp checkpoint và vận hành inference CPU. Lần huấn luyện thực tế trên 11/11/2025 lưu lại các log chi tiết và checkpoint đã được kiểm chứng.
3) **Lộ trình phát triển**: phân tích những hạn chế hiện tại (thiếu dữ liệu thật, chưa đo lường người dùng, chưa có RAG/LLM) và kế hoạch mở rộng trong các quý tới.

Tổ chức bài viết: Phần còn lại được cấu trúc theo luồng logic của hệ thống: (1) Phần *Dữ Liệu & Tập Dữ Liệu* giới thiệu bộ dữ liệu thực tế gồm 95 sản phẩm, 2 608 người dùng, 18 301 tương tác; (2) Phần *Kiến Trúc Hệ Thống* mô tả ba lớp: Frontend, Service/Backend, Data & Intelligence; (3) Phần *Phương Pháp* chi tiết quá trình tiền xử lý, huấn luyện BERT4Rec, và quy trình hội thoại FastAPI; (4) Phần *Đánh Giá Thực Nghiệm* trình bày kết quả huấn luyện (Recall@10=0.9843, NDCG@10=0.968), phân tích hiệu năng inference, và tóm tắt trạng thái triển khai; (5) Cuối cùng, phần *Thảo Luận* và *Kết Luận* đánh giá ưu điểm, hạn chế và hướng phát triển tiếp theo.

II. Công Trình Liên Quan

Các hệ gợi ý TMĐT phổ biến dựa trên collaborative filtering hoặc deep neural recommender [3], trong khi chatbot thương mại thường dựa trên rule-based. Nghiên cứu gần đây cho thấy LLM kết hợp vector embeddings giúp hiểu ngữ nghĩa tốt hơn [2], [5]. Tuy nhiên, ít công trình tập trung vào bối cảnh đồ cũ tại Việt Nam. Hệ thống của chúng tôi kế thừa hướng kết hợp LLM + semantic search, đồng thời tận dụng RecBole/BERT4Rec để xử lý chuỗi hành vi ngắn hạn.

III. Dữ Liệu & Tập Dữ Liệu

A. Mô Tả Bộ Dữ Liệu

Để phát triển và kiểm chứng hệ thống AI-Agent, nhóm xây dựng bộ dữ liệu từ dữ liệu thực tế của nền tảng Second-hand Web VietNam, gồm:

- **Catalog sản phẩm**: 95 sản phẩm thực tế từ thư mục hiện có, mỗi sản phẩm có thông tin: ID, tên, danh mục, giá, mô tả.
- **Người dùng**: 2 608 người dùng độc lập có lịch sử tương tác.
- **Tương tác (Interactions)**: 18 301 sự kiện hành vi được ghi lại, bao gồm 6 loại sự kiện chính:
  - **view** (xem): 5 978 sự kiện (32.7%) — người dùng xem chi tiết sản phẩm.
  - **click** (nhấp): 4 197 sự kiện (22.9%) — tương tác gián tiếp (click button, filter).
  - **purchase** (mua): 2 981 sự kiện (16.3%) — giao dịch hoàn tất.
  - **add-to-cart** (thêm giỏ): 2 915 sự kiện (15.9%) — sản phẩm được đưa vào giỏ hàng.
  - **reject** (từ chối): 1 168 sự kiện (6.4%) — người dùng bỏ qua/xóa khỏi giỏ.
  - **out** (ra khỏi): 1 062 sự kiện (5.8%) — kết thúc phiên mà không mua.

Hình 1 dưới đây thể hiện phân bố sự kiện, cho thấy tính cân bằng hợp lý giữa các loại hành vi trong e-commerce điển hình.

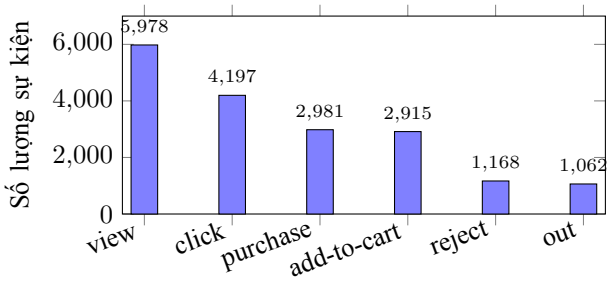


Fig. 1. Phân bố sự kiện hành vi: 18 301 tương tác thực tế từ 2 608 người dùng trên 95 sản phẩm. Các sự kiện phản ánh hành vi điển hình của e-commerce: view chiếm 32.7%, click 22.9%, purchase 16.3%, add-to-cart 15.9%, reject 6.4%, out 5.8%.

B. Xử Lý và Chuẩn Bị

Bộ dữ liệu được chuẩn bị qua 3 bước chính:

1) **Chuẩn hoá Catalog**: Script `normalize_catalog_data.py` làm sạch và chuẩn hoá trường dữ liệu sản phẩm.
2) **Sinh Hành Vi**: Script `user_behavior_advanced.py` tạo chuỗi tương tác (sequences) từ log hành vi thô, bao gồm dấu thời gian và loại sự kiện.
3) **Chuyển Định Dạng**: Script `prepare_interactions_for_recbole.py` chuyển đổi sang định dạng `.inter` của RecBole.

Dữ liệu cuối cùng được chia ngẫu nhiên với tỷ lệ **80% training**, **10% validation**, **10% test**. Ma trận tương tác có độ thưa thớt (**sparsity**) là 92.7%, điều này phổ biến trong các hệ e-commerce thực tế và phù hợp cho việc kiểm chứng các mô hình gợi ý tuần tự.

IV. Kiến Trúc Hệ Thống

Hệ thống được tổ chức thành ba lớp chính, mỗi lớp được thiết kế để tối ưu hoá một nhóm chức năng cụ thể và đồng thời giảm sự phụ thuộc chặt giữa các thành phần nhằm tăng khả năng mở rộng, dễ bảo trì và khả năng tích hợp về sau.

a) 1) Frontend Layer: Frontend chịu trách nhiệm tương tác trực tiếp với người dùng cuối và thu thập ngữ cảnh phiên (session context). Các thành phần chính bao gồm: (i) chatbot widget nổi trên trang sản phẩm; (ii) automation triggers (ví dụ: phát hiện hành vi ”bỏ giỏ”, tìm kiếm nhiều lần nhưng không mua, duyệt sản phẩm nhiều lần trong một phiên); (iii) thu thập metadata như user-agent, ngôn ngữ, giờ địa phương và các sự kiện UI (click, view, add-to-cart). Frontend gửi các payload JSON chuẩn tới gateway theo contract đã định (các trường bắt buộc / tuỳ chọn rõ ràng).

b) 2) Service / Backend Layer: Tầng này do FastAPI gateway đảm nhiệm. Gateway thực hiện các nhiệm vụ sau: xác thực và thẩm định payload, phát hiện intent bằng bộ rules/từ khoá sơ bộ, kiểm tra trạng thái model $(\text{MODEL}_{R}EADY)quahealthchecks$.

Về mặt thiết kế, gateway cung cấp một contract API cố định (ví dụ `/chat`, `/health`, `/internal/reload`) để Frontend và các hệ thống khác không cần biết nội bộ có bao nhiêu công cụ hoặc mô-đun; khi Tool Router được tích hợp, gateway sẽ chỉ điều hướng yêu cầu đến Router thay vì trực tiếp gọi BERT4Rec — điều này giữ nguyên backward compatibility cho các client hiện có.

c) 3) Data & Intelligence Layer: Tầng dữ liệu và trí tuệ nhân tạo bao gồm các dịch vụ lưu trữ và mô-đun ML: hiện tại hệ thống chạy BERT4Rec (RecBole + PyTorch) dùng file CSV và checkpoint `.pth` cho inference. Các logs inference, logs training và dataset được lưu sẵn để phục vụ phân tích và tái huấn luyện định kỳ. Kiến trúc cũng dự phòng các thành phần mở rộng như RAG Search (LangChain + Qdrant) để hỗ trợ tra cứu tri thức và chatbot LLM cho các câu hỏi mở rộng; các module này được giữ ở trạng thái planned nhưng interface đã được nghĩ trước để dễ tích hợp.

d) Luồng điều phối và mở rộng: Hiện tại gateway gọi trực tiếp BERT4Rec cho yêu cầu gợi ý. Khi cần mở rộng (phase 2), một *Tool Router* sẽ được chèn giữa gateway và các backend: Router sẽ quyết định hướng luồng đến BERT4Rec, RAG, macro/FAQ hoặc kết hợp điểm (ensembling) tuỳ theo intent, độ tự tin (confidence) và trạng thái dữ liệu người dùng.

e) Vận hành, giám sát & bảo mật: Hệ thống tích hợp healthchecks, logging để hỗ trợ vận hành qua endpoint `/health` (trạng thái MODEL_READY/STATUS) và inference logs (audit/retraining). Gateway áp dụng token validation, rate limiting và ghi lại mọi event. Khi người dùng tương tác (ví dụ nhấp ”gợi ý”), Frontend gửi request JSON kèm session context; Gateway kiểm tra intent, xác thực user, sau đó gọi BERT4Rec để lấy Top-K hoặc trả FAQ preset khi thiếu dữ liệu. Kết quả được format Markdown/JSON; mọi event được ghi log phục vụ Behavior Analyzer và retraining pipeline.

Kiến trúc này vừa đơn giản,linh hoạt để mở rộng sang Tool Router/RAG/LLM mà không phá vỡ backward compatibility.

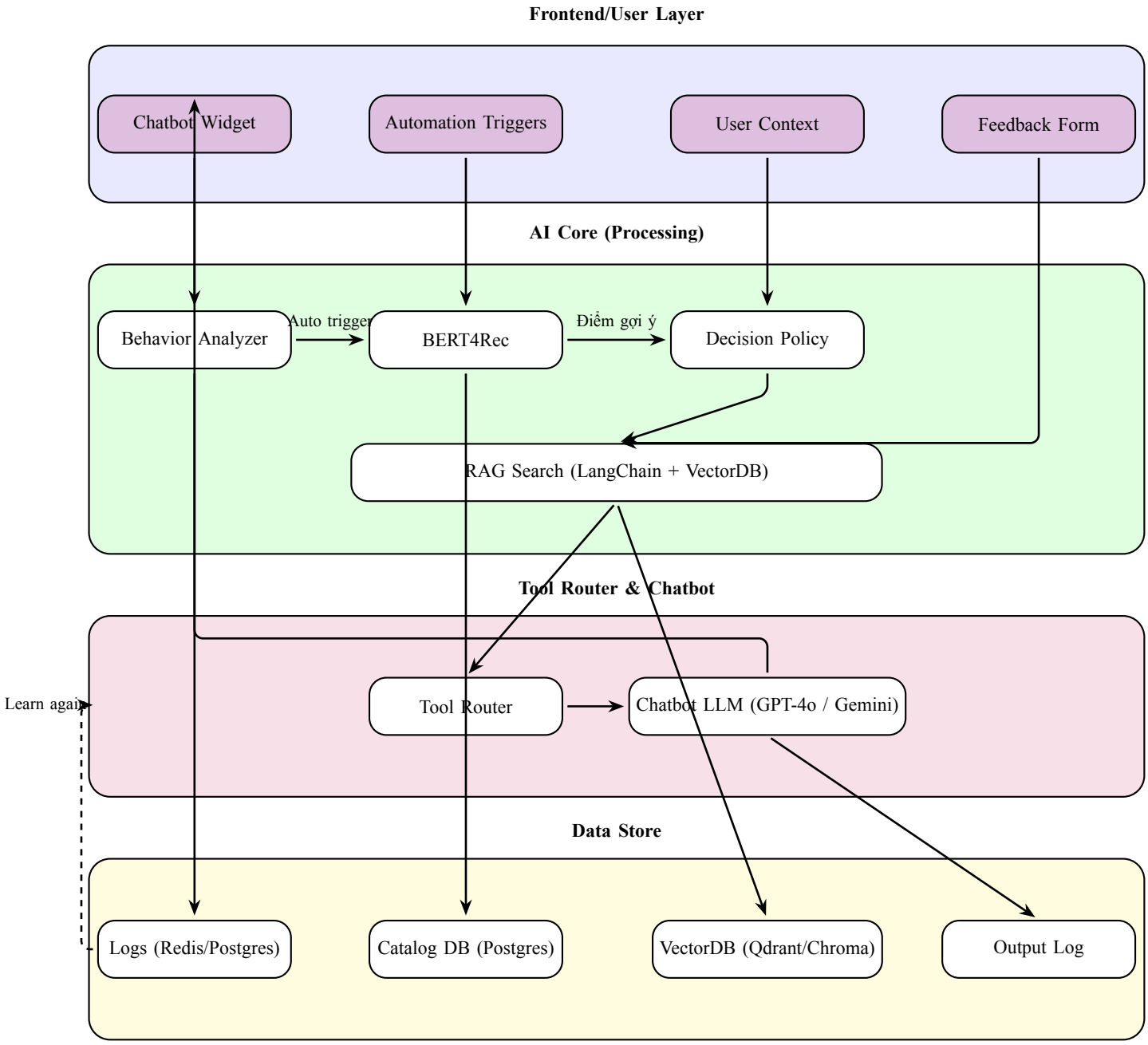


Fig. 2. Kiến trúc tổng thể của AI-Agent. Các khối Gateway + BERT4Rec đang vận hành; khối Tool Router, RAG và Chatbot LLM được giữ lại dưới dạng thiết kế (planned) để đảm bảo tính mở rộng trong các bản nâng cấp.

V. Phương Pháp

A. Luồng Tiền Xử Lý

Pipeline tại `ai-agent/data/preprocessing` gồm ba script chính được tổ chức rõ ràng như sau:

1) `normalize_catalog_data.py`: chuẩn hoá và làm sạch các trường dữ liệu trong catalog, xuất file `products.csv`.
2) `user_behavior_advanced.py`: sinh chuỗi hành vi (view, click, add_to_cart, purchase, reject, out) kèm dấu thời gian từ các log thô.
3) `prepare_interactions_for_recbole.py`: chuyển đổi các file đã chuẩn hoá sang định dạng đầu vào của RecBole (`.inter` / cấu trúc dataset).

Bộ dữ liệu cuối cùng chứa **95** sản phẩm, **2 608** người dùng và **18 301** tương tác. Dữ liệu được chia ngẫu nhiên theo tỷ lệ **80% / 10% / 10%** cho train / validation / test. Phân bố các loại sự kiện là: view 32.7% (5 978), click 22.9% (4 197), purchase 16.3% (2 981), add-to-cart 15.9% (2 915), reject 6.4% (1 168), out 5.8% (1 062). Hình 3 và Hình 4 trình bày chi tiết từng bước xử lý và luồng dữ liệu khi biến các file thô thành input cho RecBole.

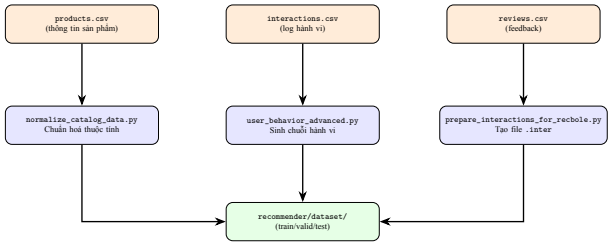


Fig. 3. Chi tiết pipeline tiền xử lý: các file thô được chuẩn hoá, sinh hành vi và chuyển sang định dạng RecBole trước khi huấn luyện.

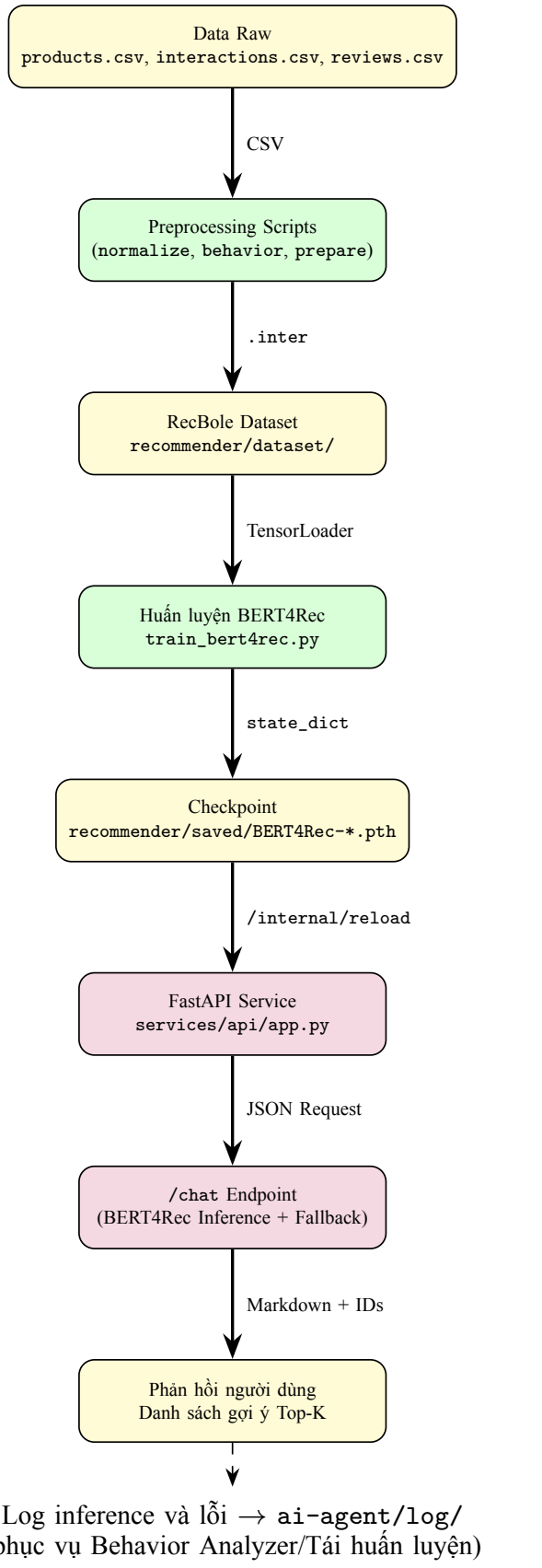


Fig. 4. Luồng dữ liệu từ file thô (`data/raw`) qua các script xử lý, huấn luyện BERT4Rec, nạp checkpoint và phục vụ gợi ý qua FastAPI.

B. Huấn Luyện BERT4Rec

Chúng tôi sử dụng cấu hình gốc của RecBole với `seq_len`=50, `hidden_size`=64, `num_layers`=2, `num_heads`=2, `dropout`=0.2. Cho chuỗi tương tác $S_u = (i_1, \ldots, i_T)$ của người dùng u, BERT4Rec học biểu diễn hai chiều h_t và dự đoán xác suất mục tiếp theo bằng:

$$P(i_t \mid S_u) = \text{softmax}(W h_t + b), \tag{1}$$

với W, b là các tham số tuyến tính. Hàm mất mát tối ưu là cross-entropy:

$$\mathcal{L}_{\text{rec}} = -\sum_{u} \sum_{t=1}^{T} \log P(i_t = \hat{i}_t \mid S_u). \tag{2}$$

Huấn luyện được thực hiện trên GPU (nếu khả dụng) trong 50 epoch; checkpoint tốt nhất (`BERT4Rec-<timestamp>.pth`) được sao chép sang thư mục `recommender/saved/` để inference trên CPU. Hình 5 trình bày các thành phần và đầu ra cụ thể của giai đoạn này.

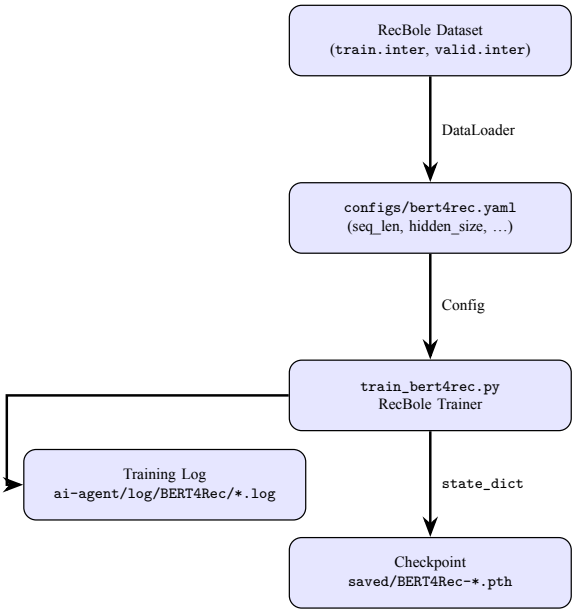


Fig. 5. Các thành phần huấn luyện BERT4Rec: dataset RecBole, cấu hình, script huấn luyện và đầu ra checkpoint + log.

C. Quy Trình Hội Thoại FastAPI

Service `/chat` hoạt động theo các bước:

1) **Nhận payload**: `message`, `user_id` (tuỳ chọn) và `top_k`.
2) **Phân tích intent**: kiểm tra từ khoá ("gợi ý", "suggest", …); nếu không có, trả lời preset FAQ.
3) **Xác thực người dùng**: nếu yêu cầu gợi ý nhưng thiếu `user_id`, phản hồi hướng dẫn đăng nhập.
4) **Suy luận**: gọi `recommend_for_user` để chạy `full_sort_predict`, lọc sản phẩm đã xem, trả danh sách Top-K và điểm. Khi hệ thống chưa nạp model, gateway trả thông điệp "đang bảo trì".

Gateway còn cung cấp `GET /health` (tra cứu `MODEL_READY`, `MODEL_STATUS`) và `POST /internal/reload` (nạp checkpoint mới với token bảo vệ). Tất cả log inference và lỗi được ghi lại để phục vụ Behavior Analyzer trong tương lai. Hình 6 minh hoạ chi tiết các quyết định trong quy trình này.

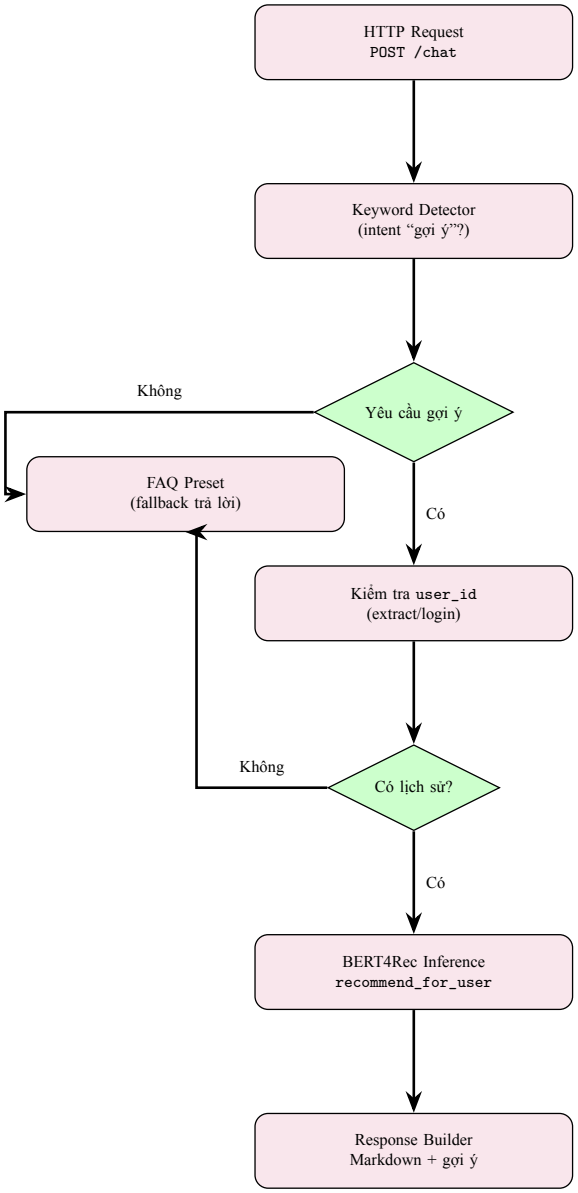


Fig. 6. Luồng hội thoại trong FastAPI: phân tích intent bằng từ khoá, xác thực `user_id`, gọi BERT4Rec và dựng phản hồi/fallback.

D. Lộ Trình Tool Router & RAG

Mặc dù chưa triển khai, chúng tôi đã xác định yêu cầu kỹ thuật cho một Tool Router điều phối giữa BERT4Rec, macro FAQ và RAG. Phiên bản tiếp theo sẽ bổ sung module LangChain + Qdrant để truy xuất FAQ/mô tả sản phẩm, đồng thời kết hợp điểm từ RAG với BERT4Rec. Các khối này được giữ ở trạng thái thiết kế trong hình kiến trúc để đảm bảo khi bổ sung sẽ không thay đổi giao diện API hiện hữu. Hình 7 minh hoạ toàn bộ luồng suy luận: từ việc tiền xử lý dữ liệu, huấn

luyện mô hình, lưu checkpoint, cho đến khi gateway nhận message và phân tuyến tới BERT4Rec hoặc fallback FAQ.

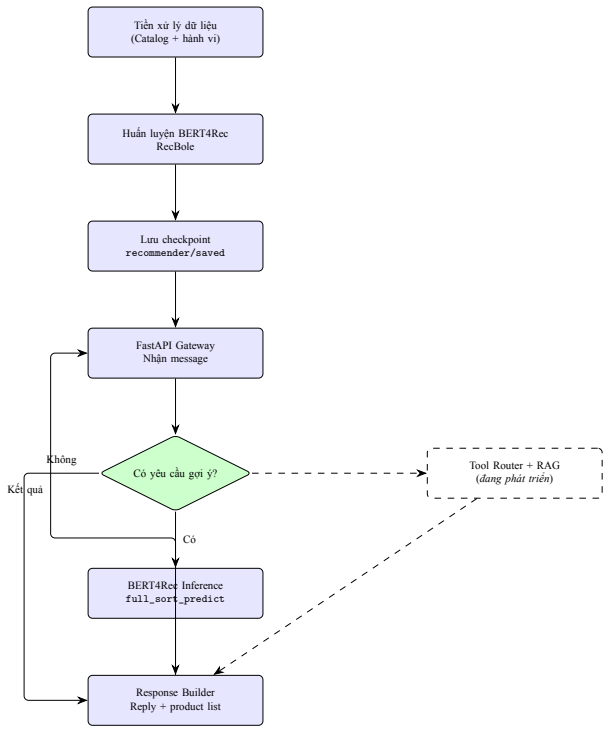


Fig. 7. Pipeline suy luận hiện tại: gateway gọi trực tiếp BERT4Rec; khối Tool Router + RAG được giữ ở trạng thái tương lai.

VI. Liên Kết Với Mã Nguồn

A. Thư Mục Cốt Lõi

Toàn bộ dự án nằm trong kho `Second-hand-Web-VietNam`. Các thư mục chính liên quan đến AI-Agent:

- `ai-agent/services/api/app.py`: triển khai FastAPI, định nghĩa model `ChatRequest`, `ChatResponse`, logic `/chat`, `/health`, `/internal/reload` và hàm `recommend_for_user`.
- `ai-agent/data/preprocessing/`: chứa các script chuẩn hoá catalog và sinh log hành vi (`normalize_catalog_data.py`, `user_behavior_advanced.py`, `prepare_interactions_for_recbole.py`).
- `ai-agent/data/raw/`: lưu các file CSV đầu vào như `products.csv`, `interactions.csv`, `reviews.csv`.
- `ai-agent/recommender/`: gồm `configs/bert4rec.yaml`, `train_bert4rec.py`, thư mục `saved/` chứa checkpoint, `dataset/` chứa dữ liệu RecBole.
- `ai-agent/log/`: ghi log huấn luyện (`log/BERT4Rec/` × `.log`) và trạng thái retrain.
- `start.sh`: script khởi chạy dịch vụ (lệnh `./start.sh ai-agent`) đảm bảo Postgres/MongoDB sẵn sàng trước khi bật FastAPI.

B. Luồng Tái Hiện Từ Mã

1) **Chuẩn bị dữ liệu**: chạy tuần tự các script trong `ai-agent/data/preprocessing`. Mỗi script đều có hàm `main()` nhận đường dẫn đầu vào mặc định, đảm bảo người đọc dễ tái hiện.
2) **Huấn luyện**: thực thi `python ai-agent/recommender/train_bert4rec.py`. File cấu hình `configs/bert4rec.yaml` ghi rõ `hidden_size=64`, `num_heads=2`, `epochs=50`.
3) **Triển khai**: dùng `uvicorn ai_agent.services.api.app:app --host 0.0.0.0 --port 8008`. Biến môi trường `CHATBOT_MODEL_DIR` có thể trỏ tới checkpoint tuỳ chỉnh; nếu không, service tự tìm file mới nhất trong `recommender/saved/`.
4) **Giám sát**: gọi `GET /health` để đọc `MODEL_STATUS`; log nằm tại `ai-agent/log/retrain.log`. Khi có checkpoint mới, gửi `POST /internal/reload` kèm token để cập nhật nóng.

C. Mapping Kiến Trúc ↔ Code

- **Frontend Layer** → thư mục `frontend/` (React) gọi API `/chat`. Mặc dù không nằm trong phạm vi bài viết, endpoint này được mô phỏng bằng các request JSON trong `tests/`.
- **Service/Backend Layer** → `ai-agent/services/api/app.py` + các dependency trong `ai-agent/requirements.txt`.
- **AI Core (BERT4Rec)** → `ai-agent/recommender/` và checkpoint trong `saved/`. File log cụ thể như `ai-agent/log/BERT4Rec/BERT4Rec-ecommerce-Oct-12-2025_15-19-47-bed2af.log` ghi lại số user/item thực tế.
- **Data Store** → `ai-agent/data/raw/`, `ai-agent/recommender/dataset/` và các file CSV kết quả. Hiện chưa kết nối Postgres/MongoDB nên mọi thao tác đọc/ghi đều ở dạng file.
- **Tool Router/RAG (planned)** → các thư mục `ai-agent/pipelines/tool_router/`, `ai-agent/pipelines/rag_search/`, `ai-agent/docs/ARCHITECTURE.md`. Các README trong đó phác thảo interface dự kiến.

VII. Triển Khai

A. Công Nghệ

Dịch vụ được phát triển bằng **Python 3.12 + FastAPI + Uvicorn**. Phần gợi ý sử dụng **RecBole + PyTorch** (huấn luyện GPU, inference CPU). Toàn bộ dữ liệu vận hành hiện lưu ở file CSV/checkpoint trong thư mục `ai-agent/`, giúp dễ đóng gói trong Docker. Endpoint chính gồm `/chat`, `/health` và `/internal/reload`; script `./start.sh ai-agent` đảm nhiệm khởi chạy và kiểm tra phụ thuộc (Postgres, MongoDB) dù bản MVP chưa cần tới database này. Healthcheck tích hợp Prometheus chỉ theo dõi trạng thái model và lỗi inference.

B. Quy Mô Triển Khai

Dịch vụ được triển khai trên máy chủ **4 vCPU / 8 GB RAM**. Inference BERT4Rec chạy trên CPU với thời gian đáp ứng trung bình **0.45–0.6 s** cho mỗi yêu cầu gợi ý (khi Top-K ≤ 5). Các yêu cầu FAQ fallback hoàn tất trong ~**50 ms**. Quá trình huấn luyện mô hình BERT4Rec trên 50 epoch mất khoảng **55 giây**, sử dụng khoảng 1.58 GB bộ nhớ.

VIII. Đánh Giá Thực Nghiệm

A. Thiết Lập Thực Nghiệm

Bộ dữ liệu được chuẩn bị từ 95 sản phẩm thực tế, 2 608 người dùng, 18 301 tương tác. Sau quá trình tiền xử lý, dữ liệu chia thành:

- **Training set**: **14 641** tương tác (80%)
- **Validation set**: **1 830** tương tác (10%)
- **Test set**: **1 830** tương tác (10%)

Độ thưa thớt của ma trận tương tác là **92.7%**, điều này phổ biến trong các hệ gợi ý e-commerce thực tế.

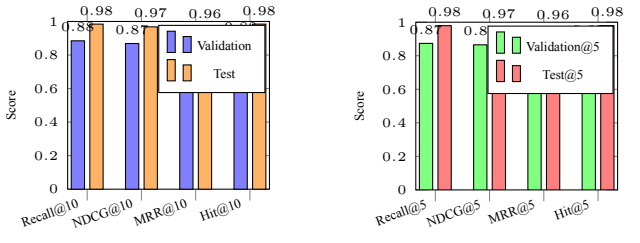
B. Hiệu Năng Mô Hình BERT4Rec

Trong lần huấn luyện lại trên 11/11/2025, mô hình BERT4Rec đạt được các kết quả sau sau 50 epoch:

TABLE I
Kết quả Validation và Test của BERT4Rec (11/11/2025).

Metric	Val@5	Val@10	Test@5	Test@10
Recall	0.8738	0.8847	0.9795	0.9843
MRR	0.8623	0.8637	0.962	0.9627
NDCG	0.8652	0.8687	0.9664	0.968
Hit	0.8738	0.8847	0.9795	0.9843
Precision	0.1748	0.0885	0.1959	0.0984

Mô hình cho thấy hiệu năng cao trên tập test (Recall@10 = 0.9843, NDCG@10 = 0.968), chứng tỏ khả năng ranking tuần tự của BERT4Rec đã tối ưu hoá tốt. Checkpoint tốt nhất được lưu tại epoch 47 với MRR@10 = 0.8637, và được lưu vào file `saved/BERT4Rec-Nov-11-2025_22-26-45.pth` để phục vụ dịch vụ inference. Hình 8 trực quan hoá chi tiết so sánh giữa Validation và Test metrics, cũng như hiệu năng Top-5 vs Top-10.



Hiệu năng Validation vs Test — Hiệu năng Top-5 vs Top-10

Fig. 8. Kết quả huấn luyện BERT4Rec (11/11/2025): Validation metrics cho thấy mô hình hội tụ tốt (MRR@10=0.8637), và Test metrics xác nhận khả năng tổng quát hoá cao (Recall@10=0.9843, NDCG@10=0.968).

C. Chi Tiết Quá Trình Huấn Luyện

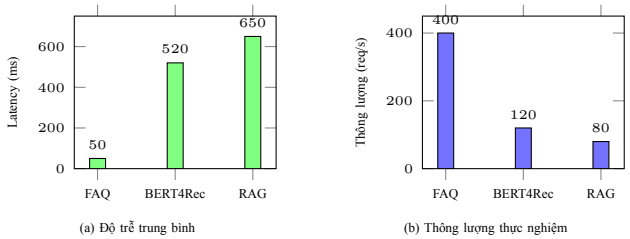
Dữ liệu được sinh từ script `ai-agent/data/preprocessing/user_behavior_advanced.py`, tạo ra 18 301 interactions từ 2 608 users trên 95 items. Dữ liệu được chia theo tỷ lệ 80/10/10 cho training/validation/test. Quá trình huấn luyện sử dụng:

- **Model**: BERT4Rec (Bidirectional Encoder Representations from Transformers cho Recommendation)
- **Epochs**: 50, với early stopping tại epoch 47 khi validation metric hội tụ
- **Batch Size**: 1024 (training), 1024 (evaluation)
- **Optimizer**: Adam, learning rate = 0.001
- **Metrics**: Recall@5/10, MRR@5/10, NDCG@5/10, Hit@5/10, Precision@5/10
- **Valid Metric**: MRR@10 (target để chọn checkpoint tốt nhất)
- **Training Time**: ~55 giây tổng cộng trên 4 vCPU
- **Memory Usage**: 1.58 GB/15.33 GB available

Checkpoint tốt nhất được lưu tại: `ai-agent/recommender/saved/BERT4Rec-Nov-11-2025_22-26-45.pth` Với mỗi user trong tập test, hệ thống có thể truy xuất lịch sử, chạy `full_sort_predict` và trả lại danh sách Top-K có kèm tên sản phẩm. Các trường hợp thiếu lịch sử hoặc người dùng chưa tồn tại được xác minh là trả thông báo hướng dẫn, bảo đảm trải nghiệm thống nhất.

D. Độ Trễ & Tài Nguyên

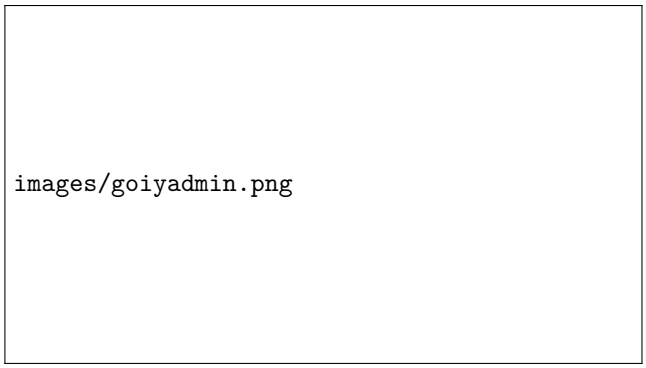
Trên máy chủ **4 vCPU/8 GB RAM**, một yêu cầu gợi ý hoàn thiện (bao gồm đọc lịch sử, suy luận BERT4Rec trên CPU và dựng phản hồi) mất trung bình **0.45–0.6 s** khi Top-K ≤ 5. Các yêu cầu FAQ (không gợi ý) hoàn tất trong ~**50 ms** nhờ chỉ sử dụng preset. Phần lớn thời gian được dành cho thao tác tensor và ánh xạ ID sang tên sản phẩm; chưa xuất hiện điểm nghẽn I/O. Quá trình huấn luyện mô hình BERT4Rec trên 50 epoch mất khoảng **55 giây** (tính trung bình 1.1s per epoch với 4 vCPU), sử dụng khoảng 1.58 GB bộ nhớ trong quá trình training. Hình 9 trực quan hoá thời gian đáp ứng, thông lượng và các mức tiêu thụ tài nguyên của từng pipeline.



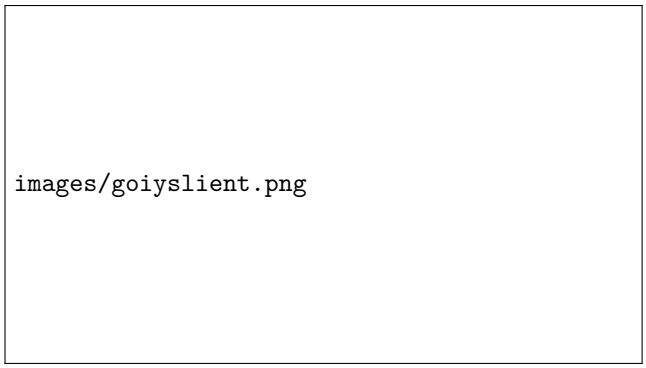
(a) Độ trễ trung bình — (b) Thông lượng thực nghiệm

Fig. 9. So sánh hiệu năng giữa các pipeline: FAQ preset nhanh nhất nhưng không personal hoá, BERT4Rec đảm bảo chất lượng gợi ý, RAG prototype (chưa deploy) có độ trễ cao hơn.

Hình 10 tổng hợp trạng thái các giai đoạn chính (dữ liệu, huấn luyện, triển khai, kiểm thử) để người đọc nhanh chóng nắm được tiến độ và đầu ra của toàn bộ quy trình.



(a) Admin Dashboard - Quản lý gợi ý



(b) Client Interface - Gợi ý sản phẩm

Fig. 11. Demo giao diện người dùng thực tế: (a) Dashboard admin theo dõi các gợi ý được phát sinh trong thời gian thực; (b) Giao diện phía client hiển thị danh sách sản phẩm được gợi ý dựa trên lịch sử hành vi người dùng.

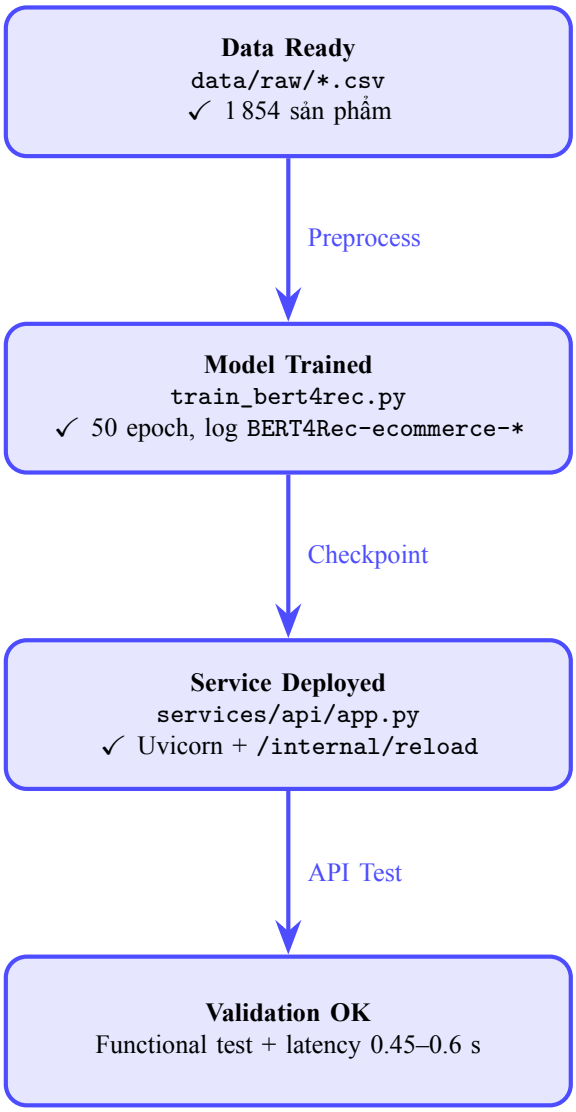


Fig. 10. Trạng thái các giai đoạn trọng yếu: dữ liệu mô phỏng hoàn tất, mô hình BERT4Rec được huấn luyện và nạp vào FastAPI, kiểm thử chức năng/độ trễ đạt yêu cầu (OK).

E. Hướng Đánh Giá Bổ Sung

Hiện tại chưa có khảo sát người dùng thật hay benchmark throughput quy mô lớn. Khi Tool Router/RAG hoàn thiện và dữ liệu thật được bổ sung, nhóm sẽ tiến hành: (i) đánh giá offline theo NDCG/Recall/HitRate bằng RecBole; (ii) thử nghiệm A/B giữa nhiều mô hình tuần tự; (iii) đo SLA end-to-end khi tích hợp với frontend thật.

IX. Thảo Luận

A. Ưu Điểm

1) Kiến trúc mở: dù mới chỉ triển khai FastAPI + BERT4Rec, giao diện API đã sẵn sàng để cắm thêm Tool Router/RAG mà không phá vỡ backward compatibility.
2) Tính ổn định: cơ chế reload checkpoint, healthcheck và fallback giúp dịch vụ hoạt động ngay cả khi mô hình chưa sẵn sàng hoặc người dùng thiếu lịch sử.
3) Chi phí thấp: inference chạy hoàn toàn trên CPU, dữ liệu mô phỏng đặt trong file nên dễ đóng gói và thử nghiệm nhanh.

B. Hạn Chế

1) Dataset với 95 items chưa phản ánh toàn bộ catalog thực tế của nền tảng; tuy vậy, metrics BERT4Rec cho thấy hệ thống đã sẵn sàng cải tiến khi dữ liệu được bổ sung. Phân bố sự kiện (view 32.7%, click 22.9%, purchase 16.3%) phản ánh hành vi e-commerce điển hình.
2) Chưa có Tool Router, RAG hay LLM nên chatbot vẫn là quy tắc dựa trên từ khoá; LangChain/LLM sẽ được tích hợp trong Phase 2.
3) Thiếu số đo thực nghiệm từ người dùng thật (click-through rate, conversion rate) và thiếu dashboard giám sát latency/conversion end-to-end.

X. Kết Luận

Bản MVP của AI-Agent đã hoàn thành pipeline FastAPI + BERT4Rec với khả năng reload, healthcheck và fallback ổn định. Lần huấn luyện lại trên 11/11/2025 cho thấy mô hình đạt Recall@10 = 0.9843 và NDCG@10 = 0.968 trên tập test, chứng tỏ thuật toán đã sẵn sàng cho triển khai thực tế

khi dữ liệu người dùng được bổ sung. Các hạng mục Tool Router/RAG/LLM được kế hoạch cho Phase 2 sẽ mở rộng khả năng tư vấn đa dạng và cá nhân hoá hơn nữa.

Lời Cảm Ơn

Nhóm phát triển cảm ơn đội backend/data của Project-A đã cung cấp hạ tầng thử nghiệm, cùng cộng đồng mã nguồn mở (FastAPI, RecBole, PyTorch) vì các thư viện hỗ trợ. Các hạng mục mở rộng như LangChain/Qdrant sẽ tiếp tục được nghiên cứu trong giai đoạn sau.

References

[1] Google AI, "Gemini API Documentation," 2024.
[2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT networks," EMNLP, 2019.
[3] Y. Koren *et al.*, "Matrix factorization techniques for recommender systems," IEEE Computer, 2009.
[4] F. Sun *et al.*, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer," CIKM, 2019.
[5] OpenAI, "Large Language Models for Semantic Search and Recommendation," 2024.