BÁO CÁO CUỐI KÌ
Hệ thống Gợi ý Mỹ phẩm Hybrid (ALS + BPR + BERT)

Trường Đại học Khoa học Tự nhiên - ĐHQGHN

Ngày 5 tháng 12 năm 2025

1. GIỚI THIỆU

Đặt vấn đề

Bối cảnh ngành thương mại điện tử Việt Nam:

- Thị trường TMĐT Việt Nam 2023: 16.4 tỷ USD, CAGR > 20%.
- Ngành mỹ phẩm - làm đẹp: tăng trưởng nhanh, cạnh tranh cao.
- Người dùng yêu cầu cá nhân hóa trải nghiệm mua sắm (91% chọn đề xuất phù hợp nhu cầu cá nhân).

Thách thức dữ liệu mỹ phẩm Việt Nam:

- **Dữ liệu thưa (Sparsity)**: 99.1% interactions bị thiếu.
- **Rating lệch (Skewed Ratings)**: 95% đánh giá 5 sao.
- **Cold-start users**: 30-40% người dùng hàng tháng là mới.

Mục tiêu đề tài

Mục tiêu tổng quát: Xây dựng hệ thống gợi ý lai (Hybrid Recommender) cho mỹ phẩm, giải quyết sparsity, rating skew và cold-start, nâng cao cá nhân hóa.

Mục tiêu cụ thể:

1. **Hệ thống Hybrid CF + Content-based:**
   - CF: ALS với implicit feedback, regularization mạnh.
   - Content-based: PhoBERT embeddings, similarity trong không gian ngữ nghĩa.
   - Kết hợp tuyến tính với cơ chế chuyển đổi CF / Content-based.
2. **Ứng dụng Deep Learning (BERT) cho ngôn ngữ Việt:**
   - Tiền xử lý text: chuẩn hóa Unicode, tách từ, chuẩn hóa tên thành phần.
   - Trích embedding 768-dim, tính cosine similarity, xây dựng knowledge graph.

2. CHIẾN LƯỢC NGƯỜI DÙNG VÀ PHÂN KHÚC

2.2 Chuẩn hóa tiếng Việt và sửa lỗi chính tả

Thách thức dữ liệu: teencode, lỗi gõ nhanh, từ dính, token sai, emoji mã hoá.

Quy trình Hybrid AI–Human gồm 6 giai đoạn:

1. Trích xuất từ vựng (50k+ token).
2. Tiền lọc: bỏ từ sạch, thương hiệu, rác, từ có dấu gạch dưới.
3. Gọi AI (Gemini 2.5 Flash) theo **chunk 200 từ**.
4. Kiểm định con người: top 10k từ xuất hiện nhiều.
5. Hậu xử lý: fix từ ghép tách sai ("đư ợc" $\rightarrow$ "được").
6. Cập nhật từ điển viết tắt thủ công.

Hiệu quả: sửa đúng 94.2%, sai 3.1%, tiết kiệm 99% chi phí API.

2.3 Phân khúc người dùng (User Segmentation)

- **Dữ liệu tương tác:**
  - Chỉ $\approx 8.6\%$ người dùng có ≥ 2 tương tác.
- **Chiến lược định tuyến theo loại user:**
  - **Trainable Users:**
    - Collaborative Filtering (ALS/BPR).
    - Reranking theo chất lượng bình luận.
  - **Cold-start Users:**
    - Content-based: PhoBERT embedding similarity.
    - Popularity-based fallback.

2.4 Tích hợp BERT Embeddings

- **Input kết hợp:**
  - Tên sản phẩm
  - Công dụng
  - Thành phần
  - Loại da phù hợp
- **Output:** Vector ngữ nghĩa (BERT embedding).
  - Dùng tính độ tương đồng nội dung.
  - Dùng làm **khởi tạo tham số** cho ALS $\rightarrow$ giúp hội tụ nhanh và chính xác hơn.

3. HUẤN LUYỆN MÔ HÌNH

3.1 Alternating Least Squares (ALS)

ALS là thuật toán Matrix Factorization tối ưu cho dữ liệu implicit.

- **Đầu vào**: Ma trận Confidence dạng CSR (rất thưa, density 0.11%).
- Confidence score:

$$c_{ui} = r_{ui} + q_{ui}$$

- Giá trị $c_{ui} \in [1, 6]$: phân biệt rating 5 sao thật vs 5 sao nhiễu.
- Kích thước: $26{,}000 \times 2{,}200$, nnz $\approx 65{,}000$.
- Thách thức: dữ liệu cực thưa $\Rightarrow$ dễ trôi dạt embedding.

ALS – BERT Initialization (Đóng góp chính)

Vấn đề: Khởi tạo Gaussian ngẫu nhiên khiến cold items bị drift.
Giải pháp: **BERT Initialization**.

1. Trích xuất embedding của sản phẩm bằng **Vietnamese_Embedding**:

$$e_i = \text{MeanPooling}(\text{Vietnamese\_Embedding}(t_i)) \in \mathbb{R}^{1024}$$

2. Giảm chiều bằng **TruncatedSVD** xuống $d = 64$.
3. Căn chỉnh theo `item_to_idx`.
4. Gán làm item factors khởi tạo:

$$V^{(0)} = \text{AlignedBERTEmbeddings}$$

Lợi ích:

- Hội tụ nhanh hơn (15 iterations).
- Cold items được neo ngữ nghĩa.
- Chuyển tri thức từ NLP $\rightarrow$ CF.

Kết quả: Recall@10 tăng **+33%**.

- **factors = 64** không gian tiềm ẩn
- **regularization = 0.1** giữ ổn định BERT init
- **alpha = 5** phù hợp confidence [1,6]
- **iterations = 15**

Hàm mất mát ALS:

$$L = \sum_{u,i} C_{ui}(p_{ui} - u_u^\top v_i)^2 + \lambda \left(\sum_u \|u_u\|^2 + \sum_i \|v_i\|^2 \right)$$

Tối ưu luân phiên:

$$u_u = (V^\top C_u V + \lambda I)^{-1} V^\top C_u p_u$$

$$v_i = (U^\top C_i U + \lambda I)^{-1} U^\top C_i p_i$$

Huấn luyện dùng **implicit C++ backend**, thời gian: **1–2 phút**.

3.2 Bayesian Personalized Ranking (BPR)

Mục tiêu: tối ưu **ranking** (pairwise), không dự đoán rating tuyệt đối.
Đầu vào: bộ ba (u, i, j)

- i: positive item ($r_{ui} \geq 4$)
- j: negative item (chưa tương tác hoặc rating thấp)

Số lượng triplets:

$$|D_S| = 65{,}000 \times 5 = 325{,}000$$

Mục tiêu tối ưu:

$$\hat{r}_{ui} > \hat{r}_{uj}, \quad \forall (u, i, j) \in D_S$$

BPR – Hard Negative Mining

Vấn đề: random negatives quá dễ gradient gần 0.
Giải pháp: Dual Hard Negative Mining

1. **Explicit hard negatives**: rating thấp (≤ 3).
2. **Implicit hard negatives**: sản phẩm phổ biến nhưng user không mua.

Mixing:

$$p(\text{hard}) = 0.3, \quad p(\text{random}) = 0.7$$

Lợi ích:

- Gradient giàu thông tin hơn.
- Giảm popularity bias.
- Học được ranh giới tinh tế giữa các sản phẩm tương tự.

BPR – Loss Function

Hàm mất mát BPR:

$$L_{\text{BPR}} = -\sum_{(u,i,j)} \ln \sigma(\hat{r}_{ui} - \hat{r}_{uj}) + \lambda(\|u_u\|^2 + \|v_i\|^2 + \|v_j\|^2)$$

Huấn luyện:

- SGD + mini-batch
- Xavier init cho embeddings
- Adam optimizer
- 50–80 epochs

Kết quả: cải thiện ranking cho các user có hành vi đa dạng.

3.3 Model Registry & Versioning

Tất cả artifacts được lưu trữ theo version:
`artifacts/cf/als/v1_20251127/`

- **als_U.npy** — user factors $(26000, 64)$
- **als_V.npy** — item factors $(2200, 64)$
- **als_params.json** — hyperparameters
- **als_metrics.json** — Recall@10, NDCG@10
- **als_metadata.json** — score ranges, BERT init info, git commit

Mục đích:

- Tái lập mô hình (reproducibility)
- So sánh version
- Hỗ trợ Hybrid Reranking (chuẩn hóa CF scores)

4. HỆ THỐNG SERVICE VÀ HYBRID RERANKING

4.1 Kiến trúc Serving

Yêu cầu hệ thống:

- Độ trễ P95 $<$ 100ms
- Availability $\geq$ 99.9%
- Throughput $\geq$ 100 req/s

Luồng xử lý (Latency Breakdown):

Client $\rightarrow$ API Gateway (5ms) $\rightarrow$ User Router (10ms) $\rightarrow$ Scoring Engine (50–8

Thành phần chính:

- API Gateway: validation, auth, rate limit.
- User Router: quyết định CF / Fallback.
- Scoring Engine: CF scoring hoặc Content scoring.
- Reranking: tích hợp multi-signal (nếu bật).

Kiến trúc Phân Tầng (Layered Architecture)

- L1: **API Layer** – FastAPI entrypoint
- L2: **Business Logic** – Recommender, Fallback, Reranker, Cache
- L3: **Search Layer** – PhoBERT semantic search
- L4: **Configuration Layer**

Nguyên tắc:

- Single Responsibility
- Dependency Injection
- Graceful degradation với multi-layer fallback

User Routing: Trainable vs Cold-start

Phân đoạn người dùng:

- Trainable users (8.6%): $|H_u| \geq 2$ và có rating ≥ 4.
- Cold-start users (91.4%): còn lại.

Routing rule:

$$path(u) = \begin{cases} \text{CF Path}, & \text{nếu user trainable} \\ \text{Fallback Path}, & \text{ngược lại} \end{cases}$$

Ý nghĩa:

- CF chỉ dùng cho user đủ dữ liệu.
- Cold-start ưu tiên Content + Popularity.

CF Path: Scoring Pipeline

6 bước xử lý:

1. Index mapping user $\rightarrow u_{\text{cf}}$
2. CF scoring: $\hat{r}_{ui} = u^\top v_i$
3. Seen-item filtering
4. Attribute filtering
5. Top-K (lấy 5K nếu có reranking)
6. Hybrid Reranking

Fallback Path: Content-based + Popularity

Score components:

$$s_{\text{content}}(u, i) = \cos(\tilde{e}_u, \tilde{e}_i)$$

$$s_{\text{pop}}(i) = \frac{\log(1 + \text{sold}_i)}{\max_j \log(1 + \text{sold}_j)}$$

Hybrid score:

$$S_{\text{fallback}}(u, i) = w_{\text{content}} * s_{\text{content(u,i)}} + w_{\text{pop}} * s_{\text{pop(i)}} + w_{\text{quality}} * s_{\text{quality}}$$

New user ($m = 0$): chỉ dùng popularity.

Performance Optimization & Fallback

Caching:

- Popular items cache: top-50
- User profile cache (LRU)
- Similarity cache (2,500 pairs)

Batch processing:

$$\hat{R}_{\text{batch}} = U_{\text{batch}} V^{\top}$$

Fallback layers:

- CF lỗi $\rightarrow$ Content-based
- BERT lỗi $\rightarrow$ Popularity-only
- Filter xoá hết items $\rightarrow$ Bỏ filter

Mục tiêu:

- Tối ưu relevance (độ phù hợp)
- Tăng diversity (đa dạng)
- Kết hợp nhiều tín hiệu scoring

Input: Candidate set C_u từ CF hoặc Fallback.
Output: Top-K danh sách cuối cùng.

4.3 Smart Search Integration

Smart Search sử dụng:

- Vietnamese Embedding
- Metadata + caching dùng chung với Serving

Luồng tích hợp:

$$\text{CF/Fallback Output} \rightarrow \text{Hybrid Reranking} \rightarrow \text{Response}$$

$$\text{Smart Search Output} \longrightarrow \text{Hybrid Reranking}$$

Lợi ích:

- Một pipeline thống nhất cho search + recommend
- Tối ưu reuse caching, model loading, monitoring

5. ĐÁNH GIÁ VÀ THỰC NGHIỆM

5.1 So sánh tổng thể các mô hình Collaborative Filtering

Model	R@5	R@10	R@20	NDCG@10	Coverage
Baselines					
Random	0.0024	0.0053	0.0123	0.0022	1.0000
Popularity	0.0336	0.0550	0.1276	0.0283	0.0162
ALS Variants					
ALS (artifact)	0.1523	0.1828	0.2261	0.1423	0.5910
ALS (checkpoint)	0.1557	0.1842	0.2288	0.1445	0.5798
BERT-ALS Variants					
BERT-ALS (best)	0.1542	**0.1888**	0.2256	0.1463	0.2389
BERT-ALS (grid search)	0.1554	0.1813	0.2246	0.1414	0.3626
BPR					
BPR (advanced, 128d)	0.0947	0.1029	0.1179	0.0895	0.6852

- BERT-ALS (best) đạt Recall@10 = **0.1888** (tăng 243.6% so với Popularity).
- Tất cả mô hình CF vượt trội so với baseline với p-value ≈ 0.
- BPR có **coverage cao nhất** (68.5%), nhưng Recall thấp hơn ALS/BERT-ALS.

5.2 Cải thiện so với Popularity Baseline

Mức cải thiện Recall@10 so với Popularity Baseline

Model	Recall@10	Improvement
BERT-ALS (best)	0.1888	+243.6%
ALS (checkpoint)	0.1842	+235.2%
ALS (artifact)	0.1828	+232.6%
BERT-ALS (grid_search)	0.1813	+230.0%
ALS-ColdAug	0.1784	+224.7%
BERT-ALS-ColdAug	0.1765	+221.1%
BPR (advanced)	0.1029	+87.2%
Popularity (baseline)	0.0550	–

- Tất cả mô hình CF đều vượt trội so với baseline một cách có ý nghĩa thống kê.
- BERT-ALS (best) có cải thiện cao nhất **+243.6%**, chứng minh lợi ích của khởi tạo từ Vietnamese Embedding.
- BPR có improvement thấp hơn (+87.2%) nhưng đạt **coverage cao nhất** (68.5%).

5.3 So sánh ALS vs BERT-ALS

Hiệu quả của Vietnamese Embedding Initialization

Metric	ALS	BERT-ALS	Change
Recall@10	0.1842	0.1888	+2.5%
NDCG@10	0.1445	0.1463	+1.2%
Coverage	0.5798	0.2122	-63.4%
Diversity	0.4521	0.2021	-55.3%

- BERT-ALS cải thiện Recall và NDCG nhưng giảm đáng kể coverage và diversity :
- Coverage: giảm 63.4%
- Diversity: giảm 55.3%
- Nguyên nhân: do BERT embeddings tập trung vào các items có ngữ nghĩa tương đồng

5.4 Hiệu quả Hybrid Reranking

Kết quả:

- **Recall@20 tăng**: $+9.2\%$ $\rightarrow$ cải thiện long-tail.
- **Recall@10 giảm nhẹ**: -1.8% (không có ý nghĩa thống kê).
- **Diversity giảm**: -2.2%.
- **Coverage giảm**: -41.1%.
- **Latency**: tăng từ 0.56ms $\rightarrow$ 2.72ms (vẫn $<$ 10ms).

Kết luận:

- Hybrid không cải thiện đáng kể so với CF-only vì CF đã tích hợp content signal từ BERT.
- Hybrid phù hợp cho: long-tail (K lớn), cold-start users.

5.5 So sánh Coverage vs Recall Trade-off

- **BERT-ALS**: Recall cao nhất nhưng Coverage thấp → phù hợp accuracy-focused scenarios.
- **ALS**: căn bằng tốt giữa recall (0.18) và coverage (0.59) - phù hợp cho production
- **BPR**: Coverage cao nhất (0.69) nhưng recall thấp - phù hợp cho diversity-focused scenarios
- Lựa chọn mô hình phụ thuộc: **Accuracy vs Diversity**.

Hiệu quả của BERT Initialization

- Recall@10 +2.5% so với ALS không có BERT initialization.
- Giảm cold-start cho sản phẩm mới nhờ embeddings có ngữ nghĩa.
- Embeddings có ngữ nghĩa từ đầu, không phụ thuộc hoàn toàn vào interactions
- Đặc biệt hiệu quả trong domain mỹ phẩm Việt Nam với các từ chuyên ngành
- SVD projection (1024 $\rightarrow$ 64 dims) giữ lại 64.9% variance.

5.4 Thảo luận – Điểm mạnh

- Cải thiện lớn: BERT-ALS tăng 243.6% so với baseline.
- Tất cả mô hình có ý nghĩa thống kê ($p < 0.05$).
- Nhiều lựa chọn model cho accuracy/diversity.
- Latency $<$ 50ms – đáp ứng real-time.
- Hệ thống mở rộng tốt (modular, scalable).

Hạn chế

- Sparsity cao: 91.3% user cold-start.
- Rating skew: 95% ratings là 5 sao, giảm khả năng phân biệt preference.
- Coverage thấp cho BERT-ALS (21.2%).
- Hybrid không cải thiện đáng kể cho trainable users.

Hướng cải tiến

- Contrastive learning để cải thiện embeddings.
- Knowledge graph cho cold-start users.
- Cache pre-computed similarities để giảm latency.
- A/B testing với real users để đánh giá business metrics.

6. THIẾT KẾ HỆ THỐNG VÀ XỬ LÝ DỮ LIỆU

6.1 Kiến trúc tổng quan hệ thống

Hệ thống RabbitMart + VieComRec được thiết kế theo mô hình **Client–Server nhiều tầng**. Gồm các thành phần:

- **Frontend (ReactJS – cổng 3000)** SPA tối ưu UI/UX, giao diện người dùng.
- **Backend (Node.js/Express – cổng 5000)** Xử lý nghiệp vụ, xác thực, điều phối dữ liệu (API Gateway).
- **Database (MongoDB 6.0 – cổng 27017)** Lưu trữ người dùng, sản phẩm, đơn hàng, hành vi.
- **Recommender (FastAPI – cổng 8000)** Dịch vụ AI gợi ý chạy độc lập bằng Docker.

Các lớp giao tiếp qua REST API, hỗ trợ cả user đăng nhập và guest.

6.2 Phân hệ Frontend (ReactJS)

Công nghệ sử dụng:

- React 18.1 (SPA)
- Redux Toolkit – State management
- React Router DOM – Routing SPA
- Axios – Giao tiếp API
- Framer Motion – Animation UI

Các trang chính:

- Home – sản phẩm nổi bật + gợi ý từ VieComRec
- Products – phân trang + lọc + semantic search
- Product Detail – thông tin + similar items
- Cart, Checkout (Stripe), Wishlist
- Admin Panel – quản trị + AI Dashboard

Cấu trúc thư mục Frontend

```
client/src/
  api/                  # API clients
  components/           # UI components
  pages/                # Route pages
  product-card/         # Hiển thị sản phẩm
  home/                 # Trang chủ
  products/             # Danh sách sản phẩm
  product-detail/       # Chi tiết sản phẩm
  cart/ checkout/       # Giỏ hàng - Thanh toán
  wishlist/ order/      # Wishlist - Lịch sử đơn
  authentication/       # Login/Register
  admin/                # Admin Panel
```

Luồng xử lý dữ liệu phía Client

1. User truy cập trang (Home/Products)
2. Client gọi `loadRecommendations(userId)`
3. VieComRec trả về danh sách `product_id + score`
4. Client gọi Backend: `/api/products/arr`
5. Backend lấy dữ liệu MongoDB và trả về thông tin đầy đủ
6. Merge dữ liệu $\rightarrow$ render ProductCard

6.3 Phân hệ Backend (Node.js/Express)

Công nghệ sử dụng:

- Node.js 16, Express 4
- Mongoose 6 – kết nối MongoDB
- JWT – xác thực
- Stripe – thanh toán
- Axios – gọi VieComRec API

Vai trò chính:

- Xử lý logic nghiệp vụ
- Xác thực user (JWT)
- Kết nối MongoDB
- Làm API Gateway cho VieComRec

Các nhóm API chính

Authentication (/api/auth)

- Đăng ký, đăng nhập, verify token
- Wishlist: GET/PATCH

Products (/api/products)

- Lấy sản phẩm theo trang
- Chi tiết sản phẩm, reviews
- Similar items
- POST /arr – lấy nhiều sản phẩm theo ID

Orders (/api/orders) Tạo đơn hàng, xem đơn hàng, admin quản lý.
Recommendation Proxy (/api/recommend)

- recommend, search, similar
- health check VieComRec

6.4 Recommender Service (VieComRec)

Mô hình gợi ý hỗ trợ:

- ALS / BPR – Collaborative Filtering
- Vietnamese Embedding – Content-based
- Semantic Search tiếng Việt
- Hybrid Reranking: CF + content + popularity

Triển khai:

- FastAPI + Docker Compose
- Chạy độc lập, giao tiếp qua Backend proxy
- Fallback cold-start bằng content-based + popularity

VieComRec – API chính

- `/recommend` – gợi ý theo user
- `/search` – semantic search tiếng Việt
- `/similar_items` – sản phẩm tương tự
- `/scheduler/status` – monitoring pipeline
- `/health` – kiểm tra trạng thái

Vai trò: cung cấp kết quả gợi ý nhanh, score chuẩn hóa, hỗ trợ backend hợp nhất dữ liệu trước khi trả về cho frontend.

6.5 Phân hệ Database (MongoDB)

Triển khai:

- MongoDB 6 chạy qua Docker
- Document-based – linh hoạt với dữ liệu sản phẩm
- Hỗ trợ scale-out bằng replica sets

Lưu trữ:

- Users, Products, Orders
- Reviews, Behavior logs
- Text search index phục vụ tìm kiếm

7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 Kết luận

Hệ thống đã giải quyết thành công các thách thức chính:

1. **Dữ liệu thưa (Sparsity):** Dual-path routing: CF cho trainable users, content-based fallback (BERT + popularity) cho cold-start users. **100% requests được phục vụ.**
2. **Rating skew:** Sentiment-enhanced confidence score giúp phân biệt các tương tác thực sự tích cực.
3. **Ngữ nghĩa tiếng Việt:** BERT Initialization (BERT embeddings) cải thiện cold-start và chất lượng embeddings latent.
4. **Kiến trúc production-ready:** Latency P95 $<$ 100ms, availability 99.9%, pipeline tự động đảm bảo data freshness và model updates.

7.2 Hướng phát triển

Ngắn hạn:

- A/B Testing Framework: đánh giá hiệu quả reranking (CTR, conversion rate).
- Real-time Features: tích hợp session-based signals (recently viewed, cart items) vào scoring.

Trung hạn:

- Graph Neural Networks: mô hình hóa quan hệ user-item-attribute (heterogeneous graph).
- Multi-task Learning: đồng thời tối ưu click prediction và purchase prediction.

Dài hạn:

- Reinforcement Learning: contextual bandits hoặc Q-learning để học policy gợi ý tối ưu theo thời gian thực.
- Explainable Recommendations: sinh giải thích cho từng gợi ý dựa trên LLM.

Cảm ơn thầy và các bạn đã lắng nghe!