|  |  |  |
| --- | --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**  **KHOA CÔNG NGHỆ THÔNG TIN**  Description: C:\Documents and Settings\Administrator\Desktop\logo dai hoc_khong nen.png  **ĐỒ ÁN MÔN HỌC**  **Phát triển hệ thống khuyến nghị sản phẩm dựa trên collaborative filtering**  **Giảng viên hướng dẫn : ThS. Phạm Đình Tài**  **Sinh viên thực hiện : Nguyễn Lê Quốc Bảo**  **MSSV : 2100004053**  **Môn học : Khóa luận tốt nghiệp**  **Khóa : 2021**  Tp.HCM, …tháng ... năm 2025 | | |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**  **KHOA CÔNG NGHỆ THÔNG TIN**  Description: C:\Documents and Settings\Administrator\Desktop\logo dai hoc_khong nen.png  **ĐỒ ÁN MÔN HỌC**  **Phát triển hệ thống khuyến nghị sản phẩm dựa trên Collaborative Filtering**  **Giảng viên hướng dẫn : ThS. Phạm Đình Tài**  **Sinh viên thực hiện : Nguyễn Lê Quốc Bảo**  **MSSV : 2100004053**  **Môn học : Khóa luận tốt nghiệp**  **Khóa : 2021**  Tp.HCM, …tháng ... năm 2025 | |
| TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH  **TRUNG TÂM KHẢO THÍ** | | **KỲ THI KẾT THÚC HỌC PHẦN**  **HỌC KỲ …..… NĂM HỌC …….. - ….…** | |

**PHIẾU CHẤM THI TIỂU LUẬN/ĐỒ ÁN**

Môn thi: Khóa Luận Tốt Nghiệp………………………..Lớp học phần: 21DTH.HL13……

Nhóm sinh viên thực hiện :

1.Nguyễn Lê Quốc Bảo Tham gia đóng góp: 100%

Ngày thi: Phòng thi:

Đề tài tiểu luận/báo cáo của sinh viên :………………………………………………..

Phần đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tiêu chí (theo CĐR HP)** | **Đánh giá của GV** | | **Điểm tối đa** | **Điểm đạt được** |
| Cấu trúc của báo cáo |  | | **1** |  |
| Nội dung |  | | **8** |  |
| * Các nội dung thành phần |  | | **6.5** |  |
| * Lập luận |  | | **1** |  |
| * Kết luận |  | | **0.5** |  |
| Trình bày |  | | **1** |  |
| **TỔNG ĐIỂM** |  | |  |  |
|  | | **Giảng viên chấm thi**  *(ký, ghi rõ họ tên)* | | | |

# LỜI CẢM ƠN

Trước tiên, nhóm chúng tôi xin chân thành cảm ơn [Tên giảng viên hướng dẫn] đã tận tình hướng dẫn, hỗ trợ và định hướng chuyên môn trong suốt quá trình thực hiện dự án "Xây dựng hệ thống đề xuất phim sử dụng phương pháp Collaborative Filtering".

Chúng tôi cũng xin gửi lời cảm ơn đến các thầy cô bộ môn [Tên bộ môn/khoa], trường [Tên trường], đã tạo điều kiện học tập và cung cấp những kiến thức nền tảng quý báu trong suốt thời gian học tập và thực hiện dự án.

Bên cạnh đó, nhóm xin chân thành cảm ơn các bạn đồng hành, những người đã đóng góp ý kiến, chia sẻ tài liệu và hỗ trợ kỹ thuật để dự án được hoàn thiện tốt nhất.

Mặc dù đã rất nỗ lực, nhưng do giới hạn về thời gian và kinh nghiệm, báo cáo không tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được những góp ý quý báu từ quý thầy cô và bạn đọc để dự án được hoàn thiện hơn trong tương lai.

Xin chân thành cảm ơn!

# LỜI MỞ ĐẦU

Trong bối cảnh ngành công nghiệp giải trí số bùng nổ, số lượng nội dung phim ảnh được sản xuất và phát hành ngày càng tăng mạnh. Người dùng ngày nay phải đối mặt với vô vàn lựa chọn, dẫn đến tình trạng "quá tải thông tin" khi tìm kiếm những bộ phim phù hợp với sở thích cá nhân. Để giải quyết bài toán này, hệ thống gợi ý phim đã trở thành một phần không thể thiếu trong các nền tảng xem phim trực tuyến như Netflix, Disney+, HBO Max và Amazon Prime Video.

Trong số các phương pháp xây dựng hệ thống gợi ý, Collaborative Filtering (CF) là một trong những kỹ thuật phổ biến và hiệu quả nhất. CF tận dụng hành vi và đánh giá của người dùng để dự đoán sở thích cho những bộ phim mà họ chưa từng xem, từ đó cá nhân hóa trải nghiệm giải trí cho từng người dùng.

Dự án này được thực hiện với mục tiêu xây dựng một mô hình đề xuất phim dựa trên phương pháp Collaborative Filtering, nhằm:

Hiểu rõ hơn về mối liên hệ giữa người dùng và các bộ phim.

Tối ưu hóa trải nghiệm tìm kiếm phim theo hướng cá nhân hóa.

Tăng mức độ hài lòng và tỷ lệ gắn bó của người dùng đối với nền tảng cung cấp phim.

Thông qua dự án, chúng tôi không chỉ tập trung vào việc áp dụng lý thuyết CF cơ bản, mà còn khai thác các kỹ thuật nâng cao để cải thiện độ chính xác và khả năng mở rộng của hệ thống trong thực tế.

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Hình thức (Bố cục, trình bày, lỗi, các mục, hình, bảng, công thức, phụ lục, ):

2. Nội dung (mục tiêu, phương pháp, kết quả, sao chép, các chương, tài liệu,..):

3. Kết luận:

*TPHCM, Ngày tháng năm 202*

**Giáo viên hướng dẫn**

(Ký tên, ghi rõ họ tên)

# NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

1. Hình thức (Bố cục, trình bày, lỗi, các mục, hình, bảng, công thức, phụ lục, ):

2. Nội dung (mục tiêu, phương pháp, kết quả, sao chép, các chương, tài liệu,..):

3. Kết luận:

*TPHCM, Ngày tháng năm 202*

**Giáo viên phản biện**

(Ký tên, ghi rõ họ tên)

# MỤC LỤC

[LỜI CẢM ƠN 4](#_Toc197563793)

[LỜI MỞ ĐẦU 5](#_Toc197563794)

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN 6](#_Toc197563795)

[NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN 7](#_Toc197563796)

[MỤC LỤC 8](#_Toc197563797)

[DANH MỤC ẢNH 13](#_Toc197563798)

[CHƯƠNG I. GIỚI THIỆU 1](#_Toc197563799)

[1. Giới thiệu đề tài 1](#_Toc197563800)

[1.1. Lý do chọn đề tài 1](#_Toc197563801)

[1.2. Ý tưởng 2](#_Toc197563802)

[1.3. Mô tả đề tài, bài toán 3](#_Toc197563803)

[2. Tính thực tiễn của đề tài 4](#_Toc197563804)

[2.1. Tính thực tiễn và hiệu quả kinh tế của hệ thống gợi ý 4](#_Toc197563805)

[2.2. Xu hướng tất yếu và sự cần thiết của hệ thống gợi ý 4](#_Toc197563806)

[2.3. Lĩnh vực thời trang 5](#_Toc197563807)

[3. Công nghệ sử dụng 7](#_Toc197563808)

[3.1. Phần Deep Learning: 7](#_Toc197563809)

[3.2. Phần Web Development: 7](#_Toc197563810)

[4. Kết quả mong muốn 8](#_Toc197563811)

[CHƯƠNG II. CƠ SỞ LÝ LUẬN 11](#_Toc197563812)

[1. Xử lý ngôn ngữ tự nhiên 11](#_Toc197563813)

[1.1. Cơ sở 11](#_Toc197563814)

[1.2. Một số điểm nổi bật riêng của NLP 11](#_Toc197563815)

[1.3. Các điểm đặc biệt của NLP so với xử lý số, xử lý ảnh và xử lý âm thanh 11](#_Toc197563816)

[2. Các kiến trúc và bài toán nổi bật 13](#_Toc197563817)

[2.1. Mạng nơ-ron hồi tiếp (Recurrent Neural Network – RNN) 13](#_Toc197563818)

[2.2. LSTM và GRU (Long Short-Term Memory & Gated Recurrent Unit) 13](#_Toc197563819)

[2.3. Mạng CNN cho văn bản (Convolutional Neural Network – CNN) 14](#_Toc197563820)

[2.4. Transformer 14](#_Toc197563821)

[2.5. BERT (Bidirectional Encoder Representations from Transformers) 15](#_Toc197563822)

[2.6. Các kiến trúc khác và mô hình lai (Hybrid) 16](#_Toc197563823)

[2.7. Kết luận 16](#_Toc197563824)

[3. Xử lý dữ liệu cho các bài toán ngôn ngữ 17](#_Toc197563825)

[3.1. Mã hóa chữ thành số trong xử lý ngôn ngữ tự nhiên: encryption và hàm số học mã hóa 17](#_Toc197563826)

[3.2. hashing(hashing trick) 17](#_Toc197563827)

[3.3. Character level encoding 19](#_Toc197563828)

[3.4. Byte level encoding 21](#_Toc197563829)

[3.5. Tokenize và Vocabulary Mapping 22](#_Toc197563830)

[3.6. One-hot embedding 24](#_Toc197563831)

[3.7. Word Embeding cbow, skip -gram, word2vec 26](#_Toc197563832)

[4. Các kiến trúc và thuật toán nền tảng 28](#_Toc197563833)

[4.1. Kiến trúc Tranformer 30](#_Toc197563834)

[4.2. Positional Encoding 30](#_Toc197563835)

[4.3. Multi-Head Attention 31](#_Toc197563836)

[4.4. Masked Multi-Head Attention 32](#_Toc197563837)

[4.5. Feed Forward Layer 32](#_Toc197563838)

[CHƯƠNG III. MÔ HÌNH LÝ THUYẾT 34](#_Toc197563839)

[1. Kiến trúc Collaborative Filtering (CF) 34](#_Toc197563840)

[1.1. Nguyên lý hoạt động 34](#_Toc197563841)

[1.2. Thuật toán cơ bản 35](#_Toc197563842)

[1.3. Sự tối ưu và phát triển mới mẻ của CF 35](#_Toc197563843)

[1.4. Ứng dụng thực tế - Các công ty sử dụng CF 36](#_Toc197563844)

[1.5. Kết luận 37](#_Toc197563845)

[2. CF-based Encoder Transformer 38](#_Toc197563846)

[2.1. Nguyên lý hoạt động 38](#_Toc197563847)

[2.2. Thuật toán cơ bản 39](#_Toc197563848)

[2.3. Sự tối ưu và cải tiến mới mẻ 40](#_Toc197563849)

[2.4. Các công ty và ứng dụng thực tế 41](#_Toc197563850)

[2.5. Kết luận 42](#_Toc197563851)

[3. Dropout 43](#_Toc197563852)

[3.1. Cách hoạt động của Dropout 43](#_Toc197563853)

[3.2. Thuật toán tính toán cụ thể 43](#_Toc197563854)

[3.3. Liên hệ với CF-based Encoder Transformer 44](#_Toc197563855)

[4. Regularizers 45](#_Toc197563856)

[4.1. Nguyên lý cơ bản của Regularizers 45](#_Toc197563857)

[4.2. Tại sao cần Regularizers trong CF-based Encoder Transformer 47](#_Toc197563858)

[4.3. Ứng dụng thực tế trong Encoder Transformer 47](#_Toc197563859)

[5. GlobalAveragePooling1D 48](#_Toc197563860)

[5.1. Cách hoạt động của GlobalAveragePooling1D 48](#_Toc197563861)

[5.2. So sánh nhanh với Flatten 49](#_Toc197563862)

[6. Lớp Positional Encoding 50](#_Toc197563863)

[6.1. Cách hoạt động cụ thể 51](#_Toc197563864)

[6.2. Về mặt thuật toán 51](#_Toc197563865)

[7. kỹ thuật gộp input và Embeding riêng trước khi đưa vào Transformer 52](#_Toc197563866)

[8. Kiến trúc micro-service và sql server cho hệ thống backend 53](#_Toc197563867)

[9. Công nghệ cho backend và fonend của web 54](#_Toc197563868)

[CHƯƠNG IV. MÔ HÌNH THỰC NGHIỆM 55](#_Toc197563869)

[1. Dữ liệu 55](#_Toc197563870)

[1.1. Chuẩn bị dữ liệu 55](#_Toc197563871)

[1.2. Cách xây dựng dữ liệu mô phỏng: 55](#_Toc197563872)

[1.3. Tiêu chí của bộ dữ liệu 57](#_Toc197563873)

[1.4. Xử lý dữ liệu 59](#_Toc197563874)

[1.5. Mô tả phân bố dữ liệu: 61](#_Toc197563875)

[1.6. Đánh giá: 62](#_Toc197563876)

[1.7. Phân tích rủi ro tiềm ẩn từ phân bố dữ liệu: 62](#_Toc197563877)

[2. Xây dựng mô hình 63](#_Toc197563878)

[2.1. Khối Embedding đầu vào 63](#_Toc197563879)

[2.2. Khối Position Embedding 63](#_Toc197563880)

[2.3. Khối Encode (Transformer Encoder Block) 64](#_Toc197563881)

[2.4. Khối Fully Connected Output 65](#_Toc197563882)

[2.5. Tổng kết 65](#_Toc197563883)

[3. Hàm loss, hàm đo độ chính xác 66](#_Toc197563884)

[4. Huấn luyện mô hình 68](#_Toc197563885)

[4.1. Đánh giá phân bố đầu ra 70](#_Toc197563886)

[4.2. Đánh giá ma trận đầu ra và phân bố đầu ra so với nhãn 72](#_Toc197563887)

[5. Xây dựng ứng dụng web 74](#_Toc197563888)

[5.1. Xây dựng database 75](#_Toc197563889)

[5.2. Xây dựng kiến trúc micro service 77](#_Toc197563890)

[6. Xây dựng fontend bằng reactjs 86](#_Toc197563891)

[6.1. Tổng quan 86](#_Toc197563892)

[6.2. Cơ chế bảo mật token 87](#_Toc197563893)

[6.3. Cơ chế lưu ảnh và xây dựng server ảnh 87](#_Toc197563894)

[CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 89](#_Toc197563895)

[1. Kết luận 89](#_Toc197563896)

[1.1. Mô hình 89](#_Toc197563897)

[1.2. Ứng dụng web 89](#_Toc197563898)

[2. Hướng phát triển 90](#_Toc197563899)

[2.1. Mô hình 90](#_Toc197563900)

[2.2. Ứng dụng web 91](#_Toc197563901)

[DANH MỤC TÀI LIỆU THAM KHẢO 92](#_Toc197563902)

[1. Tài liệu tham khảo chính (dẫn đến các bài nghiên cứu hoặc báo cáo) 92](#_Toc197563903)

[1.1. Thông tin sự hiệu quả của hệ thống gợi ý 92](#_Toc197563904)

[1.2. Nguồn thông tin về phần Ứng dụng thực tế - Các công ty sử dụng CF 94](#_Toc197563905)

[1.3. Nguồn thông tin về phần Ứng dụng thực tế - Các công ty sử dụng CF-based Encode Tranformer 95](#_Toc197563906)

[2. Cố vấn hỗ trợ lý luận và phân tích dự án 96](#_Toc197563907)

[LIÊN KẾT NGOÀI 98](#_Toc197563908)

[1 . Nguồn lưu trữ dự án 98](#_Toc197563909)

[2. Nguồn dữ liệu dự án 98](#_Toc197563910)

[3. Tài liệu mô tả chức năng người dùng (FSD) 98](#_Toc197563911)

[4. Tài liệu mô tả database (DDD) 98](#_Toc197563912)

[5. Tài liệu mô tả api (API Documentation) 98](#_Toc197563913)

[6. Diagram database 98](#_Toc197563914)

# DANH MỤC ẢNH

[Hình II‑1Sơ đồ tổng quát so sánh các loại dữ liệu 12](#_Toc197563915)

[Hình II‑2 hình mô tả trực quan về cách Hashing Trick hoạt động 19](#_Toc197563916)

[Hình II‑3 Minh họa thêm về cách hoạt động của byte-level encoding 22](#_Toc197563917)

[Hình II‑4 Ảnh minh họa công thức và quy trình hoạt động 24](#_Toc197563918)

[Hình III‑1 Ảnh minh họa cho kiến trúc CF 38](#_Toc197563919)

[Hình III‑2 Ảnh minh họa kiến trúc CF-based Encode Tranformer 42](#_Toc197563920)

[Hình IV‑1 Sơ đồ phân bố dữ liệu 61](#_Toc197563921)

[Hình IV‑2 kiến trúc mô hình thực tế 66](#_Toc197563922)

[Hình IV‑3 Phân bổ loss của tập train và test trong quá trình train 68](#_Toc197563923)

[Hình IV‑4 Sự phân bổ đầu ra của mô hình 70](#_Toc197563924)

[Hình IV‑5 Ma trận phân bố đầu ra 72](#_Toc197563925)

[Hình IV‑6 Biểu đồ phân bố nhãn so với dự đoán 73](#_Toc197563926)

[Hình IV‑7 Kiến trúc microservice 77](#_Toc197563927)

# GIỚI THIỆU

## Giới thiệu đề tài

### Lý do chọn đề tài

Trong bối cảnh các nền tảng thương mại điện tử ngày càng phát triển, hệ thống đề xuất sản phẩm trở thành một công cụ thiết yếu để cá nhân hóa trải nghiệm người dùng, tăng tỷ lệ chuyển đổi và giữ chân khách hàng. Các mô hình truyền thống như Collaborative Filtering (CF) đã chứng minh được hiệu quả thực tiễn nhưng vẫn còn một số hạn chế, đặc biệt là khi dữ liệu người dùng phân tán, không đồng nhất hoặc khan hiếm.

Gần đây, các mô hình dựa trên kiến trúc Transformer – vốn nổi bật trong xử lý ngôn ngữ tự nhiên – đã được nghiên cứu và ứng dụng thành công vào nhiều bài toán recommendation nhờ khả năng học được các mối quan hệ phức tạp giữa người dùng và sản phẩm. Việc kết hợp CF với Transformer mở ra một hướng tiếp cận hiện đại và tiềm năng, tận dụng khả năng attention để biểu diễn đặc trưng người dùng/sản phẩm một cách sâu sắc và linh hoạt hơn.

Dự án được chọn vì những lý do sau:

* Tính thực tiễn cao, giúp giải quyết bài toán gợi ý sản phẩm trong thương mại điện tử.
* Ứng dụng các kỹ thuật hiện đại, đặc biệt là Transformer-based Encoder, giúp cải thiện độ chính xác của mô hình so với CF truyền thống.
* Mở rộng khả năng cá nhân hóa, bằng cách tích hợp thêm dữ liệu hành vi, lịch sử tìm kiếm hoặc đặc trưng nội dung sản phẩm.
* Tăng cường kỹ năng chuyên môn về xử lý dữ liệu, xây dựng mô hình deep learning, tối ưu hóa mô hình và đánh giá hiệu suất hệ thống.
* Thông qua việc triển khai mô hình Collaborative Filtering kết hợp Transformer, đề tài không chỉ mang tính ứng dụng thực tế cao mà còn thể hiện khả năng khai thác các xu hướng công nghệ mới vào giải quyết các bài toán AI trong thương mại điện tử.

### Ý tưởng

Dự án hướng tới xây dựng một hệ thống gợi ý sản phẩm thông minh, giúp cá nhân hóa trải nghiệm mua sắm của từng người dùng. Thay vì chỉ dựa vào lịch sử tương tác như các hệ thống truyền thống, hệ thống này kết hợp thông tin từ cả người dùng và sản phẩm để đưa ra các gợi ý chính xác và phù hợp hơn.

Ý tưởng cốt lõi là thu thập và biểu diễn một nhóm gồm 30 đặc trưng, bao gồm thông tin hành vi, sở thích của người dùng và đặc điểm sản phẩm. Các đặc trưng này sau đó được kết hợp lại thành từng cặp người dùng – sản phẩm, rồi đưa vào mô hình để học cách đánh giá mức độ phù hợp giữa hai bên. Việc biểu diễn riêng biệt từng loại đặc trưng và sau đó đưa vào quá trình học sâu giúp tối ưu hóa khả năng mô tả, từ đó nâng cao chất lượng gợi ý.

Điểm nổi bật của dự án:

* Ứng dụng cách tiếp cận hiện đại kết hợp giữa Collaborative Filtering và học sâu, giúp khắc phục hạn chế của các phương pháp CF truyền thống.
* Tận dụng đồng thời cả thông tin người dùng và sản phẩm, cho phép hệ thống hiểu rõ hơn về mối quan hệ tiềm ẩn giữa hai đối tượng.
* Thiết kế hướng mở, dễ dàng mở rộng thêm các đặc trưng mới hoặc tích hợp thêm các nguồn dữ liệu bổ sung như lịch sử tìm kiếm, hành vi tương tác gần đây.
* Đảm bảo khả năng cá nhân hóa tốt hơn, đề xuất sản phẩm sát với nhu cầu thực tế của từng cá nhân.

Bên cạnh phần mô hình gợi ý, dự án cũng sẽ xây dựng một trang web hoàn chỉnh, đóng vai trò như nền tảng thương mại điện tử mô phỏng. Trang web bao gồm các chức năng như:

* Giao diện người dùng hiển thị danh sách sản phẩm và các sản phẩm được đề xuất.
* Tính năng tìm kiếm, lọc, xem chi tiết sản phẩm và quản lý giỏ hàng.
* Hệ thống đăng nhập người dùng, theo dõi lịch sử tương tác và cập nhật hành vi mua sắm.
* Kết nối trực tiếp với mô hình đề xuất để hiển thị sản phẩm phù hợp theo thời gian thực.

Việc kết hợp giữa mô hình gợi ý và một hệ thống web hoàn chỉnh giúp kiểm chứng hiệu quả thực tế của mô hình trong môi trường mô phỏng, đồng thời mang đến trải nghiệm người dùng đầy đủ và mạch lạc.

### Mô tả đề tài, bài toán

Dự án tập trung vào việc xây dựng một hệ thống gợi ý sản phẩm cá nhân hóa, với mục tiêu dự đoán xác suất người dùng sẽ quan tâm hoặc xem một bộ phim cụ thể. Đây là một bài toán thuộc lĩnh vực Recommendation Systems, trong đó hệ thống cần học được hàm ánh xạ từ không gian đặc trưng kết hợp giữa người dùng và sản phẩm sang xác suất tương tác (interaction likelihood).

Bài toán được mô hình hóa theo hướng supervised learning, đầu vào là tập hợp các cặp người dùng – sản phẩm, mỗi cặp được biểu diễn thông qua việc kết hợp 16 đặc trưng định danh, hành vi và nội dung (bao gồm thông tin của người dùng và sản phẩm). Đầu ra là một nhãn nhị phân biểu thị khả năng người dùng sẽ tương tác hoặc mua sản phẩm đó.

Để học được biểu diễn tiềm ẩn (latent representation) chất lượng cao cho người dùng và sản phẩm, các đặc trưng đầu vào sẽ được ánh xạ qua các lớp embedding riêng biệt, trước khi được kết hợp và đưa vào một mô hình học sâu. Việc tách biệt embedding theo từng nhóm đặc trưng cho phép mô hình học được ngữ nghĩa riêng biệt của từng loại dữ liệu và tối ưu hóa khả năng biểu diễn mối quan hệ người dùng – sản phẩm.

So với các phương pháp Collaborative Filtering truyền thống vốn chủ yếu dựa vào ma trận tương tác, hướng tiếp cận này cho phép tận dụng hiệu quả các đặc trưng phi cấu trúc, xử lý tốt hơn trong trường hợp dữ liệu thưa (data sparsity) hoặc người dùng/sản phẩm mới (cold-start problem).

Bên cạnh phần mô hình học sâu, dự án cũng triển khai một hệ thống web hoàn chỉnh làm nền tảng mô phỏng thương mại điện tử, qua đó tích hợp và kiểm chứng khả năng hoạt động của hệ thống gợi ý trong môi trường thực tế.

## Tính thực tiễn của đề tài

### Tính thực tiễn và hiệu quả kinh tế của hệ thống gợi ý

Hệ thống gợi ý sản phẩm đã trở thành một phần không thể thiếu trong các nền tảng thương mại điện tử hiện đại. Chúng không chỉ giúp người dùng khám phá sản phẩm dễ dàng mà còn đóng vai trò quan trọng trong việc tăng doanh thu và cải thiện trải nghiệm mua sắm cá nhân hóa.​

Tăng doanh thu đáng kể: Theo nghiên cứu, các đề xuất sản phẩm chiếm trung bình 31% doanh thu của các trang thương mại điện tử. Đặc biệt, tại Amazon, hệ thống gợi ý đóng góp tới 35% doanh thu của công ty.​

Tăng tỷ lệ chuyển đổi và giá trị đơn hàng: Các hệ thống gợi ý giúp tăng tỷ lệ chuyển đổi lên đến 300%, đồng thời tăng giá trị đơn hàng trung bình (AOV) thông qua việc đề xuất các sản phẩm bổ sung phù hợp.​

Tăng giá trị vòng đời khách hàng (CLV): Việc cá nhân hóa trải nghiệm mua sắm thông qua hệ thống gợi ý đã giúp các nhà bán lẻ tăng 22% giá trị vòng đời khách hàng, nhờ vào việc xây dựng mối quan hệ bền vững và khuyến khích mua sắm lặp lại.​

Giảm tỷ lệ thoát trang và tăng thời gian duyệt web: Bằng cách cung cấp các đề xuất phù hợp, hệ thống gợi ý giúp người dùng ở lại trang web lâu hơn và khám phá nhiều sản phẩm hơn, từ đó giảm tỷ lệ thoát trang và tăng khả năng mua hàng.​

### Xu hướng tất yếu và sự cần thiết của hệ thống gợi ý

Trong bối cảnh cạnh tranh khốc liệt của thị trường thương mại điện tử, việc triển khai hệ thống gợi ý sản phẩm cá nhân hóa không chỉ là một lựa chọn mà đã trở thành một xu hướng bắt buộc để duy trì và phát triển kinh doanh.​

Sự bùng nổ dữ liệu: Người dùng ngày càng tương tác nhiều hơn trên các nền tảng mua sắm trực tuyến, tạo ra lượng dữ liệu lớn về hành vi, sở thích, và thói quen. Hệ thống đề xuất hiện đại sử dụng học sâu cho phép khai thác hiệu quả nguồn dữ liệu này để tạo ra các gợi ý sát thực tế hơn.​

Trải nghiệm cá nhân hóa là yếu tố cạnh tranh sống còn: Theo báo cáo của Accenture, 91% người tiêu dùng cho biết họ sẵn sàng mua hàng từ các thương hiệu ghi nhớ họ và đưa ra các gợi ý phù hợp với họ. Trong bối cảnh cạnh tranh cao, cá nhân hóa không chỉ là một tiện ích, mà là lợi thế cạnh tranh rõ ràng.​

Khả năng mở rộng và ứng dụng đa lĩnh vực: Các mô hình gợi ý hiện đại không chỉ giới hạn trong thương mại điện tử mà còn ứng dụng mạnh mẽ trong giáo dục, y tế, giải trí, ngân hàng,… Do đó, việc phát triển mô hình gợi ý không chỉ phục vụ mục đích học thuật mà còn có tiềm năng ứng dụng rộng lớn trong thực tế.​

Hiệu quả kinh tế và vận hành rõ rệt: So với các chiến dịch marketing đại trà, hệ thống gợi ý hướng đến đúng nhóm khách hàng mục tiêu, từ đó giảm chi phí tiếp thị không cần thiết. Một nghiên cứu của Boston Consulting Group (2021) cho thấy việc áp dụng cá nhân hóa có thể tăng gấp đôi ROI trong các chiến dịch bán hàng.​

Với những minh chứng và số liệu cụ thể trên, việc phát triển và triển khai hệ thống gợi ý sản phẩm cá nhân hóa bằng học sâu không chỉ mang lại lợi ích kinh tế rõ rệt mà còn đáp ứng xu hướng tất yếu trong lĩnh vực thương mại điện tử hiện đại.

### Lĩnh vực thời trang

#### Tăng trưởng doanh thu và hiệu quả kinh doanh

Tăng doanh thu lên đến 31%: Theo nghiên cứu từ Stylitics, các hệ thống gợi ý cá nhân hóa có thể chiếm tới 31% doanh thu của các trang thương mại điện tử thời trang. ​

Tăng tỷ lệ chuyển đổi lên đến 300%: Nghiên cứu từ McKinsey cho thấy các đề xuất sản phẩm có thể tăng tỷ lệ chuyển đổi lên đến 300%, giúp cải thiện hiệu quả kinh doanh. ​

Tăng giá trị đơn hàng trung bình (AOV): Theo báo cáo từ ViSenze, khách hàng tương tác với các đề xuất sản phẩm có thể tăng giá trị đơn hàng trung bình lên đến 5.5%. ​

#### Ứng dụng thực tiễn trong ngành thời trang

Marks & Spencer (M&S): M&S đã triển khai công nghệ AI để cung cấp tư vấn thời trang cá nhân hóa dựa trên hình dáng cơ thể và sở thích phong cách của người mua sắm. Hệ thống này đã thu hút 450.000 người dùng hoàn thành bài kiểm tra phong cách, giúp tăng doanh số bán hàng trực tuyến và cải thiện trải nghiệm khách hàng. ​

The Yes: Ứng dụng mua sắm AI-driven này sử dụng học máy để đề xuất quần áo, tương tự như Tinder, nhưng tinh chỉnh lựa chọn dựa trên sở thích người dùng. Ứng dụng này đã thu hút hơn 7 triệu điểm dữ liệu, giúp cá nhân hóa trải nghiệm mua sắm trực tuyến. ​

OneOff: Nền tảng tìm kiếm thời trang AI này cho phép người dùng tìm kiếm trang phục theo phong cách của người nổi tiếng như Hailey Bieber và Addison Rae. Nền tảng này sử dụng các mô hình ngôn ngữ lớn như ChatGPT và Google Gemini để cung cấp kết quả tìm kiếm, giúp người dùng khám phá sản phẩm thời trang một cách dễ dàng và nhanh chóng. ​

#### Xu hướng và tầm quan trọng của hệ thống gợi ý trong ngành thời trang

Yếu tố cạnh tranh sống còn: Theo báo cáo từ Accenture, 91% người tiêu dùng cho biết họ sẵn sàng mua hàng từ các thương hiệu ghi nhớ họ và đưa ra các gợi ý phù hợp với họ. Trong bối cảnh cạnh tranh cao, cá nhân hóa không chỉ là một tiện ích, mà là lợi thế cạnh tranh rõ ràng.​

Tăng trưởng thị trường toàn cầu: Thị trường hệ thống gợi ý toàn cầu được định giá 3,92 tỷ USD vào năm 2023 và dự kiến tăng trưởng với tỷ lệ CAGR là 36,3% từ năm 2024 đến 2030, cho thấy sự quan trọng ngày càng tăng của hệ thống gợi ý trong thương mại điện tử.

## Công nghệ sử dụng

### Phần Deep Learning:

TensorFlow (TF): Đây là thư viện mã nguồn mở mạnh mẽ được sử dụng để xây dựng và triển khai các mô hình học sâu. TensorFlow sẽ được sử dụng để xây dựng và huấn luyện các mô hình dự đoán, bao gồm các mô hình học sâu như Collaborative Filtering và các mô hình học sâu khác trong dự án.

Pandas: Là thư viện phổ biến để xử lý và phân tích dữ liệu, Pandas sẽ được sử dụng để làm việc với dữ liệu đầu vào, chuẩn hóa và tiền xử lý dữ liệu (như xử lý các tập tin CSV, dữ liệu dạng bảng, v.v.), giúp chuẩn bị dữ liệu cho các mô hình học sâu.

Scikit-learn (sklearn): Thư viện này cung cấp các công cụ để phân tích và mô hình hóa dữ liệu, với các thuật toán học máy như phân loại, hồi quy và clustering. Scikit-learn sẽ hỗ trợ trong các tác vụ như tiền xử lý dữ liệu, đánh giá mô hình và các công cụ tối ưu hóa.

NumPy: Thư viện quan trọng để tính toán khoa học trong Python, NumPy sẽ được sử dụng để xử lý các mảng dữ liệu số, giúp tối ưu hóa các phép toán ma trận và vector, phục vụ cho các mô hình học sâu.

### Phần Web Development:

Spring Boot: Là một framework phổ biến trong phát triển ứng dụng web với Java, Spring Boot sẽ được sử dụng để xây dựng backend của hệ thống, quản lý API và xử lý các yêu cầu từ người dùng. Nó giúp đơn giản hóa việc xây dựng các ứng dụng web mạnh mẽ và dễ bảo trì.

Flask: Là một framework micro của Python, Flask sẽ được sử dụng để xây dựng các API nhẹ nhàng, phục vụ việc xử lý dữ liệu và tạo ra các dịch vụ web nhỏ gọn. Flask có thể giúp kết nối giữa frontend và backend trong hệ thống, đặc biệt là khi cần triển khai các mô hình học máy hoặc Deep Learning qua API.

React: Là thư viện JavaScript phổ biến để xây dựng giao diện người dùng (UI), React sẽ được sử dụng để phát triển frontend của hệ thống. React giúp tạo ra các ứng dụng web tương tác và mượt mà, cung cấp trải nghiệm người dùng hiệu quả và dễ bảo trì. Nó sẽ giúp xây dựng các giao diện người dùng động và quản lý trạng thái ứng dụng.

## Kết quả mong muốn

Dự án hệ thống đề xuất sản phẩm này nhắm đến mục tiêu cung cấp một nền tảng mạnh mẽ và hiệu quả giúp cá nhân hóa trải nghiệm mua sắm cho người dùng, tối ưu hóa các lựa chọn sản phẩm dựa trên các mô hình học sâu và học máy tiên tiến. Kết quả mong muốn của dự án không chỉ đơn thuần là việc xây dựng một hệ thống đề xuất sản phẩm chính xác mà còn bao gồm một loạt các cải tiến về cả mặt trải nghiệm người dùng lẫn hiệu suất kinh doanh.

Đầu tiên, hệ thống sẽ mang lại một độ chính xác cao trong việc dự đoán sản phẩm mà người dùng có khả năng mua trong tương lai. Bằng cách sử dụng các mô hình học sâu như Collaborative Filtering và Transformer để xử lý dữ liệu người dùng và sản phẩm, hệ thống sẽ phân tích hành vi người dùng, lịch sử tìm kiếm, các yếu tố sở thích cá nhân và hành vi tương tác trước đó để tạo ra những đề xuất sản phẩm có khả năng khiến người dùng quyết định mua cao hơn.

Một trong những kết quả quan trọng là tăng trưởng doanh thu thông qua việc cải thiện tỷ lệ chuyển đổi (conversion rate) và giảm thiểu tỷ lệ bỏ giỏ hàng. Với khả năng dự đoán chính xác hơn về các sản phẩm mà người dùng quan tâm, hệ thống sẽ giúp người dùng dễ dàng tìm thấy những món hàng mà họ sẽ yêu thích ngay khi truy cập vào trang web. Điều này dẫn đến việc người dùng sẽ có khả năng đưa ra quyết định mua sắm nhanh chóng hơn, và các chiến lược marketing cũng có thể được tối ưu hóa để tập trung vào những sản phẩm tiềm năng.

Một kết quả mong muốn khác là cải thiện sự hài lòng của khách hàng. Khi người dùng nhận được các đề xuất sản phẩm phù hợp hơn với sở thích và nhu cầu thực sự của họ, họ sẽ cảm thấy trải nghiệm mua sắm của mình trở nên dễ dàng và thú vị hơn. Điều này không chỉ giúp tăng độ trung thành của người dùng mà còn thúc đẩy sự quay lại của họ trong tương lai. Các đánh giá và phản hồi từ khách hàng sẽ có xu hướng tích cực hơn, khi họ cảm nhận được rằng nền tảng này thực sự hiểu họ.

Hiệu quả tối ưu hóa dữ liệu cũng sẽ là một kết quả quan trọng. Việc thu thập và phân tích các đặc trưng dữ liệu liên quan đến người dùng và sản phẩm sẽ cho phép hệ thống hiểu sâu sắc hơn về hành vi người dùng và xu hướng thị trường. Từ đó, các quyết định về việc trưng bày sản phẩm, quảng cáo và chiến lược marketing có thể được đưa ra một cách chính xác và phù hợp hơn, giúp tiết kiệm chi phí và tăng hiệu quả.

Ngoài ra, một trong những kết quả lâu dài mà dự án mong muốn đạt được là xây dựng được một hệ sinh thái học máy có thể mở rộng. Khi dự án được triển khai và đạt được sự thành công nhất định, các mô hình và thuật toán được phát triển có thể được mở rộng và áp dụng cho các ngành hàng khác nhau trong lĩnh vực thương mại điện tử. Việc này giúp dự án không chỉ tạo ra giá trị trong ngành thời trang mà còn có tiềm năng áp dụng rộng rãi cho các lĩnh vực khác như điện tử, thực phẩm, hoặc thậm chí các sản phẩm dịch vụ khác.

Đặc biệt, tính linh hoạt và khả năng tự cải thiện của hệ thống cũng là một kết quả quan trọng. Hệ thống đề xuất sản phẩm sẽ được thiết kế sao cho có thể tự động cập nhật và cải thiện mô hình dựa trên các dữ liệu mới mà không cần sự can thiệp quá nhiều từ người dùng hoặc các kỹ sư. Điều này giúp hệ thống duy trì hiệu quả lâu dài trong bối cảnh thị trường thay đổi nhanh chóng và nhu cầu của người tiêu dùng liên tục thay đổi.

Cuối cùng, tính bền vững và khả năng thích ứng với sự thay đổi của thị trường sẽ là một yếu tố quyết định thành công của dự án trong dài hạn. Với việc ứng dụng các công nghệ hiện đại như Deep Learning và các phương pháp phân tích dữ liệu tiên tiến, hệ thống không chỉ có thể đáp ứng được yêu cầu hiện tại mà còn có thể phát triển và điều chỉnh để phù hợp với những thay đổi và thách thức trong tương lai.

Tóm lại, kết quả mong muốn của dự án không chỉ dừng lại ở việc xây dựng một hệ thống đề xuất sản phẩm đơn giản, mà còn là việc xây dựng một nền tảng giúp nâng cao trải nghiệm người dùng, tối ưu hóa quy trình bán hàng, tăng trưởng doanh thu, cải thiện sự hài lòng khách hàng và phát triển các chiến lược marketing hiệu quả. Tất cả những yếu tố này sẽ tạo ra một hệ thống bền vững, có khả năng thích ứng và đổi mới liên tục trong môi trường kinh doanh đầy biến động.

# CƠ SỞ LÝ LUẬN

## Xử lý ngôn ngữ tự nhiên

### Cơ sở

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) là lĩnh vực nghiên cứu và ứng dụng liên quan đến việc cho máy tính khả năng hiểu, phân tích, tạo ra và tương tác bằng ngôn ngữ tự nhiên của con người. NLP kết hợp kiến thức từ nhiều ngành, bao gồm khoa học máy tính, ngôn ngữ học, và trí tuệ nhân tạo (AI), nhằm thu hẹp khoảng cách giao tiếp giữa con người và máy móc.

Các bài toán phổ biến trong NLP bao gồm: phân loại văn bản, dịch máy, tóm tắt văn bản tự động, phân tích cảm xúc, trả lời câu hỏi, chatbot, sinh văn bản, và nhận diện thực thể có tên (NER).

### Một số điểm nổi bật riêng của NLP

Ngôn ngữ tự nhiên không tuân theo quy luật cứng như ảnh (hình học) hay âm thanh (tần số). Nó giàu ngữ nghĩa và mơ hồ, đòi hỏi máy phải "hiểu" theo cấp độ ngữ cảnh.

Vấn đề đa nghĩa (polysemy) và đồng nghĩa (synonymy) cực kỳ phổ biến, đòi hỏi các mô hình phải học sâu mối liên hệ giữa các từ trong ngữ cảnh cụ thể.

Cấu trúc văn bản có tính phân cấp (chữ → từ → câu → đoạn → bài), nên việc modeling hiệu quả cần xử lý cả các mối quan hệ ngắn hạn và dài hạn.

Sự phát triển của các mô hình Transformer (ví dụ: BERT, GPT) đã tạo ra bước nhảy vọt trong NLP, vì khả năng tự động học ngữ cảnh động thay vì dựa trên quy tắc thủ công.

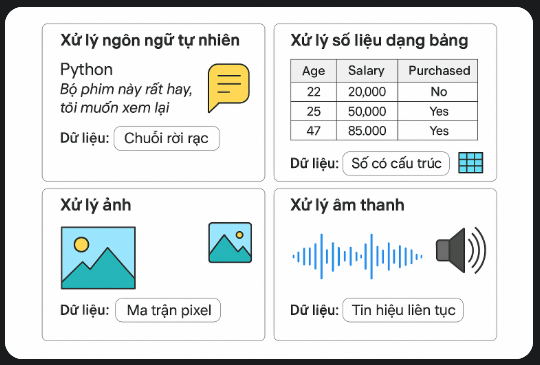
### Các điểm đặc biệt của NLP so với xử lý số, xử lý ảnh và xử lý âm thanh

Xử lý ngôn ngữ tự nhiên (NLP) là lĩnh vực chuyên biệt xử lý dữ liệu dưới dạng ngôn ngữ con người, như văn bản hoặc lời nói. Khác với xử lý số liệu dạng bảng – vốn làm việc với các con số có cấu trúc rõ ràng và định dạng chuẩn hóa – NLP phải đối mặt với dữ liệu dạng chuỗi rời rạc, không có cấu trúc cố định, đa dạng về ngữ nghĩa và cú pháp. Đây là thách thức lớn khi mô hình không chỉ cần "đọc" mà còn phải "hiểu" được ý nghĩa sâu xa trong ngôn ngữ tự nhiên.

So với xử lý ảnh, nơi dữ liệu đầu vào là các ma trận pixel với tính chất liên tục và có thể khai thác bằng các phép toán hình học hoặc biến đổi không gian, NLP lại đòi hỏi khả năng mô hình hóa các mối quan hệ ngữ nghĩa phức tạp giữa các từ, câu và đoạn văn, đồng thời phải xử lý hiện tượng đồng nghĩa, đa nghĩa, và ngữ cảnh.

Bên cạnh đó, khác với xử lý âm thanh, vốn xử lý tín hiệu liên tục theo thời gian như sóng âm hoặc tần số, NLP chủ yếu làm việc với chuỗi rời rạc các ký tự hoặc từ. Âm thanh thiên về phân tích đặc trưng vật lý (như phổ tần), trong khi NLP cần nắm bắt ý nghĩa ngữ nghĩa và ngữ pháp, vốn giàu tính trừu tượng hơn.

Chính vì vậy, NLP yêu cầu các kỹ thuật riêng biệt như tokenization, embedding, attention mechanism, và mô hình hóa ngữ cảnh đa chiều, khiến nó trở thành một lĩnh vực đầy thách thức và khác biệt so với xử lý số liệu, ảnh và âm thanh.



Hình II‑1Sơ đồ tổng quát so sánh các loại dữ liệu

## Các kiến trúc và bài toán nổi bật

### Mạng nơ-ron hồi tiếp (Recurrent Neural Network – RNN)

Cách hoạt động:

* RNN xử lý dữ liệu chuỗi bằng cách duy trì một trạng thái ẩn (hidden state) nối tiếp qua từng bước thời gian. Tại mỗi token đầu vào, trạng thái ẩn được cập nhật dựa trên đầu vào hiện tại và trạng thái ẩn trước đó, cho phép “nhớ” thông tin từ các bước trước.

Đặc trưng:

* Xử lý tốt phụ thuộc tuần tự ngắn hạn (short-term dependencies).
* Kịp thời nắm bắt mối quan hệ theo thứ tự xuất hiện của từ.
* Dễ bị vanishing/exploding gradient khi chuỗi quá dài, gây khó khăn cho việc học phụ thuộc dài hạn.

Phù hợp với:

* Bài toán phân loại chuỗi ngắn (ví dụ: sentiment analysis với câu ngắn).
* Sequence labeling trên các đoạn văn ngắn (ví dụ: POS tagging, NER) khi ngữ cảnh không quá xa.

### LSTM và GRU (Long Short-Term Memory & Gated Recurrent Unit)

Cách hoạt động:

* LSTM và GRU là biến thể của RNN được thiết kế để giải quyết vấn đề vanishing gradient. Chúng sử dụng cơ chế “cổng” (gates) để quyết định thông tin nào cần giữ lại, bỏ qua hay đưa vào trạng thái ẩn mới. LSTM có ba cổng chính (input, forget, output), trong khi GRU gộp một số cổng lại để đơn giản hơn.

Đặc trưng:

* Khả năng học phụ thuộc dài hạn (long-term dependencies) tốt hơn RNN cơ bản.
* Cơ chế cổng giúp mô hình linh hoạt hơn trong việc điều tiết thông tin.
* Tốn kém tính toán và thời gian huấn luyện so với RNN thuần.

Phù hợp với:

* Bài toán yêu cầu hiểu ngữ cảnh xa (ví dụ: tóm tắt văn bản, dịch máy với câu dài).
* Question answering, khi cần dựa vào ngữ cảnh dài để trích xuất câu trả lời.

### Mạng CNN cho văn bản (Convolutional Neural Network – CNN)

Cách hoạt động:

* CNN áp dụng các bộ lọc (filters) trượt trên chuỗi từ hoặc embedding sequence để trích xuất các đặc trưng cục bộ (local n-gram features). Sau đó dùng pooling để chọn các đặc trưng nổi bật nhất.

Đặc trưng:

* Khả năng học đặc trưng cục bộ hiệu quả (ví dụ: cụm từ mang ý nghĩa).
* Song song hoá tính toán tốt, huấn luyện nhanh.
* Hạn chế trong việc nắm bắt phụ thuộc dài hạn quá xa.

Phù hợp với:

* Phân loại văn bản, sentiment analysis, vì các cụm từ cục bộ thường quyết định nhãn.
* Tasks cần trích xuất feature nhanh từ tài liệu ngắn hoặc trung bình.

### Transformer

Cách hoạt động:

* Transformer sử dụng cơ chế self-attention để mô hình hóa mối quan hệ giữa mọi cặp token trong chuỗi đầu vào, không phụ thuộc tuần tự tính toán theo thời gian. Cấu trúc gồm các lớp encoder/decoder xếp chồng, mỗi lớp có multi-head attention và feed-forward network.

Đặc trưng:

* Nắm bắt phụ thuộc ngắn hạn và dài hạn đồng thời nhờ self-attention.
* Dễ dàng song song hoá (parallelization) trong huấn luyện, tiết kiệm thời gian.
* Kích thước mô hình lớn, đòi hỏi tài nguyên tính toán và dữ liệu huấn luyện nhiều.

Phù hợp với:

* Hầu hết các bài toán NLP hiện đại: dịch máy, tóm tắt, question answering, text generation.
* Các ứng dụng đòi hỏi hiểu ngữ cảnh toàn diện và quan hệ token-to-token.

### BERT (Bidirectional Encoder Representations from Transformers)

Cách hoạt động:

* BERT là mô hình Transformer chỉ gồm phần encoder và được huấn luyện trước (pre-train) với hai nhiệm vụ: Masked Language Modeling (MLM) – dự đoán từ bị che – và Next Sentence Prediction (NSP). Việc học bidirectional attention giúp BERT hiểu ngữ cảnh từ hai phía trái và phải.

Đặc trưng:

* Khả năng biểu diễn ngữ cảnh hai chiều (bidirectional) mạnh mẽ.
* Dễ fine-tune cho nhiều tasks downstream với ít bước huấn luyện.
* Kích thước lớn, cần tài nguyên để triển khai real-time.

Phù hợp với:

* Sequence labeling (NER, POS tagging), text classification, question answering (ví dụ: SQuAD).
* Mọi bài toán cần hiểu ngữ cảnh sâu từ cả hai phía xung quanh token.

GPT (Generative Pre-trained Transformer)

Cách hoạt động:

* GPT là mô hình Transformer chỉ gồm phần decoder, huấn luyện trước theo hình thức autoregressive: dự đoán token tiếp theo trong chuỗi. Nhờ vậy GPT mạnh về sinh văn bản liên tục và mạch lạc.

Đặc trưng:

* Xuất sắc trong text generation, completion, dialogue.
* Khả năng zero-shot, few-shot learning khi prompt phù hợp.
* Ít tối ưu hơn BERT cho các task classification hoặc sequence labeling vì tính unidirectional.

Phù hợp với:

* Sinh văn bản (text generation), chatbot, viết sáng tạo, completion tasks.
* Ứng dụng cần tạo nội dung mới hoặc trả lời câu hỏi theo phong cách hội thoại.

### Các kiến trúc khác và mô hình lai (Hybrid)

Encoder–Decoder Transformer (ví dụ: T5, BART) kết hợp hai thành phần encoder và decoder, hỗ trợ cả generation và understanding. Thích hợp cho tóm tắt abstractive, translation, question answering generative.

Graph Neural Networks (GNN) áp dụng với NLP để mô hình hóa mối quan hệ phức tạp giữa thực thể hoặc các khối thông tin, phù hợp cho knowledge graph completion, relation extraction.

### Kết luận

Mỗi kiến trúc đều đặt ra một bộ đổi trade-off giữa khả năng nắm bắt phụ thuộc dài hạn, hiệu quả tính toán và phù hợp cho từng bài toán cụ thể. Việc lựa chọn kiến trúc phù hợp phụ thuộc vào yêu cầu của ứng dụng, quy mô dữ liệu và tài nguyên tính toán sẵn có.

## Xử lý dữ liệu cho các bài toán ngôn ngữ

### Mã hóa chữ thành số trong xử lý ngôn ngữ tự nhiên: encryption và hàm số học mã hóa

Trong các bài toán xử lý ngôn ngữ tự nhiên (NLP), việc chuyển đổi (mã hóa) các đơn vị ngôn ngữ như chữ cái, từ, câu thành số là một bước cơ bản để có thể thực hiện các phép toán và thuật toán trên máy tính.

Ngoài các kỹ thuật học máy hiện đại, một hướng tiếp cận cổ điển nhưng rất có giá trị về mặt lý thuyết là sử dụng Encryption (mã hóa) dựa trên hàm số học, trong đó mỗi đơn vị ngôn ngữ được ánh xạ qua một hàm toán học có cấu trúc chặt chẽ, có thể đảo ngược.

Phương pháp này rất mạnh để:

* Kiểm soát hoàn toàn quá trình mã hóa và giải mã.
* Đảm bảo tính nhất quán, có thể phục hồi dữ liệu.
* Cho phép mô hình hóa ngôn ngữ dưới dạng số học đơn giản mà không cần học.

### hashing(hashing trick)

Trong bài toán xử lý ngôn ngữ tự nhiên, Hashing Trick là kỹ thuật mã hóa từ hoặc cụm từ thành số bằng cách sử dụng một hàm băm để ánh xạ trực tiếp mỗi từ vào một vị trí trong vector đặc trưng có kích thước cố định. Thay vì phải xây dựng một từ điển mapping từ → số như cách truyền thống, Hashing Trick giúp tính toán trực tiếp chỉ số bằng phép lấy phần dư của giá trị băm (modulo D), trong đó D là kích thước vector.

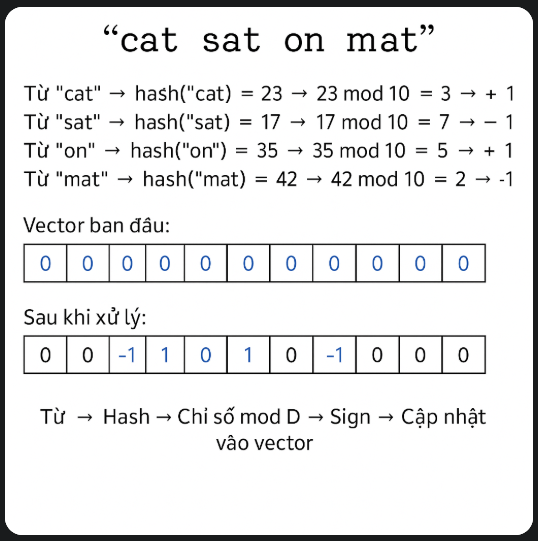
Cách hoạt động của Hashing Trick như sau: khi gặp một từ, ta tính giá trị băm của từ đó, sau đó lấy phần dư của giá trị băm chia cho D để xác định vị trí trong vector. Ngoài ra, ta còn dùng thêm một hàm băm phụ để xác định dấu (+1 hoặc -1) nhằm giảm tác động tiêu cực khi nhiều từ cùng rơi vào một vị trí (gọi là collision). Mỗi lần gặp một từ, ta tăng hoặc giảm giá trị tại vị trí tương ứng trong vector tùy theo dấu đã tính.

Kỹ thuật này tối ưu ở chỗ nó không cần lưu từ điển, tiết kiệm bộ nhớ rất lớn đặc biệt khi tập từ vựng có thể lên tới hàng triệu từ. Tốc độ xử lý nhanh vì việc băm và cập nhật vector chỉ gồm các phép toán cơ bản. Mặc dù chấp nhận khả năng va chạm băm, Hashing Trick vẫn hoạt động tốt nếu chọn D đủ lớn so với số lượng từ thực tế. Đây là một sự đánh đổi nhỏ lấy hiệu quả lớn trong bài toán scale-up.

Về thuật toán tính toán cụ thể, giả sử có một văn bản gồm nhiều từ. Ta khởi tạo một vector x có D phần tử bằng 0. Với mỗi từ w, ta tính chỉ số i bằng hash(w) mod D, sau đó xác định dấu sign(w) bằng cách băm phụ từ w và kiểm tra nó âm hay dương. Cuối cùng, ta cộng sign(w) vào x[i]. Sau khi duyệt hết văn bản, vector x chính là đặc trưng số hóa biểu diễn văn bản ban đầu.

Để minh họa, giả sử câu "cat sat on mat" và chọn D = 10. Khi băm từng từ, giả định ta thu được: từ "cat" băm ra 23, modulo 10 bằng 3 và có dấu +1; từ "sat" băm ra 17, modulo 10 bằng 7 và có dấu -1; từ "on" băm ra 35, modulo 10 bằng 5 và có dấu +1; từ "mat" băm ra 42, modulo 10 bằng 2 và có dấu -1. Ban đầu vector đặc trưng toàn số 0. Sau khi lần lượt cập nhật theo từng từ, vector kết quả là [0, 0, -1, 1, 0, 1, 0, -1, 0, 0]. Như vậy toàn bộ quá trình chuyển văn bản thành vector chỉ gồm băm, lấy phần dư, và cộng dồn dấu.

Dưới đây là hình mô tả trực quan về cách Hashing Trick hoạt động:



Hình II‑2 hình mô tả trực quan về cách Hashing Trick hoạt động

### Character level encoding

Character-level Encoding là một kỹ thuật trong xử lý ngôn ngữ tự nhiên (NLP) nhằm mã hóa thông tin từ từng ký tự thay vì từng từ. Điều này giúp mô hình dễ dàng nhận diện và xử lý thông tin chi tiết hơn trong ngữ cảnh, đặc biệt là khi đối phó với từ mới, từ ngữ lạ hoặc các ngôn ngữ ít được sử dụng trong dữ liệu huấn luyện.

Cách hoạt động của character-level encoding bắt đầu từ việc biến từng ký tự trong văn bản thành các chỉ số số học. Một cách đơn giản nhất là xây dựng một bảng ánh xạ từ các ký tự trong tập ký tự (alphabet) sang các chỉ số số nguyên. Ví dụ, có thể ánh xạ mỗi ký tự vào một số nguyên duy nhất, như: a -> 0, b -> 1, c -> 2, ..., z -> 25.

Kỹ thuật này cực kỳ hữu ích khi dữ liệu có chứa các từ chưa từng thấy trong tập huấn luyện (out-of-vocabulary words) hoặc khi cần xử lý các ngôn ngữ không có từ điển chuẩn. Một ưu điểm lớn của việc mã hóa ký tự là mô hình có thể học được các đặc điểm ngữ nghĩa từ cấu trúc con của từ, thay vì chỉ nhìn vào từng từ riêng biệt.

Thuật toán tính toán của character-level encoding có thể được mô tả qua các bước sau:

Tiền xử lý văn bản: Văn bản được tách thành các ký tự riêng biệt. Ví dụ, câu "hello" sẽ được tách thành ['h', 'e', 'l', 'l', 'o'].

Ánh xạ ký tự thành số: Mỗi ký tự trong câu được ánh xạ vào một chỉ số số nguyên. Ví dụ, nếu ta sử dụng bảng ánh xạ như trên, "hello" có thể trở thành [7, 4, 11, 11, 14].

Tạo vector đặc trưng: Mỗi chỉ số số nguyên sau đó có thể được chuyển thành vector one-hot hoặc embedding tùy thuộc vào mục đích sử dụng mô hình. Vector one-hot sẽ có kích thước bằng với số lượng ký tự trong bảng (ví dụ, 26 cho bảng chữ cái Latin), với giá trị 1 tại chỉ số tương ứng và các phần tử còn lại là 0. Còn nếu sử dụng embedding, mỗi chỉ số số nguyên sẽ được ánh xạ tới một vector có chiều dài cố định, ví dụ như 50, 100, hoặc 300.

Để minh họa, giả sử ta có câu "hi":

Ánh xạ ký tự "h" → chỉ số 7 và "i" → chỉ số 8.

Nếu sử dụng one-hot encoding, "hi" sẽ thành [0, 0, 0, 0, 0, 0, 1, 0, 0, ..., 0] cho "h" và [0, 0, 0, 0, 0, 0, 0, 1, 0, ..., 0] cho "i".

Nếu sử dụng embedding, "h" có thể thành một vector như [0.1, 0.3, -0.2, ...] và "i" có thể thành [0.05, -0.1, 0.2, ...].

Ưu điểm của Character-level Encoding:

* Chịu được sự không đồng nhất trong ngôn ngữ: Có thể mã hóa từ chưa từng gặp (out-of-vocabulary) mà không cần sửa đổi bất kỳ phần nào của mô hình.
* Hiểu được cấu trúc ngữ nghĩa con của từ: Các mô hình có thể học các mẫu như tiền tố, hậu tố, hoặc cấu trúc phổ biến trong từ ngữ.
* Linh hoạt và dễ dàng áp dụng: Không cần phải xây dựng một từ điển lớn và có thể làm việc với nhiều loại ngôn ngữ khác nhau mà không cần điều chỉnh gì nhiều.

Nhược điểm:

* Chi phí tính toán cao hơn: Vì phải xử lý nhiều ký tự thay vì từ, độ dài của chuỗi văn bản có thể kéo dài và yêu cầu tài nguyên tính toán lớn hơn.
* Không phải lúc nào cũng hiệu quả: Đặc biệt là khi từ vựng trong văn bản rất lớn và có sự phụ thuộc mạnh vào nghĩa của từ cụ thể.

### Byte level encoding

Byte-level encoding là một phương pháp mã hóa trong xử lý ngôn ngữ tự nhiên, nơi mỗi ký tự (bao gồm chữ cái, ký tự đặc biệt và dấu câu) được chuyển đổi thành các giá trị số tương ứng. Thực chất, phương pháp này thường sử dụng bảng mã hóa như ASCII hoặc UTF-8, trong đó mỗi ký tự được đại diện bởi một dãy byte cụ thể.

Mục tiêu chính của byte-level encoding là chuyển đổi thông tin từ dạng văn bản thành dạng số để máy tính có thể hiểu và xử lý được. Một trong những lợi ích của phương pháp này là tính đơn giản và hiệu quả. Mỗi ký tự hoặc chuỗi ký tự có thể được biểu diễn thông qua một số byte nhất định, từ đó giúp hệ thống có thể xử lý văn bản một cách nhanh chóng mà không cần phải quan tâm đến ngữ nghĩa của từ ngữ, mà chỉ tập trung vào cách thức biểu diễn của nó.

Kỹ thuật tính toán trong byte-level encoding chủ yếu dựa trên việc chuyển đổi trực tiếp các ký tự thành dãy số. Ví dụ, khi sử dụng bảng mã ASCII, mỗi ký tự có thể được biểu diễn dưới dạng một số nguyên trong phạm vi 0-255. Ví dụ, ký tự 'A' trong mã ASCII sẽ có giá trị 65, và ký tự 'a' sẽ có giá trị 97. Điều này tạo ra một sự chuyển đổi trực tiếp từ ký tự thành số mà không cần bất kỳ quá trình xử lý phức tạp nào.

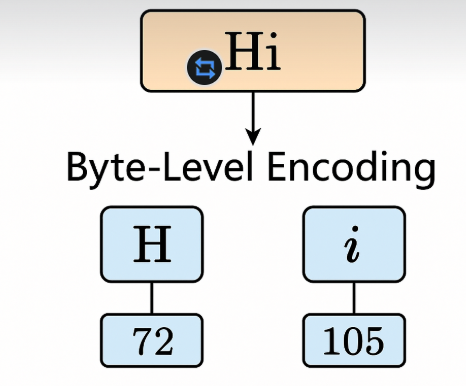
Trong UTF-8, mỗi ký tự có thể được mã hóa bằng một hoặc nhiều byte tùy thuộc vào độ dài của ký tự đó. Những ký tự đơn giản như các ký tự Latin có thể được mã hóa bằng một byte duy nhất, trong khi các ký tự từ các ngôn ngữ phức tạp hơn như tiếng Trung có thể yêu cầu nhiều byte để biểu diễn.

Ví dụ tay cho byte-level encoding: Giả sử bạn có chuỗi văn bản "Hi", để mã hóa nó theo byte-level encoding với bảng ASCII:

'H' sẽ được mã hóa thành 72.

'i' sẽ được mã hóa thành 105. Vậy chuỗi "Hi" sẽ được mã hóa thành hai byte [72, 105].

Sự tối ưu của byte-level encoding nằm ở khả năng sử dụng các bảng mã hóa đã được chuẩn hóa, giúp giảm bớt độ phức tạp trong việc biểu diễn các ký tự và đảm bảo tính tương thích giữa các hệ thống khác nhau. Tuy nhiên, điểm yếu của phương pháp này là nó không thể nắm bắt các mối quan hệ ngữ nghĩa hoặc ngữ cảnh giữa các từ trong câu, vì vậy nó thường không được sử dụng độc lập trong các mô hình phức tạp như các mạng nơ-ron trong xử lý ngôn ngữ tự nhiên.



Hình II‑3 Minh họa thêm về cách hoạt động của byte-level encoding

### Tokenize và Vocabulary Mapping

Trong bài toán xử lý ngôn ngữ tự nhiên, Tokenize và Vocabulary Mapping là hai bước nền tảng để biến văn bản thành các dãy số mà máy tính có thể xử lý.

Tokenize là quá trình tách chuỗi văn bản thành các đơn vị nhỏ hơn gọi là token. Token có thể là từ, subword hoặc thậm chí là từng ký tự, tùy vào cách thiết kế. Mục tiêu chính của bước tokenize là phân rã ngôn ngữ tự nhiên thành các thành phần mà mô hình học máy có thể hiểu và học được. Sau khi văn bản đã được phân tách thành các token, bước tiếp theo là Vocabulary Mapping, nghĩa là ánh xạ mỗi token thành một số nguyên duy nhất dựa trên một từ điển đã định nghĩa trước.

Về kỹ thuật tính toán, Tokenize đơn giản là một phép tách chuỗi dựa trên dấu cách, dấu câu hoặc các nguyên tắc đặc thù. Các token thu được sau bước này sẽ được ánh xạ vào từ điển (vocabulary), trong đó mỗi token có một chỉ số cố định. Từ điển có thể được xây dựng thủ công hoặc tự động dựa trên tập dữ liệu huấn luyện bằng cách duyệt qua toàn bộ tập văn bản, đếm tần suất xuất hiện và chọn ra các token phổ biến nhất.

Giả sử ta có câu "I love cats", việc tokenize sẽ tách ra thành ba token: ["I", "love", "cats"]. Tiếp theo, nếu vocabulary đã định nghĩa ánh xạ là {"I":1, "love":2, "cats":3}, thì câu này sẽ được chuyển thành dãy số [1, 2, 3].

Một điểm then chốt trong sự tối ưu của kỹ thuật này là nó cực kỳ đơn giản và nhanh, vì chỉ cần một lần tra cứu từ điển là ánh xạ xong, với độ phức tạp tính toán trung bình là O(1) cho mỗi token nếu từ điển được triển khai dưới dạng bảng băm (hashmap). Ngoài ra, vì mỗi token được gán một chỉ số duy nhất, nên việc truyền vào mô hình học máy rất thuận lợi và giảm thiểu bộ nhớ cần thiết so với việc lưu trữ các chuỗi dài.

Tuy nhiên, cách tiếp cận này cũng có một số bất lợi, ví dụ như từ chưa từng gặp (out-of-vocabulary, OOV) sẽ không có ánh xạ, nên thường người ta phải dự phòng thêm token đặc biệt như <UNK> cho các trường hợp này.

Công thức biểu diễn đơn giản quá trình tokenize và vocabulary mapping có thể viết như sau:

* tokens=Tokenize(s)
* indices=[Vocab(t) for t in tokens]

Trong đó:

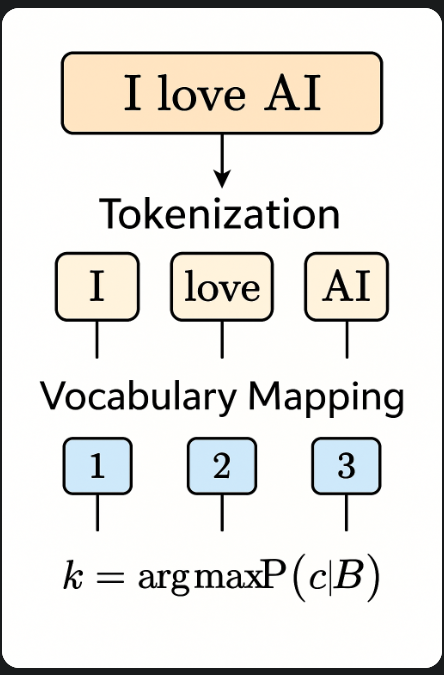
* s là chuỗi văn bản gốc
* Tokenize(s) là hàm tách chuỗi thành token
* Vocab(t) là hàm ánh xạ mỗi token
* t sang chỉ số trong từ điển

Ví dụ tay cho rõ hơn:

Chuỗi: "hello world" Tokenize thành ["hello", "world"]

Vocabulary: {"hello":5, "world":8}

Mapping ra dãy số [5,8]



Hình II‑4 Ảnh minh họa công thức và quy trình hoạt động

### One-hot embedding

One-hot embedding là bước tiếp nối trực tiếp sau Tokenize và Vocabulary Mapping trong quá trình mã hóa chữ thành số trong xử lý ngôn ngữ tự nhiên.

Sau khi văn bản được tách thành các token, mỗi token được ánh xạ thành một chỉ số duy nhất trong vocabulary, ta có thể biểu diễn mỗi token đó bằng một vector có chiều dài đúng bằng kích thước của toàn bộ từ điển. Vector này chứa toàn số 0, chỉ duy nhất vị trí tương ứng với chỉ số của token đó được gán giá trị 1. Đây chính là kỹ thuật one-hot embedding.

Về kỹ thuật tính toán, nếu một câu đã được mã hóa thành các chỉ số thông qua Vocabulary Mapping, thì với mỗi chỉ số đó ta tạo ra một vector 𝑣 ∈ 𝑅 𝑁 v∈R N , với 𝑁 N là kích thước từ điển, sao cho:

Vi ​ ={ 1​ nếu i = index của token, 0 nếu i != index của token} ​

Trong thực tế, thay vì thực sự lưu trữ một ma trận toàn 0 và 1 như vậy (vì nó rất thưa thớt), ta chỉ cần lưu chỉ số index và xây dựng phép biến đổi tại thời điểm cần sử dụng.

Ví dụ tay:

Giả sử ta có câu "I love AI", đã được tokenize thành ["I", "love", "AI"], và ánh xạ vocabulary là {"I":0, "love":1, "AI":2},

với kích thước từ điển là 3.

"I" -> chỉ số 0 -> vector [1,0,0]

"love" -> chỉ số 1 -> vector [0,1,0]

"AI" -> chỉ số 2 -> vector [0,0,1]

Vậy toàn bộ câu sẽ được mã hóa thành một ma trận 3×3, mỗi dòng là một vector one-hot tương ứng với một token.

Về sự tối ưu, one-hot embedding có ưu điểm là cực kỳ đơn giản, dễ cài đặt và không có tham số cần học, rất phù hợp cho các bước tiền xử lý nhanh chóng. Tuy nhiên, nhược điểm lớn nhất là tốn bộ nhớ do vector thưa thớt và không chứa thông tin về mối quan hệ giữa các từ (mỗi vector hoàn toàn trực giao). Để cải thiện vấn đề này trong các mô hình thực tế, người ta thường chuyển sang các embedding học được, nhưng ở đây ta chỉ tập trung vào one-hot.

Công thức gọn:

Giả sử:

* s là chuỗi ban đầu
* T=Tokenize(s) là danh sách token
* I=Mapping(T) là danh sách index

V là tập từ điển có kích thước N Thì biểu diễn one-hot của token t là:

* onehot(t)=[0,…,0,1,0,…,0](1 tại vị trí I(t))

Một công thức dạng ma trận cho cả câu:

* O=onehot\_matrix(I,N)

Với O là ma trận len ( 𝐼 ) × 𝑁 len(I)×N, mỗi dòng là vector one-hot.

### Word Embeding cbow, skip -gram, word2vec

Đầu tiên, khi thực hiện xử lý ngôn ngữ tự nhiên, ta cần biến chuỗi văn bản thành dạng số. Việc đầu tiên là Tokenize chuỗi thành các từ đơn lẻ và xây dựng Vocabulary Mapping. Ví dụ, từ câu "I love deep learning", ta thu được danh sách tokens ["I", "love", "deep", "learning"] và ánh xạ từ sang chỉ số như "I" → 0, "love" → 1, "deep" → 2, "learning" → 3. Sau đó, ta tiến tới bài toán nhúng từ (word embedding) bằng mô hình Word2Vec, với hai cách tiếp cận chính là CBOW và Skip-gram.

Trong CBOW (Continuous Bag of Words), mục tiêu là dự đoán từ trung tâm dựa trên các từ ngữ cảnh xung quanh nó. Công thức xác suất cần tối ưu là:

P(wt​∣wt−m​,…,wt−1​,wt+1​,…,wt+m​)

với:

wt ​ là từ trung tâm tại vị trí t, và m là kích thước cửa sổ ngữ cảnh. Về mặt tính toán, ta lấy các từ ngữ cảnh, ánh xạ chúng qua bảng nhúng thành các vector, sau đó tính trung bình các vector này để dự đoán từ trung tâm. Kỹ thuật này giúp mô hình học được đại diện cho từ sao cho những từ thường xuất hiện trong cùng ngữ cảnh sẽ có vector gần nhau trong không gian.

Ví dụ tay: với câu "I love deep learning", nếu chọn "love" làm từ trung tâm và cửa sổ ngữ cảnh size 1, ngữ cảnh là ["I", "deep"]. Các bước cụ thể:

"I" và "deep" được tra chỉ số từ vocabulary (0 và 2), sau đó ánh xạ thành hai vector embedding (ví dụ 100 chiều).

Trung bình hai vector đó → vector ngữ cảnh.

Vector ngữ cảnh được nhân ma trận trọng số (hoặc mạng nơron đơn giản) để dự đoán từ "love" (index 1).

Ngược lại, Skip-gram đảo hướng bài toán: từ từ trung tâm, ta dự đoán các từ ngữ cảnh. Công thức tối ưu là:

P(wt−m​,…,wt−1​,wt+1​,…,wt+m​∣wt​)

Ở Skip-gram, ta lấy từ trung tâm, ánh xạ nó qua bảng nhúng thành vector, rồi dùng vector đó để lần lượt dự đoán từng từ ngữ cảnh. Ví dụ, vẫn với "love" là từ trung tâm, mô hình sẽ học cách từ vector "love" có thể tái tạo lại "I" và "deep".

Trong Word2Vec, cả CBOW và Skip-gram đều dùng cùng một ý tưởng: ánh xạ từ các token vào một không gian vector ẩn. Về bản chất, kiến trúc Word2Vec chỉ là một mạng nơron đơn giản gồm một lớp ẩn linearsize rất nhỏ (ví dụ 100-300 chiều), và nhiệm vụ huấn luyện là tối ưu các vector embedding sao cho mô hình dự đoán đúng nhất. Để giảm chi phí tính toán, Word2Vec thường áp dụng kỹ thuật Negative Sampling, tức là chỉ cập nhật một phần nhỏ các vector từ mỗi lần dự đoán thay vì toàn bộ từ điển.

Kỹ thuật tính toán cụ thể là: mỗi token đầu tiên được ánh xạ thành một vector one-hot (từ bước tokenize và mapping ở trên), sau đó nhân với ma trận trọng số để lấy vector nhúng. Công thức đơn giản:

Embedding(wi​)=W⋅OneHot(wi​)

với W là ma trận embedding kích thước N×d, trong đó N là số lượng từ trong từ điển, d là số chiều embedding. Vì OneHot(w i) chỉ có một phần tử bằng 1, phép nhân thực tế chỉ là thao tác "lấy dòng thứ i" trong ma trận W.

Để mô tả quá trình một cách trực quan:

CBOW: (Nhiều từ ngữ cảnh) → (Vector trung bình) → (Dự đoán từ trung tâm)

Skip-gram: (Từ trung tâm) → (Vector nhúng) → (Dự đoán từng từ ngữ cảnh)

Cả hai hướng này đều khiến các vector embedding sau huấn luyện chứa đầy thông tin về ngữ nghĩa: các từ có ngữ nghĩa tương tự sẽ có vector gần nhau trong không gian vector.

Tóm tắt lại là tạo một mạng với câu trúc dạng đầu vào là một vector onehot chứa ngữ cảnh và đầu ra là một vector onehot chứa từ mục tiêu sau đó đưa vào một mạng word2vec bản chất mạng này sẽ có dạng input – ebeding(lớp Embeding nhúng) – dense – softmax cho đầu ra, sau khi huấn luyện ta sẽ có một mạng ánh xạ mối quan hệ ngữ cảnh – mục tiêu và khi cần dùng ta sẽ lấy lớp Embeding đã huấn luyện đó, nó là lớp Embeding với trọng số đã tối ưu hóa trên một bộ dữ liệu lớn và truyền mỗi một từ đã được ánh xạ trên thư viện ta sẽ có vector của từ đó, vector này đã được huấn luyện với cặp ngữ cảnh – mục tiêu nên sẽ phản ánh chính xác nhất nghĩa của một từ, sau đó ta lại dùng vector này để đưa vào các mô hình cần dùng. Và trong dự án này sẽ không dùng các thuật toán này mà dùng lớp Embeding để nhúng trực tiếp và training lại luôn lớp Embeding đó, lý do là do đặc trưng không phải toand chữ nên không thể dùng theo cơ chế anhd xạ ngữ cảnh – mục tiêu như vậy được.

## Các kiến trúc và thuật toán nền tảng

Trong xử lý ngôn ngữ tự nhiên (NLP), những kiến trúc nền tảng đời đầu đóng vai trò cực kỳ quan trọng trong việc mở đường cho các hệ thống hiện đại sau này. Các kiến trúc như RNN, LSTM, Bi-LSTM, và Seq2Seq lần lượt ra đời để giải quyết những hạn chế của nhau và dần dần hoàn thiện khả năng hiểu và sinh ngôn ngữ tự nhiên của máy.

RNN (Recurrent Neural Network) là kiến trúc đầu tiên chuyên biệt cho dữ liệu tuần tự. RNN xử lý dữ liệu bằng cách lặp qua từng phần tử một, và tại mỗi bước, nó vừa nhận dữ liệu mới vừa nhớ trạng thái quá khứ thông qua một biến ẩn (hidden state). Quá trình tính toán tại mỗi bước có dạng: lấy đầu vào hiện tại và trạng thái ẩn trước đó, cộng dồn thông tin lại rồi truyền tiếp. Điều này giúp RNN có khả năng ghi nhớ ngữ cảnh trước trong một chuỗi. Tuy nhiên, RNN gặp phải vấn đề rất lớn là vấn đề biến mất gradient (vanishing gradient problem). Khi chuỗi dữ liệu quá dài, những thông tin cũ bị suy giảm dần qua từng bước truyền dẫn, khiến cho RNN gần như không thể học được các quan hệ dài hạn trong văn bản.

LSTM (Long Short-Term Memory) ra đời để khắc phục vấn đề này. LSTM giới thiệu cơ chế cổng (gates): cổng quên, cổng cập nhật, cổng đầu ra. Các cổng này điều khiển dòng thông tin đi qua mạng, giúp mạng lựa chọn cái gì nên nhớ, cái gì nên quên. Về kỹ thuật, LSTM bổ sung thêm một vector cell state chạy song song, giữ luồng thông tin ổn định suốt chuỗi và chỉ thay đổi có kiểm soát. Nhờ đó, LSTM có thể ghi nhớ mối quan hệ ngữ nghĩa xa hơn nhiều bước trong chuỗi so với RNN. Tuy nhiên, LSTM có cấu trúc phức tạp hơn, nhiều tham số hơn và tính toán chậm hơn so với RNN.

Bi-LSTM (Bidirectional LSTM) mở rộng LSTM bằng cách cho phép mạng đọc chuỗi dữ liệu theo hai chiều: từ trái sang phải và từ phải sang trái. Ở mỗi bước, trạng thái ẩn của từ hiện tại là kết quả ghép giữa hai trạng thái: một từ hướng xuôi và một từ hướng ngược. Điều này cực kỳ có lợi trong những bài toán mà ý nghĩa của một từ không chỉ phụ thuộc vào những gì xảy ra trước đó mà còn phụ thuộc vào những gì xảy ra sau đó. Ví dụ, trong bài toán phân loại thực thể (Named Entity Recognition), việc biết từ trước và từ sau sẽ giúp mô hình phân loại từ hiện tại chính xác hơn. Nhược điểm của Bi-LSTM là tốc độ chậm hơn LSTM đơn thuần và vẫn chịu ảnh hưởng khi chuỗi đầu vào quá dài.

Seq2Seq (Sequence to Sequence) là một kiến trúc mang tính đột phá khi lần đầu tiên cho phép ánh xạ một chuỗi đầu vào thành một chuỗi đầu ra có độ dài khác biệt. Seq2Seq sử dụng hai mạng riêng biệt: Encoder và Decoder. Encoder nhận toàn bộ chuỗi đầu vào, mã hóa thông tin vào một vector trạng thái cuối cùng. Decoder sử dụng vector này để bắt đầu sinh ra chuỗi đầu ra từng bước một. Seq2Seq rất mạnh trong các bài toán như dịch máy (Machine Translation), tóm tắt văn bản (Summarization), hay chatbot. Tuy nhiên, điểm yếu nghiêm trọng của Seq2Seq gốc là toàn bộ thông tin của cả câu đầu vào bị ép vào một vector duy nhất (context vector). Khi câu quá dài hoặc phức tạp, mô hình dễ bị mất thông tin và dự đoán kém chính xác. Điều này đã thúc đẩy các nghiên cứu tiếp theo để cải tiến kiến trúc này.

Tổng kết lại, các kiến trúc đời đầu như RNN, LSTM, Bi-LSTM, và Seq2Seq đều có một mục tiêu chung: tìm cách hiểu được chuỗi ngôn ngữ bằng cách lưu trữ ngữ cảnh khi di chuyển qua từng phần tử. Chúng đều cố gắng cân bằng giữa việc ghi nhớ thông tin dài hạn và tối ưu tốc độ tính toán. Dù vậy, các kiến trúc này đều có giới hạn nhất định, đặc biệt là khi xử lý những đoạn văn bản dài và phức tạp, dẫn đến sự ra đời của các mô hình tiên tiến hơn sau này.

### Kiến trúc Tranformer

Transformer là một kiến trúc mạng thần kinh chuyên biệt cho bài toán xử lý chuỗi, ra đời để giải quyết những vấn đề mà RNN gặp phải, như khó song song hóa và khó học được các mối liên hệ dài hạn. Transformer hoạt động hoàn toàn dựa trên cơ chế Attention, cho phép mô hình nhìn toàn bộ chuỗi một cách trực tiếp, thay vì đi tuần tự từng bước như RNN.

Bản thân kiến trúc Transformer bao gồm hai thành phần lớn: Encoder và Decoder.

Encoder nhận một chuỗi đầu vào và biến đổi nó thành một chuỗi vector ẩn giàu ngữ nghĩa. Decoder từ chuỗi vector ẩn đó sinh ra chuỗi đầu ra theo một cách tuần tự.

Trong Encoder và Decoder đều có các thành phần chính như sau: positional encoding, multi-head attention, masking, feed-forward network. Bây giờ ta sẽ đi lần lượt từng phần chi tiết.

### Positional Encoding

Transformer không có cơ chế tuần tự tự nhiên như RNN, vì vậy để mô hình hiểu được thứ tự của các từ trong chuỗi, cần phải cộng thêm vào vector embedding một thông tin về vị trí. Lớp positional encoding thực hiện việc đó.

Positional Encoding được tính bằng một công thức định nghĩa cố định, không có tham số học:

Với mỗi vị trí

pos và mỗi chiều i trong embedding vector, ta định nghĩa:

* PE(pos,2i) = sin(pos/(10000^2i/dmodel))
* PE(pos,2i+1) = cos(pos/(10000^2i/dmodel))

Ở đây:

* pospospos là vị trí từ trong câu.
* iii là chỉ số chiều của vector embedding.
* dmodeld\_{model}dmodel​ là độ dài vector embedding.

Nhờ sin và cos với tần số khác nhau, mô hình có thể suy ra vị trí tương đối và tuyệt đối giữa các từ.

Vector positional encoding này sẽ được **cộng trực tiếp** vào vector embedding từ trước khi đưa vào mạng.

### Multi-Head Attention

Attention là cơ chế cho phép mỗi từ trong câu "chú ý" vào tất cả các từ còn lại trong câu, thay vì chỉ dựa trên các bước liền kề.

Mỗi Attention "head" thực hiện việc tính toán Attention độc lập, rồi các kết quả từ nhiều head sẽ được nối lại.

Cơ chế Attention vận hành như sau: với mỗi từ, ta tạo ra ba vector:

Q (Query): biểu diễn yêu cầu cần tìm thông tin.

K (Key): biểu diễn những gì có thể cung cấp thông tin.

V (Value): chính nội dung thông tin.

Với Q,K,V được tạo ra từ cùng input embedding thông qua các lớp dense khác nhau.

Attention tính toán sự tương quan giữa Query và Key bằng tích vô hướng, sau đó dùng nó để trộn Value. Công thức:

* Attention(QKV) = softmax( ( (Q\*K^t) / sqrt(dk) ) \* V)

Ở đây:

* dk là kích thước của vector Key, dùng để làm chuẩn hóa tránh trị số quá lớn khi tích vô hướng.
* Softmax biến đổi điểm tương quan thành phân phối xác suất.

Khi có nhiều đầu (multi-head), ta thực hiện Attention nhiều lần độc lập:

* MultiHead(QKV) = Concat(head1,…, headn)\*W0

Mỗi head:

* Headi = Attention(Q\*wi^Q, K\*Wi^K, Vwi^V)

Các ma trận WiQ, WiK, WiV ​ là tham số học được. Multi-head attention giúp mô hình học được nhiều kiểu tương tác khác nhau giữa các từ.

### Masked Multi-Head Attention

Ở Decoder, để sinh từ thứ t, ta chỉ được phép sử dụng các từ trước đó (không được nhìn vào từ tương lai). Do đó ta cần Mask Attention.

Mask đơn giản là chặn các kết nối từ từ hiện tại đến các từ tương lai.

Cụ thể, khi tính QK T, ta cộng thêm một ma trận mask có giá trị -∞tại các vị trí bất hợp lệ, khiến softmax tại đó bằng 0.

* MaskedAttention(QKV) = softmax( ( (Q\*K^t+M) / sqrt(dk) ) \* V)

Trong đó:

* M là ma trận mask, giá trị −∞ nếu cần chặn.
* Nhờ đó, Decoder không "nhìn trước" khi sinh câu.

### Feed Forward Layer

* FFN(x) = Relu(xW+b)

Sau mỗi Attention block sẽ có một lớp Feed Forward Network giống nhau áp dụng cho từng từ (position-wise). Nó bao gồm 2 lớp dense:

Feed Forward cho phép mô hình học các biểu diễn phức tạp hơn tại mỗi vị trí độc lập.

Kỹ thuật tối ưu và đặc điểm

Transformer áp dụng:

* Residual Connection: cộng đầu vào ban đầu với đầu ra của mỗi block Attention và Feed Forward, để tránh mất mát thông tin khi qua nhiều tầng.
* Layer Normalization: chuẩn hóa mỗi đầu ra theo từng chiều, giúp ổn định gradient.
* Dropout: được chèn sau Attention và Feed Forward để chống overfitting.

Transformer giải quyết tốt bài toán song song hóa vì không phụ thuộc vào tính tuần tự, nhưng vẫn có điểm yếu: vì không có cơ chế tự nhiên cho vị trí như RNN, nên hoàn toàn phải dựa vào positional encoding. Ngoài ra, Attention toàn cục (global) sẽ khiến chi phí bộ nhớ tăng nhanh theo O(n 2) với độ dài chuỗi, làm Transformer khó áp dụng trực tiếp cho các văn bản cực kỳ dài.

# MÔ HÌNH LÝ THUYẾT

## Kiến trúc Collaborative Filtering (CF)

Collaborative Filtering (CF) là một trong những nguyên lý nền tảng và quan trọng nhất trong lĩnh vực hệ thống gợi ý (recommendation systems).

Mục tiêu của CF là dự đoán sự quan tâm hoặc hành vi của một người dùng đối với một sản phẩm, dựa trên mối quan hệ tương tác giữa người dùng và sản phẩm trong quá khứ.

Điểm đặc biệt của CF là nó không cần hiểu bản thân sản phẩm là gì, không cần hiểu người dùng là ai mà chỉ cần dựa trên mẫu hành vi được ghi nhận để đưa ra dự đoán.

### Nguyên lý hoạt động

CF hoạt động dựa trên một giả định đơn giản nhưng rất mạnh mẽ:

Nếu hai người dùng có hành vi tương tự trong quá khứ, thì trong tương lai họ cũng sẽ có sở thích tương tự.

Tương tự, nếu hai sản phẩm được nhiều người dùng chung yêu thích, thì chúng cũng có khả năng liên quan với nhau.

Có hai cách chính mà nguyên lý này được khai thác:

User-based CF: Dựa trên việc tìm những người dùng có hành vi giống nhau. Ví dụ, nếu A và B đều thích các sản phẩm X, Y, thì có khả năng sản phẩm Z mà A thích cũng sẽ hợp với B.

Item-based CF: Dựa trên việc tìm những sản phẩm tương tự nhau dựa trên các tập người dùng đã từng tương tác với chúng. Ví dụ, nếu sản phẩm X và Y thường được mua cùng nhau bởi nhiều người, thì khi ai đó mua X, ta sẽ gợi ý thêm Y.

CF làm việc thông qua việc so sánh "mẫu tương tác" (interaction patterns) giữa người dùng và sản phẩm để tìm ra sự tương đồng và từ đó dự đoán hành vi tiếp theo.

### Thuật toán cơ bản

Trong hình thức cổ điển, CF sử dụng các phương pháp tính độ tương đồng để đưa ra gợi ý. Một số kỹ thuật cơ bản:

Tính similarity giữa người dùng với người dùng, hoặc sản phẩm với sản phẩm, dựa trên cosine similarity, Pearson correlation, hoặc Jaccard index.

Xếp hạng các đối tượng dựa trên điểm số similarity.

Đưa ra đề xuất dựa trên những đối tượng có similarity cao nhất.

Trong thực tế, các hệ thống quy mô lớn sẽ làm việc với ma trận tương tác cực kỳ thưa (sparse matrix) và phải thực hiện các kỹ thuật giảm chiều, phân rã ma trận (matrix factorization) để dự đoán giá trị cho các ô còn thiếu.

CF thực chất là một bài toán suy diễn mối liên hệ ngầm giữa các phần tử trong hệ thống dựa trên quan sát hành vi, thay vì dựa trên các mô tả nội dung rõ ràng.

### Sự tối ưu và phát triển mới mẻ của CF

Ban đầu, CF chủ yếu sử dụng các kỹ thuật thống kê đơn giản để đo độ tương đồng và suy luận. Nhưng khi lượng dữ liệu và người dùng tăng lên nhanh chóng, CF phải đối mặt với nhiều thách thức:

Sparsity Problem: Dữ liệu thưa làm cho việc tính toán độ tương đồng trở nên kém tin cậy, do số lượng mẫu để so sánh rất ít.

Scalability: Khi số người dùng và sản phẩm tăng lên, việc tính toán độ tương đồng trở nên tốn kém, cần tối ưu hóa thuật toán hoặc sử dụng các kỹ thuật approximate.

Cold Start Problem: Khi có người dùng hoặc sản phẩm mới chưa có lịch sử tương tác, CF không thể đưa ra dự đoán tốt.

Để tối ưu, các hệ thống CF hiện đại đã tiến hành nhiều cải tiến:

Matrix Factorization: Ánh xạ người dùng và sản phẩm vào không gian vector ẩn (latent space) có chiều thấp để dễ dàng so sánh và dự đoán. Đây là nguyên lý đứng sau nhiều hệ thống như Netflix Prize Model.

Implicit Feedback: Thay vì yêu cầu người dùng đánh giá rõ ràng (rating), nhiều hệ thống CF hiện nay học từ các tương tác ngầm như click, view, time-spent để khai thác hành vi thực tế.

Hybrid Systems: CF được kết hợp với các hệ thống content-based filtering, metadata filtering để giảm thiểu cold-start và tăng tính chính xác.

Incremental Training: Thay vì train lại toàn bộ mô hình, nhiều hệ thống hiện đại cho phép cập nhật mô hình CF theo thời gian thực (online learning) khi có dữ liệu mới.

Sự đổi mới của CF trong kỷ nguyên dữ liệu lớn không chỉ dừng lại ở việc tối ưu mô hình mà còn ở việc tích hợp CF vào các nền tảng sản phẩm phức tạp như mobile, streaming media, hay personalized advertisement.

### Ứng dụng thực tế - Các công ty sử dụng CF

Collaborative Filtering là nền tảng cốt lõi đứng sau thành công của rất nhiều hệ thống gợi ý lớn trên thế giới:

Netflix: Netflix sử dụng matrix factorization làm nền tảng cho hệ thống gợi ý phim. Netflix Prize competition năm 2006 đã thúc đẩy sự phát triển mạnh mẽ của kỹ thuật CF.

Amazon: Hệ thống "Customers who bought this also bought" của Amazon là một ứng dụng kinh điển của item-based CF, nơi sản phẩm được đề xuất dựa trên lịch sử mua hàng của người dùng tương tự.

Spotify: Spotify áp dụng CF để đề xuất playlist cá nhân hóa cho người dùng, dựa trên hành vi nghe nhạc và tương tác của hàng triệu người dùng khác.

YouTube: Một phần quan trọng của recommendation engine của YouTube sử dụng CF để xác định video nào có khả năng người dùng sẽ thích, dựa trên lịch sử xem của những người dùng tương tự.

Alibaba: Trong hệ thống thương mại điện tử, CF được sử dụng để phân tích hành vi click và mua hàng, tối ưu hóa quá trình cá nhân hóa sản phẩm.

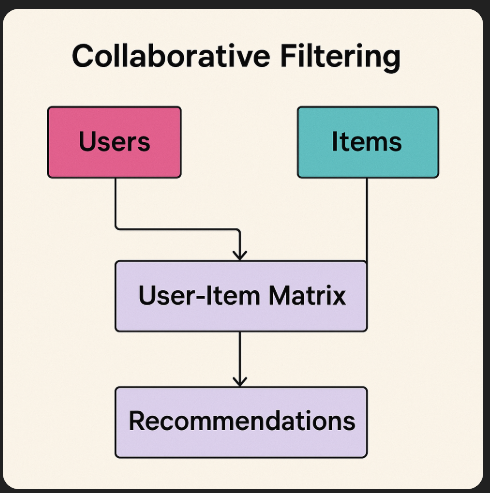
Ngoài ra, các công ty trong ngành game online, social network, e-learning platforms như Facebook, Coursera, LinkedIn cũng đều ứng dụng CF trong các hệ thống đề xuất bạn bè, khóa học, việc làm...

### Kết luận

Collaborative Filtering dù ra đời từ rất sớm nhưng vẫn đóng vai trò trung tâm trong nhiều hệ thống gợi ý hiện đại. Nguyên lý hoạt động đơn giản, trực tiếp và mang tính người dùng mạnh mẽ giúp CF có khả năng khai thác tốt tập tính hành vi ẩn trong dữ liệu.

Những tiến bộ mới về scale, tính mềm dẻo trong mô hình hóa, cùng với việc tích hợp các phương pháp hybrid, đã đưa CF trở thành một trong những kiến trúc "bền vững" và lâu dài nhất trong lĩnh vực AI cho hệ thống gợi ý.

Ngay cả trong thời đại Deep Learning bùng nổ, nhiều hệ thống recommendation thành công vẫn duy trì Collaborative Filtering như nền tảng cốt lõi, cho thấy sức sống và sự thích nghi đáng kinh ngạc của kiến trúc này.



Hình III‑1 Ảnh minh họa cho kiến trúc CF

## CF-based Encoder Transformer

Collaborative Filtering based Encoder Transformer là một kiến trúc hiện đại được phát triển nhằm kết hợp sức mạnh của Collaborative Filtering (CF) với năng lực biểu diễn mạnh mẽ của Transformer Encoder trong bài toán hệ thống gợi ý.

Nếu CF truyền thống chỉ dựa trên việc so sánh mẫu tương tác người dùng - sản phẩm một cách tuyến tính hoặc đơn giản hóa thành factorization, thì CF-based Encoder Transformer tiến thêm một bước lớn: học trực tiếp mối quan hệ phức tạp phi tuyến giữa người dùng và sản phẩm, thông qua cơ chế Attention đa chiều, không giới hạn.

### Nguyên lý hoạt động

Ý tưởng cốt lõi của CF-based Encoder Transformer là:

Thay vì chỉ ánh xạ người dùng và sản phẩm thành các vector cố định rồi nhân vô hướng đơn giản, ta để cho một mô hình dựa trên Attention tự động học cách liên kết người dùng và sản phẩm qua tương tác lịch sử của họ.

Mỗi người dùng sẽ được biểu diễn không chỉ bằng một vector ngẫu nhiên, mà bằng chuỗi lịch sử các sản phẩm họ đã tương tác.

Tương tự, mỗi sản phẩm cũng được biểu diễn bởi các đặc trưng động dựa trên hành vi của các người dùng đã tương tác với nó.

Transformer Encoder sẽ đọc các chuỗi tương tác này và học cách mô hình hóa mối quan hệ giữa các item, cách người dùng hành xử theo lịch sử tương tác, đồng thời tự động phát hiện pattern ngữ nghĩa sâu sắc hơn nhiều so với các phương pháp CF cổ điển.

### Thuật toán cơ bản

Về mặt thuật toán, hệ thống hoạt động như sau:

* Với mỗi người dùng, ta thu thập một chuỗi các sản phẩm mà người đó đã tương tác (mua, xem, click...).
* Các sản phẩm này được embedding thành vector đặc trưng.
* Chuỗi embedding đó được đưa vào Transformer Encoder, nơi từng sản phẩm trong chuỗi có thể "chú ý" tới các sản phẩm khác.
* Output cuối cùng là một vector đặc trưng toàn diện đại diện cho người dùng, được tổng hợp từ lịch sử hành vi đã qua.
* Vector người dùng này sau đó sẽ được dùng để tính toán điểm tương tác (scoring) với sản phẩm cần dự đoán, thông qua một cơ chế matching (như dot product hoặc multilayer perceptron).

Transformer Encoder ở đây có các khối chính:

* Positional Encoding: Bổ sung thông tin về thứ tự lịch sử tương tác (rất quan trọng vì hành vi người dùng theo thời gian có xu hướng thay đổi).
* Multi-Head Self Attention: Cho phép sản phẩm ở bước t có thể học sự liên kết không chỉ với sản phẩm kề trước mà còn với tất cả các sản phẩm khác trong lịch sử tương tác.
* Feed-Forward Layer: Tăng cường khả năng phi tuyến, giúp mô hình hóa các quan hệ phức tạp giữa các hành vi.

Điều đặc biệt là:

thay vì truyền một điểm tương tác đơn lẻ giữa user và item như CF truyền thống, toàn bộ quá trình tương tác quá khứ của người dùng được encode thành một hàm ngữ cảnh động, từ đó suy ra xác suất tương tác tiếp theo.

### Sự tối ưu và cải tiến mới mẻ

CF-based Encoder Transformer mang đến những cải tiến lớn so với các phương pháp CF trước đây:

Học phi tuyến: Attention có khả năng học các kiểu quan hệ phi tuyến giữa các sản phẩm, không cần giả định tương tác là tuyến tính như matrix factorization.

Khả năng hiểu thứ tự hành vi: Với Positional Encoding, mô hình biết được không chỉ những sản phẩm nào từng được tương tác, mà còn hiểu thứ tự chúng xảy ra — điều rất quan trọng trong modeling hành vi thời gian.

Tự động phát hiện pattern dài hạn: Transformer có receptive field toàn cục ngay từ đầu, nên dễ dàng học được mối liên hệ giữa các sản phẩm cách xa nhau trong lịch sử tương tác.

Xử lý chuỗi dài mạnh mẽ: Khả năng song song hóa tính toán của Transformer giúp hệ thống xử lý lịch sử dài nhanh hơn RNN-based CF rất nhiều.

Tuy nhiên, nhược điểm tự nhiên của Transformer cũng hiện hữu:

Chi phí tính toán: Do Attention toàn cục cần O(n2)bộ nhớ và thời gian với độ dài chuỗi, hệ thống có thể khó mở rộng trực tiếp với lịch sử siêu dài.

Overfitting: Với khả năng biểu diễn mạnh, hệ thống dễ bị overfit khi dữ liệu tương tác lịch sử không đủ lớn hoặc không đa dạng.

Để khắc phục, các hệ thống hiện đại thường giới hạn độ dài lịch sử (ví dụ lấy 50 lần tương tác gần nhất), áp dụng kỹ thuật regularization như dropout, label smoothing, và sử dụng cơ chế masking phù hợp.

### Các công ty và ứng dụng thực tế

Kiến trúc CF-based Encoder Transformer đã được nhiều công ty lớn nghiên cứu và áp dụng:

YouTube: Recommendation engine của YouTube hiện đại sử dụng các kiến trúc tương tự Transformer Encoder để encode lịch sử xem video của người dùng nhằm gợi ý video tiếp theo.

Alibaba: Nghiên cứu tại Alibaba cho thấy việc dùng Transformer Encoder để model hóa chuỗi click/mua hàng giúp hệ thống gợi ý thương mại điện tử hiệu quả hơn các phương pháp CF truyền thống.

Amazon: Trong các hệ thống Amazon Personalize, người ta đã ứng dụng các kiến trúc Attention-based để học vector đại diện người dùng từ sequence of events thay vì chỉ rely vào CF factorization.

TikTok: Hệ thống gợi ý nội dung video ngắn của TikTok heavily sử dụng Transformer-based encoders để capture hành vi ngữ nghĩa giữa các lượt xem và tương tác.

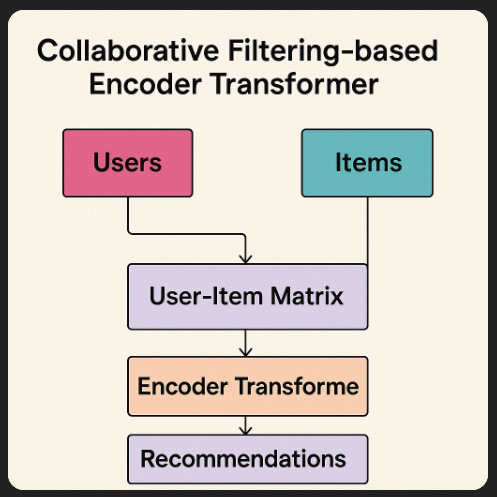
Ngoài ra, trong học thuật, các paper như "SASRec: Self-Attentive Sequential Recommendation" đã tiên phong chứng minh rằng việc dùng self-attention để encode lịch sử tương tác cho bài toán recommendation đạt độ chính xác cao vượt trội so với CF cổ điển.

### Kết luận

CF-based Encoder Transformer không chỉ là sự nâng cấp của Collaborative Filtering, mà là một bước tiến hoàn toàn mới về cách thức chúng ta nhìn nhận dữ liệu tương tác người dùng.

Bằng cách thay đổi trọng tâm từ việc mô hình hóa từng người dùng hoặc sản phẩm thành việc mô hình hóa các chuỗi hành vi tương tác, và dùng Transformer để học mối liên hệ ngữ nghĩa sâu xa giữa các hành vi đó, CF-based Transformer đã mở ra một hướng đi hiệu quả, mạnh mẽ và linh hoạt cho các hệ thống gợi ý hiện đại.

Trong tương lai gần, với sự tối ưu về tính toán và các biến thể Attention hiệu quả hơn, chắc chắn Collaborative Filtering dựa trên Encoder Transformer sẽ còn chiếm vị trí trung tâm trong mọi hệ thống recommendation lớn trên toàn thế giới.



Hình III‑2 Ảnh minh họa kiến trúc CF-based Encode Tranformer

## Dropout

Trong học sâu (deep learning), dropout là một kỹ thuật regularization quan trọng, được phát triển nhằm giảm hiện tượng overfitting. Overfitting xảy ra khi mô hình học quá kỹ các chi tiết nhiễu trong tập huấn luyện, dẫn tới khả năng tổng quát kém trên dữ liệu mới. Dropout ra đời như một phương pháp đơn giản nhưng cực kỳ hiệu quả để làm cho các mô hình học sâu trở nên mạnh mẽ hơn, đặc biệt khi số lượng tham số lớn và dữ liệu không hoàn toàn lý tưởng.

### Cách hoạt động của Dropout

Trong quá trình huấn luyện, tại mỗi lần forward (lan truyền tiến), dropout sẽ ngẫu nhiên chọn ra một tỷ lệ phần trăm các neuron trong mạng và bỏ qua chúng (tạm thời set output bằng 0). Ví dụ, với tỷ lệ dropout là 0.5, thì trung bình 50% số lượng neuron trong lớp đó sẽ bị bỏ qua trong một lần tính toán. Các neuron bị dropout sẽ không tham gia cả vào tính toán đầu ra và gradient khi backpropagation.

Tuy nhiên, khi đánh giá mô hình (inferencing), dropout sẽ không được áp dụng. Thay vào đó, đầu ra của lớp sẽ được nhân với xác suất giữ lại (thường là

1-p, với p là tỷ lệ dropout) để cân bằng độ lớn kỳ vọng của tín hiệu đầu ra. Điều này đảm bảo rằng kỳ vọng của tổng đầu ra tại train và inference là như nhau.

### Thuật toán tính toán cụ thể

Giả sử một lớp neural network có đầu vào là vector x, trọng số W, và hàm kích hoạt f.

Tại bước training:

* Sinh một vector mask m có cùng kích thước với x,
* trong đó mỗi phần tử của m là 1 với xác suất 1-p và 0 với xác suất p.
* Tính đầu vào bị dropout: x ′ =x×m
* Tính output: y=f(Wx ′ )

Tại bước inference:

* Không áp dụng mask, sử dụng toàn bộ x.
* Tuy nhiên, các trọng số hoặc output có thể được nhân với hệ số (1−p) để cân bằng cường độ tín hiệu.

Công thức dropout trong forward pass lúc training:

* Y = f(W(x\*m))

Trong đó:

* ⊙ là phép nhân từng phần tử.

Dropout làm cho mạng giống như việc ensemble hàng ngàn mô hình con nhỏ với các neuron khác nhau được bật/tắt khác nhau. Vì mỗi lần huấn luyện sẽ có một tập các neuron hoạt động khác nhau, mô hình buộc phải học các biểu diễn đặc trưng ổn định hơn, không dựa dẫm vào bất kỳ neuron đơn lẻ nào. Điều này làm giảm khả năng overfit.

Dropout cũng làm tăng khả năng kháng nhiễu của mạng, bởi vì nó tạo ra sự ngẫu nhiên có kiểm soát trong lúc học.

### Liên hệ với CF-based Encoder Transformer

Trong kiến trúc Collaborative Filtering (CF) based Encoder Transformer, chúng ta thường dùng một module encoder gồm nhiều lớp attention và feed-forward nối tiếp nhau. Đây là mô hình với số lượng tham số lớn, và đặc biệt khi áp dụng cho bài toán gợi ý sản phẩm/phim, dữ liệu huấn luyện thường không đều, có nhiều trường hợp nhãn "0" hoặc "1" không cân bằng. Điều này khiến mô hình dễ dàng bị overfitting: học quá nhanh các pattern nhỏ lẻ, làm giảm hiệu quả thực tế.

Trong Encoder Transformer, dropout thường được chèn vào những vị trí sau:

Sau khi tính attention (trên output của multi-head attention).

Sau khi tính feed-forward (dense).

Trên embedding đầu vào của từ hoặc sản phẩm.

Cụ thể:

Trong lớp Multi-Head Attention, sau khi tính toán Attention Score và Weighted Sum, ta áp dụng dropout để tránh tình trạng attention quá phụ thuộc vào một vài phần tử.

Trong lớp Feed-Forward Network (FFN), sau khi đi qua Dense-ReLU-Dense, ta áp dụng dropout trước khi cộng với skip connection.

Trên embedding input (embedding lookup), ta cũng có thể dropout một số phần tử trong vector embedding.

Dropout trong CF-based Encoder Transformer là một cơ chế sống còn để giúp mô hình học được các biểu diễn ổn định, tránh overfitting, tăng khả năng tổng quát hóa, và làm cho vector embedding của người dùng và sản phẩm trở nên giàu thông tin hơn. Không có dropout, mô hình Transformer dù mạnh đến đâu cũng sẽ rất dễ học lệch và làm mất cân bằng độ biểu diễn giữa các chiều embedding.

## Regularizers

Trong học sâu (deep learning), regularizers là tập hợp các kỹ thuật nhằm ép mô hình học sâu phải đơn giản hóa biểu diễn của nó, từ đó giúp mô hình giảm overfitting và tăng khả năng tổng quát hóa. Khi mô hình có quá nhiều tham số (hàng triệu trọng số) và dữ liệu không đủ lớn hoặc không đủ phong phú, mô hình sẽ dễ học thuộc lòng các mẫu dữ liệu huấn luyện thay vì học các quy luật tổng quát. Regularization (sự chính quy) can thiệp vào quá trình huấn luyện để chống lại xu hướng này.

### Nguyên lý cơ bản của Regularizers

Trong thuật toán tối ưu hóa mô hình học sâu, mục tiêu thông thường là tìm bộ trọng số θ sao cho hàm loss L(θ) trên tập huấn luyện là nhỏ nhất. Tuy nhiên, nếu chỉ tối ưu loss gốc, mô hình dễ trở nên phức tạp quá mức.

Để tránh điều này, regularizers sẽ thêm vào loss gốc một hàm phạt (penalty term) nhằm trừng phạt các mô hình có trọng số quá lớn hoặc quá phức tạp. Hàm loss huấn luyện lúc này trở thành:

* Ltotal(0) = Loriginal(0) + λ \*R(0)

trong đó:

* Loriginal(θ) là loss gốc (ví dụ Cross Entropy Loss hoặc MSE Loss),
* R(θ) là hàm regularization (ví dụ L1 norm hoặc L2 norm),
* λ là hệ số điều chỉnh mức độ ảnh hưởng của regularization.

Việc thêm hàm phạt vào loss buộc mô hình phải cân bằng giữa việc tối ưu hóa loss huấn luyện và giữ cho trọng số nhỏ hoặc phân bố hợp lý.

Các kỹ thuật Regularizers phổ biến

Có nhiều kỹ thuật regularization, mỗi kỹ thuật có ý nghĩa và hiệu ứng riêng:

* L2 Regularization (Weight Decay):
* Công thức:
* Trừng phạt các trọng số lớn bằng cách làm chúng nhỏ lại.
* Làm cho mô hình ưa thích những trọng số có giá trị nhỏ, đồng đều.

L1 Regularization:

* Khuyến khích các trọng số trở thành 0.
* Dẫn đến mô hình sparse (rất nhiều trọng số bằng 0).

Elastic Net Regularization:

* Kết hợp L1 và L2:
* Vừa khuyến khích sự đơn giản (sparse), vừa tránh việc bỏ qua thông tin hữu ích.
* Early Stopping (dừng huấn luyện sớm):
* Không phải trực tiếp thêm vào loss, nhưng ý tưởng cũng là regularization: dừng training khi val loss không còn cải thiện, tránh mô hình học quá chi tiết dữ liệu train.

Dropout:

* Được xem là một regularizer ngẫu nhiên.
* Đã trình bày kỹ ở phần trước.

### Tại sao cần Regularizers trong CF-based Encoder Transformer

Kiến trúc Collaborative Filtering based Encoder Transformer là mô hình lớn, với nhiều lớp attention, feed-forward, và các vector embedding có kích thước cao. Điều này dẫn tới:

Số lượng tham số cực lớn (hàng triệu, thậm chí hàng trăm triệu).

Nguy cơ overfitting cao, nhất là khi dữ liệu lịch sử người dùng không đủ phong phú.

Các vector embedding dễ dàng học được các pattern nhỏ lẻ hoặc bias ngẫu nhiên.

Do đó, khi áp dụng CF-based Transformer, regularizers là công cụ quan trọng để đảm bảo:

Các vector embedding của người dùng và sản phẩm không bùng nổ về giá trị, tránh hiện tượng vanishing/exploding gradient.

Các lớp attention và feed-forward trong encoder học các biểu diễn có tính tổng quát, ổn định hơn.

Cải thiện hiệu quả dự đoán trên các sản phẩm, người dùng mà mô hình chưa từng thấy trong tập train.

### Ứng dụng thực tế trong Encoder Transformer

Khi training CF-based Transformer, regularization được áp dụng theo các điểm sau:

Trọng số Dense Layer: thêm kernel\_regularizer=l2(0.001) vào các Dense layer trong Attention và Feed-Forward Network.

Embedding Layer: áp dụng L2 regularization để hạn chế việc embedding trở nên quá lớn.

Dropout: thêm vào sau mỗi layer attention, feed-forward như một regularizer động.

Early Stopping: thường được kích hoạt dựa trên val loss để giảm nguy cơ overfitting.

## GlobalAveragePooling1D

Trong các mô hình deep learning, đặc biệt khi làm việc với dữ liệu dạng chuỗi (sequential data) như văn bản, lịch sử tương tác người dùng, hay chuỗi embedding sản phẩm, ta thường gặp một vấn đề:

Đầu ra từ các layer như Transformer Encoder có dạng 3 chiều (batch\_size, sequence length, feature dim), trong khi các lớp dense fully-connected phía sau thường yêu cầu đầu vào dạng 2 chiều (batch\_size, feature\_dim).

Vì vậy, ta cần một kỹ thuật để nén trục sequence lại và flatten dữ liệu thành một vector đại diện duy nhất cho cả chuỗi.

Một phương pháp rất phổ biến, hiệu quả và đơn giản cho việc này là Global Average Pooling1D.

### Cách hoạt động của GlobalAveragePooling1D

Kỹ thuật GlobalAveragePooling1D sẽ:

Tính trung bình toàn bộ các vector đặc trưng theo chiều sequence (trục 1).

Tức là, với mỗi đặc trưng (feature) riêng lẻ, ta lấy giá trị trung bình qua toàn bộ các bước trong chuỗi.

Nếu đầu vào có shape: (batch size, sequence length, feature dim)

thì đầu ra sau Global Average Pooling1D sẽ có shape: (batch size, feature dim)

Công thức toán học

Giả sử

* X là tensor đầu vào với kích thước (B,T,D),

Trong đó:

* B là batch size,
* T là chiều dài sequence,
* D là số lượng feature.
* Đầu ra

Y có kích thước (B,D), với mỗi phần tử:

Tức là tại mỗi batch và mỗi feature, ta lấy trung bình các giá trị theo trục thời gian (sequence).

Tại sao lại dùng GlobalAveragePooling1D trong CF-based Encoder Transformer

Trong CF-based Encoder Transformer, mỗi user và item sau khi đi qua encoder sẽ có biểu diễn dạng chuỗi, ví dụ 10 bước (sequence length = 10), mỗi bước 128 chiều (feature dimension = 128).

Nhưng các bước này thực chất không mang ý nghĩa thứ tự mạnh như trong bài toán ngôn ngữ, mà chủ yếu là tổng hợp các đặc trưng qua nhiều bước attention.

Vì vậy:

* Không cần giữ từng bước riêng lẻ.
* Không nên đơn giản flatten thẳng toàn bộ vì làm tăng quá nhiều tham số.
* Không phù hợp việc lấy bước cuối như RNN truyền thống.
* Thay vào đó, việc dùng GlobalAveragePooling1D giúp:
* Lấy trung bình thông tin từ toàn bộ chuỗi embedding.
* Tạo ra một vector duy nhất tóm tắt toàn bộ lịch sử.
* Đảm bảo giữ tính ổn định và tính tổng quát hóa cao.
* Giảm đáng kể số lượng tham số khi đưa vào các lớp Dense phía sau.

### So sánh nhanh với Flatten

Flatten: kéo thẳng tensor thành vector siêu dài, có shape sequence length × feature dim.

GlobalAveragePooling1D: rút gọn chiều sequence, chỉ còn lại feature dim, giảm nhiều tham số hơn và giữ tính tổng quát hóa.

Những điểm mạnh và hạn chế

Ưu điểm:

* Đơn giản, nhanh, dễ tính toán.
* Giảm đáng kể số lượng tham số, tránh overfitting.
* Hữu ích trong CF-based Transformer vì thứ tự trong chuỗi embedding không quá quan trọng.

Hạn chế:

* Làm mất thông tin chi tiết theo thời gian.
* Nếu thứ tự hoặc vị trí có ý nghĩa mạnh, thì GlobalAveragePooling1D có thể làm mất mát thông tin.

Tuy nhiên trong hệ CF-based Encoder Transformer, nơi ta tổng hợp đặc trưng người dùng hoặc sản phẩm qua nhiều tín hiệu khác nhau, việc dùng GlobalAveragePooling1D là rất hợp lý và được khuyến khích.

## Lớp Positional Encoding

Trong Transformer nguyên thủy (ví dụ trong NLP), positional encoding thường được tính bằng công thức hàm cos, sin tuần hoàn dựa trên vị trí token và chiều của vector embedding, nhằm đưa thông tin vị trí vào cho mạng.

Tuy nhiên, trong rất nhiều biến thể hiện đại, đặc biệt với bài toán như CF-based Encoder Transformer (Collaborative Filtering dạng encoder-based), kỹ thuật positional encoding có sự thay đổi:

Không dùng hàm cos-sin, mà dùng một lớp Dense (Fully Connected Layer) để ánh xạ thẳng vị trí thành một vector học được, rồi cộng vào embedding.

### Cách hoạt động cụ thể

Ban đầu, ta có một chỉ số vị trí (index) cho từng phần tử trong chuỗi, ví dụ : 0,1,2,3,...,n.

Các chỉ số vị trí này được ánh xạ thành các vector bằng cách đưa qua một lớp Dense.

Lớp Dense này có số chiều đầu ra bằng đúng số chiều của embedding (ví dụ 128, 256...).

Vector vị trí sau ánh xạ được cộng trực tiếp vào embedding tương ứng.

Tóm lại, thay vì tính pos encoding theo hàm sin-cos cố định, ta học luôn vector vị trí thông qua trọng số của một lớp Dense.

### Về mặt thuật toán

Giả sử:

* positions∈RT là vector chứa các chỉ số vị trí, T là sequence length.
* DenseLayer là lớp Dense với output dimension bằng embedding dimension D.

Quá trình tính toán positional embedding mới sẽ là:

* pos\_embedding=DenseLayer(positions)

Sau đó:

* final\_embedding=token\_embedding+pos\_embedding

Trong đó:

* token\_embedding là embedding vector từ các ID user/item.
* pos\_embedding là vector được Dense học ra cho mỗi vị trí.

Ý nghĩa của cách làm này:

* Giúp mạng học ra cách mã hóa vị trí tối ưu nhất cho bài toán, thay vì áp đặt một quy luật tuần hoàn cứng như sin-cos.
* Rất hữu ích trong các bài toán không thuần ngôn ngữ, như lịch sử hành vi người dùng, lịch sử mua sắm, lịch sử click, vốn dĩ không có nhịp tuần hoàn như ngôn ngữ tự nhiên.
* Giúp tối ưu hóa việc mã hóa vị trí phù hợp theo domain-specific data.

Ưu điểm và tối ưu

* Flexible: Không bắt buộc phải theo quy luật tuần hoàn.
* Learnable: Các vector pos encoding có thể thích nghi theo tập dữ liệu và bài toán.
* Dễ tính toán: Vì chỉ cần một lớp Dense đơn giản, không cần hàm sin-cos phức tạp.
* Nhanh: Dense đơn giản rất nhanh trong thực tế train model.

Hạn chế nhỏ

* Do pos encoding là learnable, nên yêu cầu cần có lượng dữ liệu đủ để học đúng.
* Nếu dữ liệu rất nhỏ, có thể dẫn đến việc overfit vị trí.

## kỹ thuật gộp input và Embeding riêng trước khi đưa vào Transformer

Trong kiến trúc CF-based Encoder Transformer, hoặc các biến thể khác có dữ liệu dạng structured input (ví dụ lịch sử mua hàng, hành vi người dùng), thông thường, các loại input khác nhau sẽ được gộp lại và Embeding riêng biệt theo một quy luật.

Cách làm này như sau:

* Mỗi trường dữ liệu riêng biệt (ví dụ: user\_id, item\_id, category, brand, position,...) sẽ có một bảng embedding riêng.
* Quy luật gộp input thường là dựa vào logic tự nhiên của dữ liệu: những trường discrete (dạng chỉ số rời rạc) thì dùng Embedding, còn các trường continuous (giá trị liên tục như giá tiền, thời gian) có thể normalize rồi cũng nhúng qua Dense thành vector cùng kích thước embedding.

Sau khi mỗi input qua embedding riêng:

* Các embedding vector này được nối (concatenate) lại thành một vector lớn duy nhất cho mỗi sample.
* Tuy nhiên, vì các embedding từ các trường khác nhau có thể không đồng đều, nên sau khi nối, ta cần một bước xử lý tiếp theo để chuẩn hóa kích thước đầu vào.
* Kỹ thuật reshape bằng Dense không activation, không bias
* Để xử lý đồng bộ đầu vào cho Encoder, ta cho vector sau khi nối đi qua một lớp Dense với các đặc điểm:
* Không activation (activation = None), nghĩa là kết quả chỉ đơn thuần là nhân ma trận tuyến tính.
* Không bias (use\_bias = False), nghĩa là không cộng thêm một hệ số cố định nào vào kết quả.

Mục tiêu của Dense này là:

* Reshape lại vector về đúng kích thước embedding chuẩn (ví dụ 128, 256, 512...).
* Giữ nguyên tính chất thông tin của embedding gốc, chỉ thay đổi về hình thức kích thước mà thôi.
* Không làm biến dạng hay phi tuyến tính hóa dữ liệu, đảm bảo thông tin thô vẫn còn nguyên.

## Kiến trúc micro-service và sql server cho hệ thống backend

Trong dự án web của mình, bạn đã áp dụng kiến trúc microservice cho backend để dễ dàng mở rộng và bảo trì hệ thống. Kiến trúc microservice chia nhỏ các chức năng của hệ thống thành các dịch vụ độc lập, mỗi dịch vụ đảm nhiệm một nhiệm vụ cụ thể như quản lý người dùng, xử lý các yêu cầu tìm kiếm, xử lý các hành vi người dùng (click, comment, tìm kiếm), và tính toán các đề xuất sản phẩm. Điều này không chỉ giúp tăng khả năng mở rộng mà còn cải thiện tính linh hoạt và khả năng tái sử dụng các dịch vụ. Các microservice giao tiếp với nhau qua các API RESTful hoặc gRPC, và mỗi dịch vụ có thể được triển khai độc lập, giúp việc bảo trì, nâng cấp hoặc thay thế một phần của hệ thống trở nên dễ dàng hơn. Thêm vào đó, việc áp dụng microservice giúp phân tán tải công việc và tối ưu hóa hiệu suất, đồng thời hỗ trợ việc tích hợp mô hình Collaborative Filtering (CF) dựa trên Transformer để đưa ra các đề xuất sản phẩm chính xác cho người dùng, dựa trên hành vi tìm kiếm, click và các tương tác khác.

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) do Microsoft phát triển, được sử dụng rộng rãi trong các ứng dụng doanh nghiệp để lưu trữ và quản lý dữ liệu. Với khả năng xử lý hiệu quả các truy vấn phức tạp, SQL Server hỗ trợ các tính năng mạnh mẽ như bảo mật, sao lưu và phục hồi, và khả năng mở rộng linh hoạt, giúp doanh nghiệp duy trì và quản lý lượng dữ liệu lớn. SQL Server sử dụng ngôn ngữ truy vấn SQL (Structured Query Language) để thực hiện các thao tác như truy vấn, cập nhật, chèn và xóa dữ liệu, đồng thời cung cấp các công cụ mạnh mẽ như SQL Server Management Studio (SSMS) để quản lý và giám sát cơ sở dữ liệu.

Một trong những điểm mạnh của SQL Server là tính năng tích hợp với các công cụ khác trong hệ sinh thái Microsoft, như .NET và Azure, giúp việc phát triển ứng dụng trở nên mượt mà hơn. SQL Server cũng hỗ trợ các tính năng nâng cao như chỉ mục toàn văn, báo cáo, phân tích dữ liệu và các chức năng phân tán, làm cho nó trở thành sự lựa chọn hàng đầu cho các ứng dụng đòi hỏi hiệu suất và tính ổn định cao.

## Công nghệ cho backend và fonend của web

ReactJS là thư viện JavaScript mã nguồn mở do Facebook phát triển, giúp xây dựng giao diện người dùng (UI) cho ứng dụng web. React sử dụng mô hình component, cho phép tái sử dụng và quản lý giao diện dễ dàng. Nó nổi bật với tính năng Virtual DOM, giúp tối ưu hiệu suất bằng cách chỉ cập nhật các phần thay đổi của giao diện, thay vì toàn bộ trang.

Spring Boot là một framework mã nguồn mở dựa trên Java, giúp phát triển ứng dụng web và dịch vụ microservice nhanh chóng và dễ dàng. Spring Boot tự động cấu hình nhiều tính năng, giảm thiểu việc cấu hình thủ công và cho phép triển khai ứng dụng với ít bước nhất. Nó hỗ trợ tích hợp với nhiều công nghệ như Hibernate, JPA, và các công cụ WebSocket, giúp xây dựng các ứng dụng backend mạnh mẽ và scalable.

# MÔ HÌNH THỰC NGHIỆM

## Dữ liệu

### Chuẩn bị dữ liệu

Trong quá trình xây dựng hệ thống đề xuất dựa trên kiến trúc CF-based Encoder Transformer, một vấn đề cực kỳ quan trọng là chuẩn bị tập dữ liệu phù hợp để mô hình có thể học được các mối quan hệ phức tạp giữa người dùng và sản phẩm. Tuy nhiên, trong thực tế, việc tìm kiếm hoặc thu thập một bộ dữ liệu thực sự lý tưởng cho dự án là cực kỳ khó khăn, do nhiều lý do:

* Các công ty lớn sở hữu dữ liệu real về hành vi người dùng đều bảo mật rất chặt chẽ, bởi dữ liệu này liên quan trực tiếp đến quyền riêng tư cá nhân và tài sản kinh doanh chiến lược.
* Các bộ dữ liệu mở hiện có thường quá đơn giản, thiếu chiều sâu, hoặc không thể hiện được các hành vi phức tạp như thay đổi hành vi theo mùa, theo xu hướng, theo sự kiện.
* Ngoài ra, quy mô cần thiết để train một mô hình deep learning hiệu quả thường phải đạt hàng trăm ngàn đến hàng triệu mẫu, trong khi những bộ dataset công khai đa phần nhỏ và không đủ tính đại diện.
* Các dữ liệu thương mại điện tử thực tế còn rất nhiễu, mất cân bằng, chứa nhiều missing data — đây là những đặc điểm cực kỳ quan trọng để mô phỏng đúng thực tế nhưng lại rất khó tìm trong dataset mẫu.

Vì những lý do đó, để đảm bảo rằng mô hình được huấn luyện trong điều kiện sát thực tế nhất, chúng ta quyết định tự xây dựng dữ liệu mô phỏng, vừa đảm bảo được quy mô lớn, vừa kiểm soát được mức độ phức tạp mong muốn.

### Cách xây dựng dữ liệu mô phỏng:

Bước đầu tiên là random các đặc trưng cơ bản cho từng người dùng và sản phẩm:

* Các đặc trưng người dùng như độ tuổi, giới tính, vùng miền, thói quen tiêu dùng.
* Các đặc trưng sản phẩm như thể loại phim, thời điểm phát hành, mức độ nổi tiếng, giá tiền, chủ đề theo mùa.

Mỗi trường này được gán ngẫu nhiên nhưng tuân theo một số phân phối xác suất có kiểm soát.

Tiếp theo, để tạo mối quan hệ phức tạp giữa người dùng và sản phẩm, chúng ta bổ sung các luật xác suất phức tạp:

* Ví dụ: Người dùng từ vùng lạnh có xác suất cao hơn mua áo khoát về mùa đông, hoặc phim đồ hóa trang có khả năng cao được chọn vào dịp Halloween.
* Người trẻ có xác suất cao hơn yêu thích phong cách trẻ trung năng động.
* Người dùng có thói quen tiêu nhiều tiền thường chọn sản phẩm mới ra mắt, giá cao.
* Các sản phẩm theo xu hướng (trend) được thêm xác suất boost tạm thời, nhưng giảm dần theo thời gian.

Các xác suất này không chỉ phụ thuộc vào một đặc trưng, mà còn giao thoa giữa nhiều đặc trưng cùng lúc. Ví dụ: Một người trẻ từ thành phố lớn vào mùa hè sẽ có xác suất cao mua áo sơ mi hoặc đồ mùa hè.

Ngoài ra, để tăng độ chân thực, chúng ta thêm nhiễu vào dữ liệu:

* Khoảng 10% nhãn được random hoàn toàn bất chấp logic xác suất phía trên, nghĩa là hành vi của người dùng với sản phẩm đó hoàn toàn ngẫu nhiên.
* Điều này mô phỏng các hành vi bất định, bất ngờ của người dùng trong thực tế, khi họ có thể click hoặc mua những sản phẩm ngoài mong đợi vì những lý do cá nhân không thể dự đoán được.

Tóm lại:

Việc tự xây dựng dữ liệu theo cách này cho phép mô hình:

* Tiếp xúc với logic phức tạp đa chiều, đúng như hành vi thật của người dùng trên các nền tảng thương mại điện tử.
* Chịu được nhiễu để tăng khả năng tổng quát hóa trong môi trường thực tế, nơi dữ liệu luôn chứa sai số.
* Huấn luyện với quy mô lớn (tạo ra hàng trăm nghìn đến hàng triệu mẫu tùy yêu cầu) mà không lo giới hạn bản quyền hay bảo mật.
* Cách chuẩn bị dữ liệu này gần như bắt buộc đối với các dự án nghiên cứu hoặc thử nghiệm hệ thống đề xuất mới, khi không thể tiếp cận dữ liệu real-world trong giai đoạn đầu.

### Tiêu chí của bộ dữ liệu

Phải mô phỏng các quy luật hành vi thực tế

* Dữ liệu không được random hoàn toàn.
* Phải có mối liên hệ xác suất rõ ràng giữa đặc trưng đầu vào và nhãn đầu ra.
* Các quy luật cần xuất phát từ kiến thức ngành (domain knowledge):

Ví dụ: tuổi trẻ thích phong cách trẻ trung, mùa đông thường mua áo khoát, điểm cao dễ được chọn hơn.

* Phải kết hợp đa yếu tố (multivariate relationship)
* Một hành vi (ví dụ: quyết định mua) không thể chỉ do 1 đặc trưng đơn lẻ quyết định.
* Phải kết hợp nhiều đặc trưng theo dạng cộng hưởng, ví dụ: tuổi + giới tính + mùa + thể loại + điểm số → mới quyết định xác suất mua.
* Các đặc trưng nên vừa độc lập, vừa có tương tác chéo (cross-feature interaction).

Phải đưa vào nhiễu có kiểm soát (controlled noise)

Trong thực tế, luôn có những trường hợp ngoại lệ không tuân theo bất kỳ logic nào.

Bộ dữ liệu ảo phải giả lập được điều đó, ví dụ:

* 10% mẫu được random hóa nhãn, bất chấp đặc trưng.

Điều này giúp mô hình học tính tổng quát và không overfit vào quy luật giả lập.

Phải duy trì phân phối xác suất hợp lý

* Xác suất dự đoán (buy\_prob) phải được clamp vào [0, 1].
* Không để xác suất < 0 hoặc > 1 gây lệch logic.

Phải đảm bảo tỷ lệ nhãn 0/1 tổng thể không quá mất cân bằng (trừ khi cố tình).

Phải tạo độ phong phú về đặc trưng đầu vào

Không chỉ tạo đơn giản từng đặc trưng riêng lẻ, mà phải:

* Kết hợp danh sách (top 10 đồ đã xem, top 10 thể loại yêu thích...).
* Kết hợp số liệu liên tục (điểm sản phẩm, số sản phẩm đã xem).
* Kết hợp các thông tin rời rạc (thể loại, mùa).

Điều này làm cho dữ liệu đa dạng chiều đặc trưng, giúp mô hình cần phải học tốt hơn.

Phải giả lập được các xu hướng thời gian (seasonality)

Các hành vi thay đổi theo thời gian/mùa vụ phải được đưa vào:

* Ví dụ mùa đông mua áo khoát, mùa hè mua áo thun.

Đây là một yếu tố cực kỳ quan trọng trong behavior prediction thực tế.

Phải có yếu tố tích lũy hành vi người dùng

* Người dùng đã mua nhiều sản phẩm thì khả năng mua tiếp có thể tăng nhẹ.
* Phải có tham số như count\_pay\_product\_ để phản ánh hành vi lịch sử.

Phải giữ được cân bằng giữa đơn giản và phức tạp

* Không quá đơn giản (dễ bị mô hình "thuộc lòng").
* Không quá phức tạp (gây khó khăn cho quá trình huấn luyện).

Quy luật nên ở mức vừa đủ phức tạp để thách thức mô hình, nhưng vẫn giữ được tính tổng quát.

Phải đảm bảo khả năng mở rộng dữ liệu

* Bộ dữ liệu phải thiết kế sao cho:
* Dễ dàng tăng số lượng mẫu (scalability).
* Dễ dàng thêm mới đặc trưng hoặc quy luật khác nếu cần mở rộng trong tương lai.

Phải mô phỏng sự sai lệch cá nhân (personal bias)

Không phải người dùng nào cũng theo cùng một logic:

* Ví dụ: Một số người trung niên vẫn thích phim hành động.

Dữ liệu ảo phải để chừa ra các trường hợp "bất thường" để mô hình học được sự linh hoạt.

### Xử lý dữ liệu

Trong quá trình xử lý dữ liệu đầu vào, trước hết, toàn bộ thông tin của người dùng và sản phẩm được trích xuất từ file dữ liệu gốc. Để đảm bảo tính thống nhất và thuận tiện cho việc đưa vào mô hình, các cột dữ liệu được xử lý riêng biệt dựa trên từng loại đặc trưng.

Đối với các đặc trưng dạng số như tuổi người dùng, tháng sinh, giới tính, tổng tiền mua hàng trung bình, số lần click vào sản phẩm, số lần mua hàng, số lần hủy đơn, số lần trả hàng, số lần bỏ sản phẩm khỏi giỏ, điểm đánh giá sản phẩm, số lượng sản phẩm đã bán, giá tiền sản phẩm, mức giảm giá,... các giá trị được chuyển đổi về kiểu float64 và reshape thành dạng vector cột. Một số đặc trưng có độ biến thiên lớn như tổng tiền mua hàng hoặc giá tiền sản phẩm được áp dụng phép biến đổi logarit nhằm làm giảm sự lệch phân phối và giúp mô hình học hiệu quả hơn, tránh tình trạng bị chi phối bởi các giá trị ngoại lệ.

Đối với các đặc trưng dạng phân loại, như nghề nghiệp người dùng, loại sản phẩm, mùa phát hành sản phẩm, màu sắc, thương hiệu, phong cách sản phẩm và size, các giá trị dạng text được mã hóa thành số thông qua một phương pháp mã hóa nội bộ (encryption). Việc này giúp biến đổi các chuỗi ký tự thành dạng số học, từ đó mô hình có thể tiếp nhận và xử lý được dễ dàng hơn.

Bên cạnh đó, với các đặc trưng dạng văn bản tự do như lịch sử tìm kiếm của người dùng, tên sản phẩm, mô tả sản phẩm, các bình luận đánh giá sản phẩm, và tên sản phẩm mua lần gần nhất, dữ liệu được tiền xử lý bằng cách token hóa. Quá trình token hóa này sẽ tách nhỏ nội dung văn bản thành những đơn vị đặc trưng hơn (như từ khóa chính) để giảm độ phức tạp của dữ liệu văn bản, đồng thời giữ lại được ý nghĩa chính yếu cần thiết cho mô hình học.

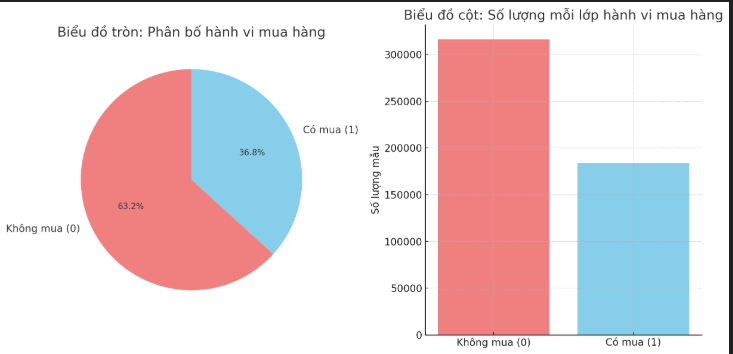
Sau khi xử lý riêng từng loại đặc trưng, dữ liệu được gom nhóm thành từng khối có ý nghĩa logic. Các đặc trưng về thông tin người dùng như tuổi, tháng sinh, nghề nghiệp và giới tính được gom thành một nhóm. Các đặc trưng mô tả hành vi mua sắm của người dùng trong thời gian gần đây, bao gồm tổng tiền mua, số lần click, lịch sử tìm kiếm, số lần mua, số lần hủy, số lần trả hàng, số lần bỏ giỏ sản phẩm, cũng được gom thành một nhóm riêng. Tương tự, tất cả các đặc trưng về sản phẩm như tên, loại, mùa phát hành, màu sắc, thương hiệu, phong cách và size cũng được nhóm lại với nhau để phản ánh đặc điểm sản phẩm một cách đầy đủ. Các thông tin liên quan đến đánh giá sản phẩm, gồm điểm số và các bình luận, được gộp thành một nhóm khác nhằm cung cấp góc nhìn từ trải nghiệm của khách hàng. Ngoài ra, các đặc trưng về tổng số lượng sản phẩm đã mua, giá tiền, và mức giảm giá cũng được xử lý cùng nhau để mô hình nhận biết giá trị sản phẩm mà người dùng quan tâm. Mô tả sản phẩm được tách riêng thành một khối độc lập vì tính chất đặc thù của văn bản dài. Cuối cùng, những hành vi mua gần nhất như tháng mua hàng, tên sản phẩm mua gần nhất và lượng sản phẩm đã bán ra cũng được gom thành một nhóm nhỏ riêng biệt.

Sau khi đã phân nhóm và xử lý đầy đủ, toàn bộ dữ liệu đặc trưng được tập hợp lại thành một tuple để chuẩn bị đưa vào mô hình. Trước khi sử dụng cho huấn luyện, toàn bộ dữ liệu và nhãn (label) được xáo trộn đồng bộ thông qua hàm shuffle, đảm bảo rằng dữ liệu không bị ảnh hưởng bởi bất kỳ sự sắp xếp cố định nào từ file gốc. Việc xáo trộn này là rất quan trọng vì nó giúp loại bỏ bias do thứ tự dữ liệu có thể gây ra và đảm bảo rằng quá trình huấn luyện của mô hình sẽ diễn ra công bằng và khách quan hơn.

Với quy trình xử lý kỹ lưỡng như vậy, dữ liệu đầu vào đảm bảo độ sạch sẽ, tính thống nhất và phù hợp cho việc đưa vào các mô hình Machine Learning hoặc Deep Learning phức tạp về sau.

### Mô tả phân bố dữ liệu:

Tập dữ liệu bao gồm hai nhãn với phân bố không đồng đều. Trong tổng số 500,000 mẫu, có 316,234 mẫu thuộc nhãn 0, chiếm khoảng 63.2%, và 183,766 mẫu thuộc nhãn 1, chiếm khoảng 36.8%. Biểu đồ tròn cho thấy tỷ lệ giữa hai nhãn một cách trực quan, với phần lớn diện tích thuộc về nhãn 0. Trong khi đó, biểu đồ cột thể hiện rõ sự chênh lệch về số lượng, khi nhãn 0 gần gấp đôi nhãn 1. Đây là một ví dụ điển hình của dữ liệu bất cân bằng, khi một lớp xuất hiện với tần suất cao hơn đáng kể so với lớp còn lại.



Hình IV‑1 Sơ đồ phân bố dữ liệu

### Đánh giá:

Bộ dữ liệu thể hiện sự mất cân bằng rõ rệt giữa hai nhãn, trong đó nhãn 0 chiếm tỷ lệ lớn hơn đáng kể so với nhãn 1. Cụ thể, khoảng 63.2% số mẫu thuộc về nhãn 0, trong khi chỉ có 36.8% thuộc về nhãn 1. Sự chênh lệch này cho thấy hành vi được gán cho nhãn 0 xảy ra phổ biến hơn gần gấp đôi so với nhãn còn lại. Mặc dù tổng số mẫu khá lớn, nhưng phân bố không đồng đều như vậy có thể ảnh hưởng đến độ đại diện của dữ liệu, đặc biệt là với những phân tích hoặc mô hình nhạy cảm với sự cân bằng nhãn. Việc nhận diện và hiểu rõ đặc điểm phân bố như thế này là bước quan trọng để đưa ra các chiến lược tiền xử lý hoặc đánh giá kết quả một cách khách quan và chính xác hơn.

### Phân tích rủi ro tiềm ẩn từ phân bố dữ liệu:

Phân bố dữ liệu bất cân bằng như hiện tại tiềm ẩn nhiều rủi ro trong quá trình xử lý và khai thác thông tin. Với tỷ lệ mẫu thuộc nhãn 0 chiếm hơn 63%, hệ thống có xu hướng nghiêng về việc phản ánh đặc trưng của nhóm này nhiều hơn, trong khi nhóm nhãn 1 – vốn ít hơn gần một nửa – có thể bị xem nhẹ hoặc mô tả không đầy đủ. Điều này có thể dẫn đến hiện tượng thiên lệch trong các kết quả phân tích hoặc mô hình học máy, khi hệ thống ưu tiên học và tối ưu cho nhãn chiếm đa số mà bỏ qua hoặc đánh giá sai các đặc trưng của nhãn thiểu số.

Một rủi ro khác là việc đánh giá hiệu suất có thể bị đánh lừa nếu sử dụng các chỉ số như accuracy, vốn dễ bị ảnh hưởng bởi sự chênh lệch nhãn. Trong trường hợp cực đoan, một mô hình chỉ cần dự đoán toàn bộ là nhãn 0 vẫn có thể đạt độ chính xác cao, nhưng lại hoàn toàn vô dụng trong việc nhận diện nhãn 1. Ngoài ra, nếu dữ liệu phản ánh hành vi thực tế, sự mất cân đối này cũng đặt ra câu hỏi về tính đại diện của tập mẫu: liệu nhãn 1 có đang bị thu thập thiếu hay thực sự hiếm gặp?

Do đó, khi làm việc với bộ dữ liệu này, cần thận trọng trong cả giai đoạn tiền xử lý, huấn luyện và đánh giá, nhằm giảm thiểu rủi ro sai lệch và đảm bảo các kết luận rút ra không bị chi phối bởi sự mất cân bằng của phân bố nhãn.

## Xây dựng mô hình

Trong quá trình xây dựng mô hình cho bài toán dự đoán hành vi người dùng, tôi đã thiết kế một kiến trúc mạng học sâu, trong đó kết hợp nhiều kỹ thuật hiện đại nhằm khai thác tối đa thông tin từ dữ liệu đầu vào. Kiến trúc tổng thể được chia thành bốn khối chính: Embedding, Position Embedding, Encode (Transformer Encoder Block) và Fully Connected Output. Toàn bộ quy trình thiết kế được mô tả cụ thể như sau:

### Khối Embedding đầu vào

Mô hình được xây dựng để xử lý 7 nhóm dữ liệu đầu vào riêng biệt. Mỗi nhóm đầu vào có thể bao gồm các loại dữ liệu khác nhau như dữ liệu phân loại (categorical), dữ liệu số (numerical), hoặc dữ liệu tuần tự (sequential). Để xử lý sự đa dạng này, tôi đã quyết định áp dụng một lớp Embedding độc lập cho từng nhóm dữ liệu.

Đối với mỗi đầu vào:

* Trước tiên, dữ liệu sẽ đi qua một lớp Embedding, nhằm chuyển đổi các giá trị rời rạc thành các vector có chiều liên tục trong không gian tiềm ẩn (latent space).
* Sau đó, output từ mỗi lớp Embedding sẽ tiếp tục được đưa qua một lớp Dense (Fully Connected Layer) mà không sử dụng activation. Mục tiêu ở bước này là điều chỉnh kích thước đầu ra embedding sao cho đồng nhất về số chiều giữa các nhóm dữ liệu, tạo điều kiện thuận lợi cho quá trình ghép nối tiếp theo.
* Sau khi các đầu vào đã được embedding và biến đổi về cùng kích thước, tôi tiến hành ghép nối (Concatenate) tất cả các luồng đầu vào theo trục đặc trưng (axis=1).

Bước này tạo ra một tensor duy nhất, đại diện cho toàn bộ thông tin kết hợp từ các nguồn dữ liệu khác nhau. Việc ghép nối như vậy cho phép mô hình có cái nhìn tổng thể và khai thác mối quan hệ giữa các loại dữ liệu ngay từ giai đoạn đầu tiên.

### Khối Position Embedding

Sau khi các đặc trưng được kết hợp, tôi áp dụng một lớp Position Embedding. Ý tưởng sử dụng Position Embedding bắt nguồn từ đặc điểm của Transformer: mặc dù các lớp Attention rất mạnh trong việc học mối liên hệ giữa các token (các đặc trưng), nhưng bản thân Attention lại không tự biết vị trí tương đối của các token trong chuỗi.

Để giải quyết vấn đề này, tôi thêm vào tensor đầu ra một embedding vị trí học được (learnable position embedding), giúp mô hình hiểu và phân biệt được vị trí của từng đặc trưng trong tensor đầu vào. Đây là một yếu tố quan trọng nhằm bảo toàn tính thứ tự, đặc biệt khi dữ liệu có cấu trúc tuần tự hoặc có mối quan hệ phụ thuộc theo thứ tự.

### Khối Encode (Transformer Encoder Block)

Tiếp theo, tensor sau Position Embedding sẽ được đưa vào khối Encode, nơi tôi triển khai một Transformer Encoder Block với đầy đủ các thành phần:

Multi-Head Attention: Lớp Attention này cho phép mô hình học cách tập trung vào những đặc trưng quan trọng nhất đối với mỗi điểm dữ liệu, thông qua nhiều "đầu" Attention song song. Nhờ đó, mô hình có khả năng học các loại mối quan hệ khác nhau giữa các đặc trưng, từ đơn giản đến phức tạp.

Residual Connection và Layer Normalization: Sau mỗi lớp Attention, tôi sử dụng cơ chế cộng thêm đầu vào ban đầu (residual) và chuẩn hóa (normalization) để ổn định dòng gradient, đẩy nhanh quá trình huấn luyện và cải thiện độ chính xác.

Feed-Forward Network (FNN): Một mạng neural nhỏ với activation ReLU được chèn vào sau lớp Attention. Mạng này đóng vai trò học các biểu diễn phi tuyến, giúp mô hình có khả năng mô hình hóa các mối quan hệ phức tạp trong dữ liệu.

Residual Connection và Layer Normalization (lần 2): Tiếp tục sử dụng các cơ chế cộng và chuẩn hóa sau khối Feed-Forward nhằm duy trì tính ổn định cho toàn bộ khối Encode.

Việc sử dụng kiến trúc Transformer cho phép mô hình của tôi tận dụng tối đa sức mạnh của Attention trong việc trích xuất và khai thác mối quan hệ giữa các đặc trưng, thay vì đơn thuần phụ thuộc vào các phép biến đổi tuyến tính như mạng Dense thông thường.

### Khối Fully Connected Output

Sau khi tensor đi qua khối Encode, tôi tiếp tục đưa nó qua một lớp Fully Connected Network (Dense) với activation ReLU. Mục tiêu của lớp này là nén và tái trích xuất thêm các đặc trưng sâu từ output của Encoder, nhằm chuẩn bị tốt nhất cho giai đoạn ra quyết định cuối cùng.

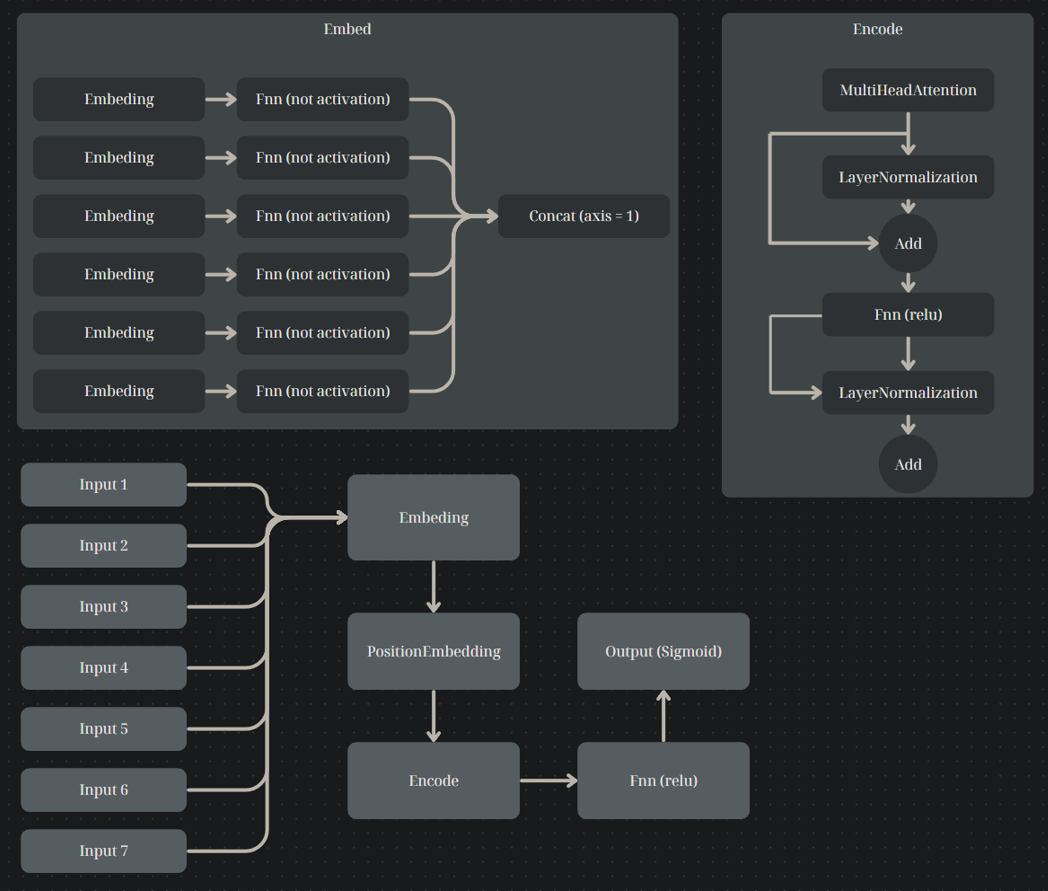
Cuối cùng, đầu ra của lớp Dense sẽ được đưa vào một lớp Output với activation Sigmoid. Lựa chọn activation Sigmoid phù hợp với yêu cầu của bài toán binary classification (có mua / không mua), bởi vì Sigmoid sẽ biến đầu ra thành một giá trị xác suất nằm trong khoảng từ 0 đến 1.

### Tổng kết

Tóm lại, mô hình mà tôi xây dựng là sự kết hợp hài hòa giữa các thành phần:

* Embedding mạnh mẽ cho đầu vào đa dạng.
* Position Embedding giúp mô hình nhận biết vị trí.
* Khối Encode dựa trên Transformer để học sâu mối quan hệ giữa đặc trưng.
* Các tầng Dense để làm giàu biểu diễn đặc trưng và đưa ra dự đoán chính xác.

Cách thiết kế này cho phép tôi xử lý dữ liệu đầu vào phong phú và phức tạp, đồng thời khai thác được những mối liên hệ ngầm giữa các yếu tố trong quá trình dự đoán hành vi người dùng. Kiến trúc này cũng có tính mở rộng cao, dễ dàng tinh chỉnh thêm các khối Attention hoặc Feed-Forward nếu cần thiết để phù hợp với các bài toán có độ phức tạp lớn hơn trong tương lai.



Hình IV‑2 kiến trúc mô hình thực tế

## Hàm loss, hàm đo độ chính xác

Trong quá trình xây dựng mô hình, việc lựa chọn hàm mất mát và các chỉ số đánh giá (metrics) đóng vai trò then chốt, quyết định trực tiếp đến chất lượng huấn luyện và khả năng tổng quát của mô hình trên dữ liệu thực tế. Sau khi cân nhắc kỹ lưỡng về đặc thù bài toán và bản chất dữ liệu, tôi đã tiến hành biên dịch mô hình (compile) với các thành phần như sau:

Cụ thể, tôi sử dụng một hàm mất mát (loss) phù hợp với bài toán phân loại nhị phân. Hàm mất mát là yếu tố đầu tiên và quan trọng nhất, có nhiệm vụ đo lường độ sai lệch giữa giá trị dự đoán của mô hình và giá trị thực tế từ dữ liệu. Trong quá trình huấn luyện, mô hình liên tục tối thiểu hóa giá trị hàm mất mát này thông qua thuật toán tối ưu hóa (optimizer), nhằm điều chỉnh trọng số các tham số bên trong mạng nơ-ron theo hướng dự đoán chính xác hơn. Việc lựa chọn đúng hàm loss không chỉ đảm bảo quá trình học tập ổn định mà còn giúp mô hình hội tụ nhanh chóng và đạt được kết quả tốt.

Song song với hàm mất mát, tôi cũng thiết lập một tập hợp các chỉ số đánh giá (metrics) nhằm theo dõi hiệu suất mô hình một cách toàn diện hơn trong suốt quá trình huấn luyện và kiểm tra. Các chỉ số mà tôi lựa chọn bao gồm:

Độ chính xác nhị phân (BinaryAccuracy): Chỉ số này đo lường tỷ lệ mẫu dự đoán đúng so với tổng số mẫu đầu vào. Trong bối cảnh bài toán phân loại nhị phân, BinaryAccuracy cho phép tôi đánh giá một cách tổng quát về khả năng phân biệt giữa hai lớp của mô hình. Tuy nhiên, trong trường hợp dữ liệu mất cân bằng, độ chính xác đơn thuần đôi khi không phản ánh đúng thực chất hiệu quả mô hình.

Độ nhạy (Recall): Recall đo lường khả năng phát hiện đúng các mẫu thực sự thuộc lớp dương tính (positive class). Tôi quyết định đưa chỉ số này vào vì muốn đảm bảo rằng mô hình có thể hạn chế tối đa việc bỏ sót các mẫu quan trọng. Một Recall cao đồng nghĩa với việc tỷ lệ false negative thấp, điều này đặc biệt cần thiết nếu các trường hợp dương tính có ý nghĩa lớn hoặc rủi ro cao trong bối cảnh ứng dụng thực tế.

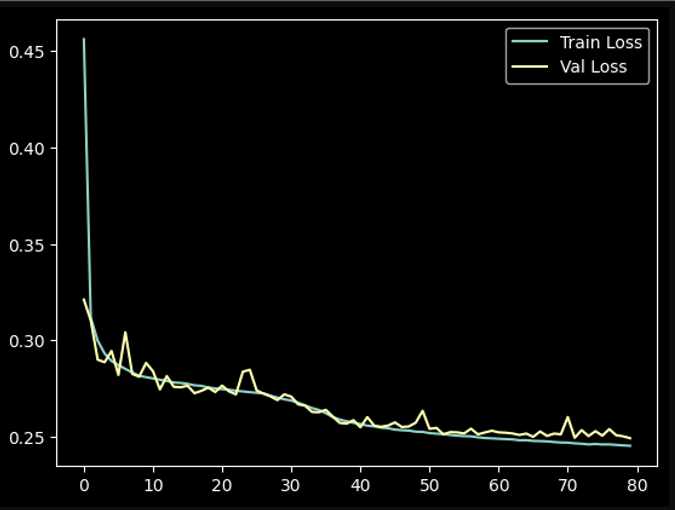
Độ chính xác (Precision): Precision phản ánh tỷ lệ dự đoán dương tính chính xác trên tổng số mẫu được mô hình gán nhãn dương tính. Khác với Recall, Precision tập trung vào việc giảm thiểu số lượng false positive. Việc theo dõi Precision song song với Recall giúp tôi đảm bảo rằng mô hình không chỉ tìm kiếm được nhiều mẫu dương tính, mà còn tìm kiếm một cách chính xác, hạn chế tối đa việc dự đoán sai lệch.

F1 Score: Đây là chỉ số trung bình điều hòa giữa Precision và Recall, mang lại một cái nhìn cân bằng và tổng quát về hiệu suất mô hình, nhất là trong trường hợp dữ liệu không đồng đều giữa các lớp. Một mô hình với F1 Score cao thể hiện rằng nó vừa có khả năng phát hiện đầy đủ các mẫu dương tính (Recall cao), vừa duy trì được độ chính xác trong dự đoán (Precision cao). Do đó, tôi sử dụng F1 Score như một tiêu chí quan trọng để tối ưu hiệu quả mô hình một cách toàn diện.

Việc lựa chọn đồng thời nhiều chỉ số đánh giá như trên xuất phát từ mong muốn xây dựng một mô hình không chỉ đạt hiệu suất tốt trên tập huấn luyện, mà còn có tính ổn định, đáng tin cậy khi triển khai thực tế. Mỗi chỉ số mang một góc nhìn riêng, phản ánh các khía cạnh khác nhau của hiệu suất mô hình. Thông qua việc giám sát tổng hợp nhiều chỉ số trong suốt quá trình huấn luyện, tôi có thể nhận diện sớm các vấn đề như overfitting, bias giữa các lớp, hoặc mất cân bằng giữa Precision và Recall, từ đó đưa ra các biện pháp điều chỉnh kịp thời nhằm tối ưu hóa hiệu quả tổng thể.

Tóm lại, việc biên dịch mô hình với thiết lập bao gồm một hàm mất mát thích hợp và một tập hợp chỉ số đánh giá đa dạng đã giúp tôi kiểm soát chặt chẽ quá trình huấn luyện và đánh giá mô hình, đồng thời đặt nền tảng vững chắc cho việc tối ưu hóa và ứng dụng mô hình vào thực tế sau này.

## Huấn luyện mô hình



Hình IV‑3 Phân bổ loss của tập train và test trong quá trình train

Quá trình huấn luyện của mô hình được thể hiện qua ba biểu đồ loss, cho thấy sự thay đổi của hàm mất mát (loss) theo số epoch từ 0 đến 80. Trong biểu đồ lớn bên trái, đường màu xanh lam nhạt biểu diễn giá trị Train Loss, còn đường màu vàng biểu diễn Validation Loss. Cả hai đường đều bắt đầu với giá trị loss khá cao ở epoch 0 — cụ thể train loss bắt đầu ở mức khoảng 0.45, trong khi validation loss khởi điểm khoảng 0.30. Từ giai đoạn đầu, cả hai chỉ số đều giảm mạnh qua các epoch đầu tiên, đặc biệt là trong 10 epoch đầu, thể hiện tốc độ suy giảm nhanh chóng của hàm mất mát trong giai đoạn đầu huấn luyện.

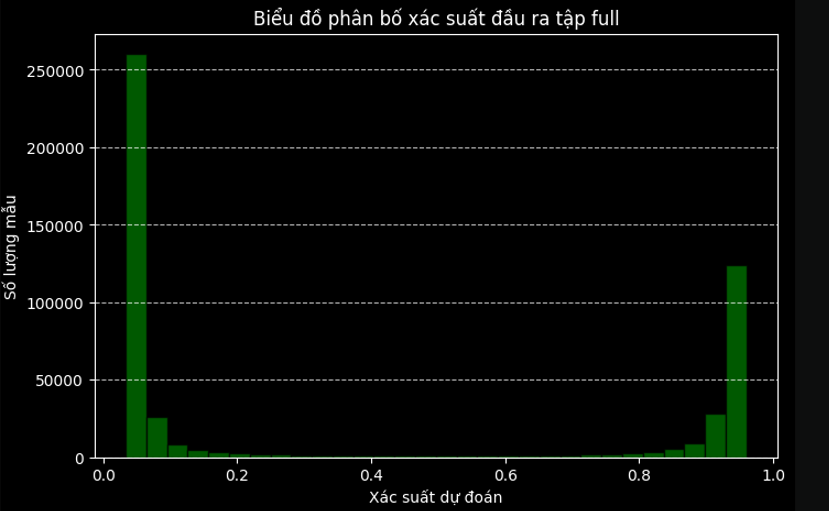
Từ epoch 10 trở đi, tốc độ giảm của cả hai hàm loss dần chậm lại. Train loss tiếp tục giảm đều, dao động nhẹ xung quanh giá trị 0.26–0.25 từ epoch 40 đến epoch 80. Validation loss cũng có xu hướng giảm tương tự nhưng dao động nhiều hơn, với một số đỉnh và đáy rõ rệt xảy ra rải rác, đáng chú ý là các điểm nhô lên tại khoảng epoch 20, 35 và 60, nhưng nhìn chung vẫn nằm trong biên độ từ 0.25 đến 0.30. Từ epoch 60 đến 80, cả hai đường train và val loss đều đi sát nhau và gần như song song, với sự ổn định cao hơn, không còn nhiều biến động rõ rệt.

Biểu đồ bên phải được chia thành hai phần. Phần trên là biểu đồ thể hiện Training Loss theo epoch, với một đường duy nhất màu xanh lá cây. Đường cong loss này cũng khởi đầu tại khoảng 0.45, sau đó giảm nhanh trong giai đoạn 10 epoch đầu tiên, rồi giảm dần đều và mượt mà đến mức khoảng 0.24–0.25 ở cuối giai đoạn huấn luyện. Biểu đồ này phản ánh xu hướng tương tự như trong biểu đồ tổng hợp bên trái, nhưng được thể hiện tách riêng và rõ ràng hơn.

Phần dưới của biểu đồ bên phải thể hiện riêng Validation Loss, với một đường màu xanh lá cây tương tự. Đường biểu diễn này bắt đầu ở mức khoảng 0.32, có nhiều dao động trong giai đoạn 30 epoch đầu tiên, sau đó dao động giảm dần về mức ổn định từ 0.26 đến 0.25 trong các epoch sau. Nhịp dao động có cường độ cao hơn so với train loss, và đường biểu diễn có nhiều đoạn lồi lõm thể hiện sự thay đổi của loss giữa các batch trên tập validation theo từng epoch.

Cả ba biểu đồ thể hiện đồng nhất quá trình huấn luyện kéo dài 80 epoch, với xu hướng chung là hàm mất mát giảm đều và chậm dần qua thời gian, và cho thấy rõ ràng mối tương quan giữa loss trên tập huấn luyện và tập validation. Sự chênh lệch giữa hai đường không quá lớn và duy trì ở mức tương đối ổn định sau khoảng epoch thứ 20. Các giá trị loss cuối cùng đạt được đều hội tụ quanh mức 0.25, và toàn bộ quá trình huấn luyện được minh họa bằng những đường cong mô tả liên tục, không có giai đoạn dừng huấn luyện sớm hoặc gián đoạn.

### Đánh giá phân bố đầu ra



Hình IV‑4 Sự phân bổ đầu ra của mô hình

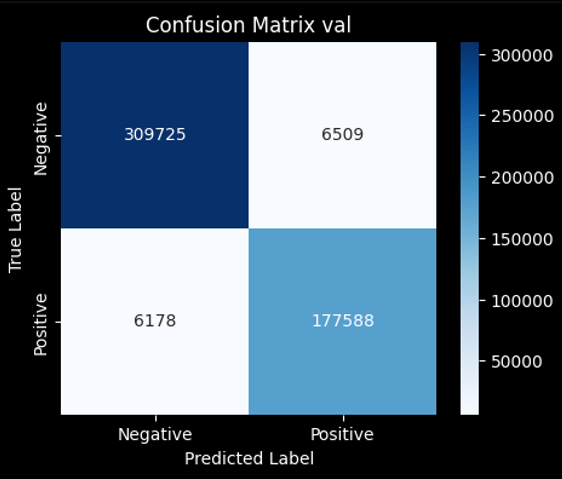
Biểu đồ thể hiện phân bố xác suất đầu ra của mô hình sau khi đã huấn luyện hoàn tất trên toàn bộ tập dữ liệu (tập full). Đây là biểu đồ histogram với trục hoành là xác suất dự đoán (probability) nằm trong khoảng từ 0 đến 1, và trục tung biểu diễn số lượng mẫu tương ứng với mỗi khoảng giá trị xác suất. Tên biểu đồ ghi rõ: “Biểu đồ phân bố xác suất đầu ra tập full”, cho thấy đây là kết quả đầu ra cuối cùng của mô hình sau khi đã trải qua quá trình huấn luyện đầy đủ.

Quan sát trực tiếp từ biểu đồ, có thể thấy phân bố xác suất của mô hình có dạng hai cực rõ rệt (bimodal distribution), thể hiện qua hai cụm thanh cột nổi bật tập trung ở hai đầu của trục hoành:

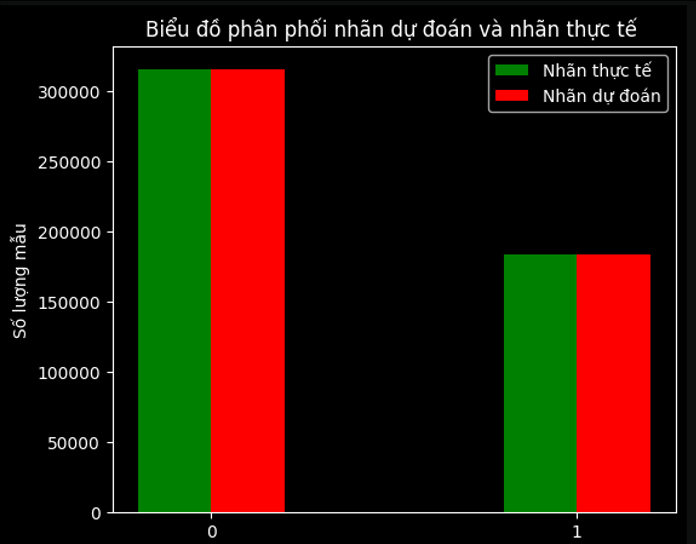
* Cực trái (gần 0): Có một số lượng rất lớn mẫu có xác suất dự đoán gần bằng 0. Cột đầu tiên – ứng với xác suất khoảng từ 0.0 đến 0.05 – cao vượt trội hơn tất cả các cột khác, với trên 250.000 mẫu, cho thấy phần lớn dữ liệu đầu ra của mô hình được dự đoán với xác suất rất thấp. Điều này hàm ý rằng mô hình rất “tự tin” khi dự đoán các mẫu thuộc lớp âm (negative class – nhãn 0).
* Cực phải (gần 1): Có một nhóm mẫu đáng kể có xác suất đầu ra nằm gần 1.0. Trong đó, cột cao nhất ở phía bên phải nằm ở vùng xác suất từ khoảng 0.95 đến 1.0, chứa khoảng gần 100.000 mẫu. Điều này phản ánh rằng mô hình cũng đưa ra các dự đoán rất chắc chắn đối với các mẫu thuộc lớp dương (positive class – nhãn 1).
* Khu vực trung gian (0.2 đến 0.8): Có thể thấy vùng giữa của trục hoành – từ khoảng 0.2 đến 0.8 – gần như trống vắng hoặc có rất ít mẫu rơi vào khoảng này. Các thanh cột trong vùng này rất thấp, sát gần trục hoành, cho thấy mô hình hiếm khi dự đoán các xác suất “mơ hồ” hay không chắc chắn. Điều này là đặc điểm điển hình của một mô hình đã được huấn luyện tốt và có xu hướng đưa ra các dự đoán phân biệt rõ ràng giữa hai lớp.
* Độ phân cực: Tổng thể, biểu đồ thể hiện rõ xu hướng phân cực trong đầu ra của mô hình – hầu hết các xác suất đều tập trung sát 0 hoặc sát 1. Mức độ phân cực này thể hiện rằng mô hình học được sự khác biệt rõ rệt giữa hai nhãn và có khả năng phân loại mạnh mẽ.
* Độ mất cân bằng dữ liệu (ngụ ý): Với số lượng mẫu ở phía gần 0 nhiều hơn đáng kể so với phía gần 1, có thể suy luận (gián tiếp) rằng tập dữ liệu có mất cân bằng nhãn, với phần lớn mẫu thuộc lớp 0 (negative). Dù biểu đồ không trực tiếp thể hiện nhãn thực tế, nhưng sự chênh lệch trong số lượng mẫu dự đoán xác suất thấp và cao gợi ý rằng lớp 0 chiếm ưu thế trong tập dữ liệu.

Tóm lại, biểu đồ phân bố đầu ra thể hiện mô hình có khả năng phân loại xác suất rõ ràng và quyết đoán, với hai cụm xác suất tập trung tại gần 0 và gần 1, đồng thời hầu như không có sự mơ hồ trong dự đoán. Sự phân cực cao cùng với độ chênh lệch về số lượng giữa hai đầu gợi ý rằng dữ liệu đầu vào có sự mất cân bằng về nhãn, nhưng mô hình vẫn có xu hướng học được đặc trưng của hai lớp một cách rõ ràng và hiệu quả.

### Đánh giá ma trận đầu ra và phân bố đầu ra so với nhãn



Hình IV‑5 Ma trận phân bố đầu ra



Hình IV‑6 Biểu đồ phân bố nhãn so với dự đoán

Dựa trên kết quả thể hiện trong hình ảnh, có thể tiến hành phân tích và đánh giá hiệu quả của mô hình một cách chi tiết như sau:

Ma trận nhầm lẫn (confusion matrix) minh họa kết quả phân loại của mô hình trên tập dữ liệu kiểm định (validation) với tổng cộng 500.000 mẫu. Cụ thể, số lượng mẫu được phân loại chính xác bao gồm 309.725 trường hợp thuộc lớp âm tính (Negative) và 177.588 trường hợp thuộc lớp dương tính (Positive). Số lượng dự đoán sai tương ứng là 6.509 trường hợp âm tính bị nhầm sang dương tính (False Positive) và 6.178 trường hợp dương tính bị phân loại sai thành âm tính (False Negative).

Từ các số liệu trên, độ chính xác tổng thể (accuracy) của mô hình đạt xấp xỉ 97,66%, phản ánh mức độ chính xác rất cao trong toàn bộ quá trình phân loại. Xét riêng trên lớp dương tính, độ nhạy (recall) – đại diện cho khả năng nhận diện chính xác các mẫu dương tính – đạt 96,63%. Đồng thời, độ chính xác của mô hình trong việc dự đoán lớp dương tính (precision) là 96,46%, cho thấy tỷ lệ mẫu dương tính được mô hình nhận diện là đúng là rất cao. Ngoài ra, độ đặc hiệu (specificity) – chỉ số phản ánh năng lực phân biệt lớp âm tính – cũng đạt mức 97,95%. Những chỉ số này cho thấy mô hình hoạt động hiệu quả và cân đối đối với cả hai lớp, không thiên lệch rõ rệt về phía nào.

Biểu đồ phân phối nhãn ở bên phải hình ảnh cung cấp cái nhìn trực quan về mối tương quan giữa phân phối nhãn thực tế và nhãn dự đoán. Cụ thể, mô hình dự đoán khoảng 316.000 mẫu thuộc lớp âm tính và 184.000 mẫu thuộc lớp dương tính, gần như trùng khớp với phân phối của nhãn thực tế. Sự tương đồng này cho thấy mô hình không chỉ hoạt động tốt trên từng mẫu riêng lẻ mà còn có khả năng tái tạo phân bố tổng thể của dữ liệu, một đặc điểm quan trọng trong các bài toán phân loại có yêu cầu về tính công bằng và không lệch nhãn.

Tổng thể, các kết quả thu được từ ma trận nhầm lẫn và biểu đồ phân phối nhãn đều chỉ ra rằng mô hình đã đạt được mức độ chính xác cao, với khả năng phân loại hiệu quả cả hai lớp và mức độ sai lệch tối thiểu. Phân bố đầu ra phù hợp với phân bố nhãn thực tế cũng phản ánh khả năng tổng quát hóa tốt của mô hình. Đây là những yếu tố cho thấy mô hình đã được huấn luyện hiệu quả và tối ưu, và có thể sẵn sàng triển khai cho các ứng dụng thực tế

## Xây dựng ứng dụng web

Trang web mà dự án xây dựng là một nền tảng thương mại điện tử chuyên cung cấp các sản phẩm thời trang, sử dụng kiến trúc microservice cho backend, một server React cho frontend và SQL Server làm cơ sở dữ liệu. Kiến trúc microservice cho phép hệ thống chia nhỏ các chức năng thành các dịch vụ độc lập, mỗi dịch vụ có thể được phát triển, triển khai và bảo trì riêng biệt, đồng thời đảm bảo khả năng mở rộng và linh hoạt của hệ thống.

Frontend của ứng dụng được phát triển bằng React, giúp cung cấp giao diện người dùng tương tác, dễ sử dụng và hiệu quả trong việc quản lý trạng thái của các thành phần UI. SQL Server được sử dụng để lưu trữ dữ liệu liên quan đến người dùng, sản phẩm và giao dịch, hỗ trợ các truy vấn dữ liệu phức tạp và đảm bảo tính ổn định và hiệu suất của hệ thống.

Đặc biệt, hệ thống tích hợp một mô hình AI dựa trên các thuật toán học máy, nhằm cung cấp các đề xuất sản phẩm được cá nhân hóa cho người dùng. Các dữ liệu hành vi người dùng, như lịch sử tìm kiếm, số lần nhấp chuột, và lịch sử mua sắm, được thu thập và sử dụng để cải thiện chất lượng của các đề xuất sản phẩm, giúp tăng cường trải nghiệm người dùng và tối ưu hóa khả năng chuyển đổi bán hàng.

### Xây dựng database

Cơ sở dữ liệu được thiết kế với mục tiêu hỗ trợ toàn diện hệ thống quản lý bán hàng trực tuyến trong lĩnh vực thời trang, nhằm đảm bảo khả năng mở rộng, tích hợp và vận hành hiệu quả trên nền tảng thương mại điện tử hiện đại. Hệ thống cơ sở dữ liệu bao gồm các bảng phục vụ quản lý thông tin người dùng, sản phẩm, đơn hàng, giỏ hàng, hành vi tiêu dùng, và các yếu tố liên quan đến quá trình mua sắm trực tuyến.

Thông tin sản phẩm được tổ chức chi tiết theo các thuộc tính chuyên biệt như loại sản phẩm, thương hiệu, chất liệu, màu sắc, kích thước, mùa phát hành và phong cách. Các bảng dữ liệu còn cho phép theo dõi mức tồn kho ở từng biến thể sản phẩm (phân theo size và màu), phục vụ cho việc quản lý hàng hóa chính xác. Hệ thống người dùng phân cấp rõ ràng giữa người mua, nhân viên và quản trị viên, kết hợp với các bảng phân quyền để đảm bảo an toàn và kiểm soát truy cập.

Đơn hàng và giỏ hàng được quản lý thông qua các bảng chuyên biệt, phản ánh chi tiết trạng thái giao dịch, phương thức thanh toán và địa chỉ giao hàng. Ngoài ra, cơ sở dữ liệu còn lưu trữ lịch sử tìm kiếm và hành vi người dùng, phục vụ cho việc phân tích hành vi và tích hợp các mô hình trí tuệ nhân tạo như hệ thống gợi ý sản phẩm. Thông tin đánh giá sản phẩm bao gồm cả điểm số, bình luận và thời điểm bình luận nhằm nâng cao trải nghiệm người dùng và hỗ trợ quản lý phản hồi.

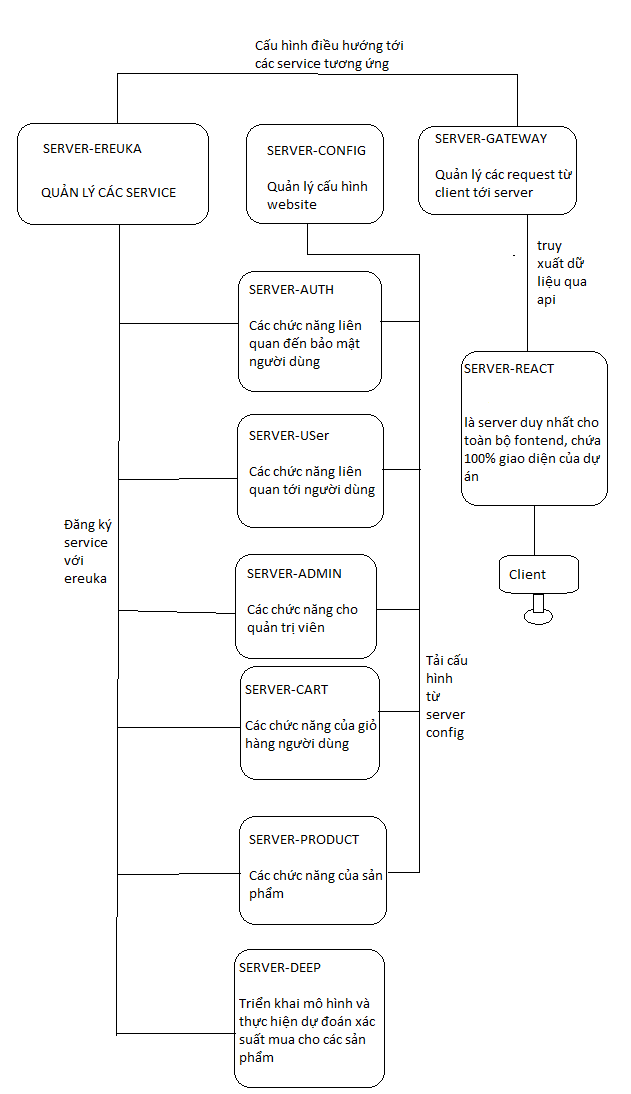
Hầu hết các bảng đều có cột xóa mềm ("day\_delete\_table" hoặc biến thể), nhằm hỗ trợ khả năng truy vết và phục hồi dữ liệu khi cần thiết. Tuy nhiên, cần chuẩn hóa tên cột xóa mềm để đảm bảo tính nhất quán (ví dụ: sử dụng thống nhất "deleted\_at").

Một số điểm cần cải thiện bao gồm:

* Khắc phục lỗi chính tả trong tên bảng và cột (ví dụ: "\_seasion", "\_coment", "\_desription", "day\_delete\_tabel").
* Bổ sung các ràng buộc dữ liệu (NOT NULL, CHECK, UNIQUE) để đảm bảo tính toàn vẹn và độ tin cậy.
* Làm rõ chức năng của các bảng chưa hoàn thiện hoặc chưa có cấu trúc cụ thể (ví dụ: bảng "\_feature\_product").

Tổng thể, cơ sở dữ liệu đã bao phủ đầy đủ các yêu cầu nghiệp vụ và sẵn sàng cho việc mở rộng sang các chức năng nâng cao như phân tích dữ liệu, cá nhân hóa sản phẩm, và tích hợp AI vào quy trình bán hàng.

### Xây dựng kiến trúc micro service



Hình IV‑7 Kiến trúc microservice

Kiến trúc microservice của hệ thống được thiết kế gồm tổng cộng 9 server, chia thành hai nhóm chính: nhóm quản lý hạ tầng và nhóm các dịch vụ chức năng.

Nhóm quản lý hạ tầng bao gồm 3 server:

* Config Server: chịu trách nhiệm lưu trữ và phân phối cấu hình trung tâm cho các service, giúp đảm bảo tính nhất quán và khả năng cập nhật cấu hình mà không cần khởi động lại hệ thống.
* Eureka Server: đóng vai trò như một service registry, nơi các dịch vụ có thể đăng ký và truy vấn thông tin lẫn nhau, hỗ trợ cơ chế tự động phát hiện dịch vụ (service discovery).
* Gateway Server: là điểm vào duy nhất của toàn bộ hệ thống, xử lý định tuyến các yêu cầu từ phía client đến các microservice tương ứng, đồng thời có thể tích hợp cơ chế bảo mật và giới hạn truy cập.

Nhóm các dịch vụ chức năng gồm 6 server, mỗi server đảm nhiệm một chức năng riêng biệt trong hệ thống:

* User Service: quản lý thông tin người dùng, bao gồm đăng ký, cập nhật hồ sơ và lưu trữ lịch sử hoạt động.
* Product Service: quản lý thông tin sản phẩm thời trang như danh mục, mô tả, giá cả, tồn kho và thuộc tính sản phẩm.
* Cart Service: xử lý chức năng giỏ hàng, bao gồm thêm, cập nhật và xóa sản phẩm trong giỏ hàng của người dùng.
* Auth Server: chịu trách nhiệm xác thực và phân quyền truy cập, quản lý token và đảm bảo an toàn cho các giao dịch.
* Admin Service: admin\_service là service quản trị trung tâm, chịu trách nhiệm quản lý người dùng, sản phẩm và hệ thống. Service này được xây dựng bằng Spring Boot, tích hợp chặt chẽ với các service khác để giám sát và điều phối hoạt động. Nó cũng hỗ trợ theo dõi và kiểm soát mô hình AI thông qua kết nối với deep\_service.
* Deep Service: là dịch vụ chuyên biệt tích hợp mô hình AI (Collaborative Filtering kết hợp Transformer) để xử lý và cung cấp các đề xuất sản phẩm dựa trên hành vi người dùng.

Toàn bộ hệ thống hoạt động theo mô hình phân tán, trong đó các service giao tiếp thông qua các API nội bộ, đảm bảo tính linh hoạt, dễ mở rộng và tối ưu hóa hiệu suất xử lý trong môi trường thực thi độc lập.

#### Server config quản lý cấu hình chung cho dự án

Trong kiến trúc microservice được triển khai, Config Server giữ vai trò trung tâm trong việc quản lý cấu hình tập trung cho toàn bộ hệ thống. Việc tách biệt cấu hình khỏi mã nguồn là một nguyên tắc quan trọng trong thiết kế hệ thống phân tán, nhằm tăng cường khả năng bảo trì, mở rộng và kiểm soát cấu hình theo từng môi trường triển khai. Để đáp ứng yêu cầu đó, hệ thống sử dụng Spring Cloud Config Server kết hợp với GitHub làm nơi lưu trữ cấu hình. Cấu hình của từng service được tổ chức thành các file YAML hoặc properties, được phân chia theo tên service và theo từng profile môi trường như dev, test, hoặc prod, giúp dễ dàng quản lý và tùy chỉnh thông số cấu hình theo nhu cầu thực tế của từng giai đoạn vận hành.

Cơ chế hoạt động của Config Server bao gồm hai thành phần chính: phía server và phía client. Trong đó, phía server được xây dựng bằng Spring Boot và Spring Cloud Config, có nhiệm vụ kết nối với kho lưu trữ GitHub để truy xuất cấu hình. Khi khởi động, Config Server sẽ tải toàn bộ cấu hình từ GitHub về bộ nhớ tạm thời và cung cấp giao diện RESTful API để các service phía client có thể gọi và lấy cấu hình tương ứng. Mỗi microservice trong hệ thống sẽ được cấu hình như một Spring Cloud Config Client, và khi khởi động, các service này sẽ gửi yêu cầu HTTP đến Config Server để nhận cấu hình tương ứng dựa trên application name và profile.

Lợi ích của mô hình này là cấu hình có thể được thay đổi trực tiếp trên GitHub mà không cần biên dịch lại mã nguồn hay triển khai lại dịch vụ. Thêm vào đó, Spring Cloud Config hỗ trợ cơ chế hot reload khi tích hợp với Spring Cloud Bus và message broker như RabbitMQ hoặc Kafka, cho phép các service tự động nhận cấu hình mới khi có thay đổi mà không cần khởi động lại, đảm bảo tính liên tục của dịch vụ.

Với cách tiếp cận này, hệ thống có thể mở rộng cấu hình linh hoạt, tích hợp CI/CD dễ dàng, đảm bảo an toàn với cơ chế phân quyền truy cập GitHub, đồng thời nâng cao tính nhất quán và khả năng kiểm soát cấu hình trong các hệ thống microservice phức tạp.

#### Server ereuka quản lý các service

Trong kiến trúc microservice, Eureka Server đóng vai trò như một service registry – một thành phần trung tâm trong việc quản lý thông tin các dịch vụ trong hệ thống. Trong môi trường phân tán, nơi mà các service có thể khởi động hoặc ngừng hoạt động bất kỳ lúc nào, việc tìm kiếm và kết nối giữa các service trở nên phức tạp nếu không có một cơ chế khám phá dịch vụ tự động (service discovery). Spring Cloud Netflix Eureka được sử dụng để giải quyết vấn đề này, đảm bảo rằng các dịch vụ có thể đăng ký và phát hiện lẫn nhau một cách linh hoạt, hiệu quả và hoàn toàn tự động.

Về cơ chế hoạt động, Eureka Server hoạt động theo mô hình client-server. Phía Eureka Server được triển khai như một ứng dụng Spring Boot đơn giản, có chức năng duy trì danh sách các service đã đăng ký. Mỗi khi một service khởi động (ví dụ như user\_service, product\_service, v.v.), nó sẽ gửi một yêu cầu HTTP (REST call) đến Eureka Server để đăng ký thông tin của mình như tên service, địa chỉ IP, port, thời gian sống (lease time), và các metadata khác. Sau đó, định kỳ (thường là mỗi 30 giây), service sẽ gửi các gói heartbeat để thông báo rằng nó vẫn đang hoạt động. Nếu Eureka không nhận được heartbeat trong một khoảng thời gian nhất định (mặc định là 90 giây), service đó sẽ bị loại khỏi registry, giúp đảm bảo tính nhất quán và độ tin cậy của hệ thống.

Các service không chỉ đăng ký vào Eureka mà còn có thể tra cứu thông tin của các service khác bằng cách truy cập API của Eureka. Nhờ vậy, các service có thể gọi nhau thông qua tên logic thay vì IP hoặc URL cứng, từ đó loại bỏ sự phụ thuộc vào thông tin hạ tầng cụ thể. Ví dụ, thay vì gọi trực tiếp địa chỉ IP của product\_service, một service khác chỉ cần gọi "http://product-service" và để phần lookup do Eureka xử lý, thường được kết hợp với RestTemplate, WebClient, hoặc tích hợp sâu hơn với Spring Cloud LoadBalancer để thực hiện cân bằng tải nội bộ.

Việc tích hợp Eureka Server vào hệ thống rất đơn giản trong Spring Boot. Eureka Server được khai báo bằng cách sử dụng annotation @EnableEurekaServer trong lớp main và cấu hình trong application.yml hoặc application.properties.

Cấu hình này biểu thị rằng đây là Eureka Server, không tự đăng ký chính nó và không cần lấy danh sách từ server khác (trong trường hợp kiến trúc chỉ có một server). Trong các hệ thống quy mô lớn, có thể triển khai nhiều instance của Eureka để tạo thành một cụm (cluster) và hỗ trợ failover.

Nhờ vào Eureka, toàn bộ hệ thống microservice trở nên linh hoạt, có khả năng tự động mở rộng, và hoạt động ổn định ngay cả trong môi trường thay đổi thường xuyên về mặt tài nguyên hoặc mạng. Đây là một thành phần cốt lõi giúp hiện thực hóa khả năng giao tiếp nội bộ hiệu quả và đáng tin cậy giữa các dịch vụ trong hệ thống.

#### Server Gateway (router)

Trong kiến trúc microservice, Gateway Server đóng vai trò là cổng vào duy nhất (API Gateway) cho toàn bộ hệ thống backend. Đây là nơi tiếp nhận mọi yêu cầu từ phía frontend hoặc client bên ngoài, sau đó thực hiện định tuyến (routing) đến các service nội bộ như user\_service, product\_service, cart\_service, v.v. Gateway giúp trừu tượng hóa toàn bộ hệ thống phía sau nó, từ đó đơn giản hóa luồng giao tiếp, tăng cường bảo mật, và hỗ trợ các tính năng mở rộng như kiểm soát truy cập, phân tích log, giới hạn tốc độ, xác thực, và cân bằng tải.

Spring Cloud Gateway được sử dụng để xây dựng Gateway Server trong hệ thống. Gateway hoạt động như một reverse proxy, nơi mỗi request được xử lý thông qua một chuỗi các filter và sau đó được định tuyến đến service đích dựa vào các quy tắc được định nghĩa trong file cấu hình. Các filter có thể được sử dụng để thực hiện các nhiệm vụ như thêm/xóa header, xác thực JWT token, kiểm tra CORS, hoặc ghi log.

Trong cấu hình trên, mọi yêu cầu bắt đầu bằng /api/users/\*\* sẽ được định tuyến đến user-service thông qua service discovery (Eureka), nhờ vào tiền tố lb://. Bộ lọc StripPrefix=1 sẽ loại bỏ /api trước khi chuyển đến service đích.

Một trong những điểm mạnh của Gateway là khả năng tích hợp với Eureka Server, cho phép định tuyến động dựa vào danh sách service hiện có trong registry mà không cần khai báo cứng địa chỉ IP hoặc hostname. Điều này giúp hệ thống trở nên linh hoạt và có khả năng mở rộng cao, đặc biệt khi số lượng service ngày càng tăng.

Ngoài định tuyến cơ bản, Gateway còn có thể kết hợp với các module như Spring Security để thực hiện xác thực, hoặc tích hợp rate limiter bằng Redis để giới hạn số lượng request nhằm bảo vệ các service khỏi bị quá tải. Các yêu cầu không hợp lệ, ví dụ như thiếu token hoặc sai định dạng, có thể được chặn ngay tại Gateway trước khi đến các service phía sau, giúp tiết kiệm tài nguyên và tăng độ an toàn của hệ thống.

Tóm lại, Gateway Server là một thành phần không thể thiếu trong hệ thống microservice hiện đại. Nó không chỉ đảm nhiệm chức năng điều phối request mà còn giúp tổ chức hệ thống trở nên gọn gàng, dễ kiểm soát và dễ mở rộng, đồng thời đóng vai trò là lá chắn đầu tiên bảo vệ toàn bộ kiến trúc backend khỏi các rủi ro bảo mật và vận hành.

#### Các server service

Trong hệ thống microservice của dự án, năm service backend chính đóng vai trò xử lý nghiệp vụ cụ thể, đảm bảo sự tách biệt trách nhiệm và khả năng mở rộng độc lập giữa các phần của hệ thống. Các service bao gồm: user\_service, product\_service, cart\_service, auth\_server, và deep\_service, mỗi service được triển khai và vận hành như một đơn vị độc lập, có thể mở rộng và triển khai riêng biệt theo nhu cầu.

User service:

* user\_service chịu trách nhiệm quản lý thông tin người dùng, bao gồm đăng ký, cập nhật hồ sơ, theo dõi hành vi người dùng như số lượt click, tìm kiếm, đánh giá sản phẩm,... Service này cung cấp các API RESTful để frontend có thể giao tiếp và hiển thị thông tin người dùng trên giao diện. Dữ liệu người dùng được lưu trữ trong SQL Server, và service này sử dụng Spring Boot để triển khai với các thành phần như JPA, Service Layer, và DTO để đảm bảo tính sạch của kiến trúc.

Product service:

* product\_service phụ trách quản lý thông tin sản phẩm, bao gồm các thuộc tính như tên, mô tả, thương hiệu, mùa phát hành, giá tiền và lượt bán. Service này cung cấp khả năng tìm kiếm, lọc và phân loại sản phẩm cho frontend. Ngoài ra, các tính năng như thống kê sản phẩm bán chạy hoặc lọc sản phẩm theo đặc trưng thời trang cũng được xử lý tại đây. Tất cả logic nghiệp vụ liên quan đến sản phẩm được gói gọn trong service này, giúp giảm sự phụ thuộc lẫn nhau giữa các service khác.

Cart service:

* cart\_service xử lý giỏ hàng và quá trình mua hàng, bao gồm các chức năng thêm/xóa sản phẩm, cập nhật số lượng, tính toán tổng tiền và xác nhận đơn hàng. Service này hỗ trợ theo dõi trạng thái đơn hàng như “Đang mua”, “Đã mua”, “Đã hủy”,... đồng thời kiểm soát lịch sử mua sắm của người dùng để phục vụ cho việc gợi ý sản phẩm sau này. cart\_service cũng đảm bảo rằng các quy tắc kiểm tra tồn kho và xác thực người dùng được thực hiện đầy đủ trước khi thực hiện giao dịch.

Auth server:

* auth\_server đảm nhận toàn bộ quá trình xác thực và phân quyền truy cập trong hệ thống. Đây là nơi phát hành JWT token khi người dùng đăng nhập thành công, đồng thời kiểm tra và giải mã token để xác định quyền truy cập của mỗi request. Spring Security được tích hợp tại đây để đảm bảo quy trình bảo mật đúng chuẩn, bao gồm mã hóa mật khẩu, giới hạn đăng nhập sai, và xử lý các request trái phép. Các service khác trong hệ thống có thể gọi đến auth\_server để xác thực thông tin người dùng mà không cần tự triển khai lại logic xác thực.

Deep service (AI Service):

* deep\_service là thành phần đặc biệt trong hệ thống vì được xây dựng bằng Flask (Python) thay vì Spring Boot. Lý do là vì service này phụ trách xử lý các mô hình AI/Deep Learning phục vụ cho đề xuất sản phẩm thông minh. Dữ liệu hành vi người dùng như lịch sử tìm kiếm, sản phẩm đã mua, lượt click và đánh giá sẽ được gửi đến deep\_service, nơi các mô hình học sâu (ví dụ: Collaborative Filtering với Transformer Encoder) phân tích và trả về danh sách sản phẩm đề xuất. Flask cho phép tích hợp trực tiếp với các thư viện machine learning mạnh mẽ như PyTorch, TensorFlow, scikit-learn, đồng thời dễ dàng triển khai và kiểm thử các mô hình học máy.
* Service này giao tiếp với phần còn lại của hệ thống thông qua API REST, và có thể được mở rộng độc lập theo chu kỳ huấn luyện mô hình hoặc thay đổi thuật toán mà không ảnh hưởng đến các phần khác của hệ thống.

Admin service:

* Trong kiến trúc microservice của hệ thống, bên cạnh các service nghiệp vụ phục vụ trực tiếp cho người dùng, một thành phần không thể thiếu là admin\_service, đóng vai trò trung tâm trong việc quản trị và giám sát toàn bộ hệ thống. Service này được triển khai độc lập sử dụng nền tảng Spring Boot, tuân thủ nghiêm ngặt các nguyên tắc thiết kế của mô hình microservice, đặc biệt là nguyên tắc tách biệt trách nhiệm và khả năng mở rộng độc lập.
* admin\_service được thiết kế nhằm phục vụ riêng cho nhóm người dùng có quyền quản trị hệ thống, cho phép thực hiện các thao tác giám sát, cấu hình, và can thiệp vào hoạt động của các service còn lại mà không làm ảnh hưởng đến kiến trúc vận hành tổng thể. Về mặt chức năng, service này đảm nhiệm nhiều vai trò quan trọng. Trước hết, nó cung cấp các công cụ quản lý người dùng, bao gồm việc cập nhật thông tin tài khoản, phân quyền truy cập, và kiểm soát trạng thái hoạt động của từng người dùng trong hệ thống. Các thao tác này được bảo mật chặt chẽ thông qua cơ chế xác thực phân quyền dựa trên JWT token, kết hợp với Spring Security để đảm bảo chỉ những người có vai trò quản trị mới có quyền truy cập vào các API nhạy cảm.
* Ngoài ra, admin\_service cho phép quản trị viên tương tác trực tiếp với product\_service và cart\_service để kiểm soát thông tin sản phẩm cũng như quá trình đặt hàng. Chức năng này đặc biệt hữu ích trong các tình huống cần kiểm duyệt sản phẩm, xử lý đơn hàng bất thường, hoặc can thiệp vào quy trình hậu mãi. Service cũng tích hợp các công cụ giám sát hiệu suất hệ thống, cho phép thu thập và hiển thị các chỉ số vận hành (metrics), giúp quản trị viên theo dõi tình trạng hoạt động của các service thông qua dữ liệu từ eureka\_server hoặc thông qua các hệ thống giám sát như Spring Actuator, Prometheus hoặc Grafana.
* Đặc biệt, trong bối cảnh hệ thống tích hợp một mô hình AI để cá nhân hóa trải nghiệm người dùng, admin\_service đóng vai trò là cầu nối giữa tầng nghiệp vụ và tầng trí tuệ nhân tạo. Thông qua kết nối với deep\_service, service này hỗ trợ kiểm tra kết quả đề xuất sản phẩm, theo dõi hiệu suất mô hình, và đưa ra yêu cầu huấn luyện lại khi cần thiết. Tính năng này không những tăng tính minh bạch cho hệ thống mà còn đảm bảo khả năng kiểm soát chất lượng đầu ra của các mô hình học máy trong thực tế triển khai.
* Tóm lại, admin\_service không chỉ là một thành phần quản trị đơn thuần mà còn là trung tâm điều phối và giám sát hoạt động toàn hệ thống. Việc triển khai service này một cách độc lập và chuẩn hóa theo kiến trúc microservice mang lại nhiều lợi ích, bao gồm khả năng mở rộng linh hoạt, đảm bảo bảo mật cao, và đặc biệt là tạo điều kiện cho việc vận hành hệ thống một cách hiệu quả, ổn định và có khả năng thích ứng cao với các yêu cầu thay đổi trong tương lai.

Tổng thể, sáu service này được tổ chức theo đúng nguyên tắc phân tách trách nhiệm trong kiến trúc microservice, đảm bảo rằng mỗi phần của hệ thống đều hoạt động độc lập nhưng phối hợp chặt chẽ thông qua Gateway và Eureka. Việc triển khai riêng deep\_service bằng Flask cho phép tích hợp tốt với môi trường học máy, trong khi các service còn lại giữ tính ổn định, bảo mật và hiệu suất cao nhờ Spring Boot. Điều này giúp hệ thống vừa mạnh mẽ trong xử lý nghiệp vụ truyền thống, vừa thông minh trong khả năng cá nhân hóa trải nghiệm người dùng bằng trí tuệ nhân tạo.

#### Kết luận

Trong quá trình triển khai hệ thống, các thành phần chính của kiến trúc microservice đã được xây dựng và kết nối thành công, với năm server dịch vụ đã hoàn thành và hoạt động cơ bản mượt mà. Hệ thống API, bao gồm tổng cộng khoảng 27 API, đã hoàn tất và hỗ trợ các chức năng thiết yếu của hệ thống. Phần frontend phía người dùng, với tổng cộng 8 trang, đã hoàn thành được 6 trang, cho phép người dùng tương tác đầy đủ với các chức năng chính của hệ thống. Tuy nhiên, vẫn còn một số công việc chưa hoàn thiện, đặc biệt là phần service quản lý người dùng (user\_service) chưa được hoàn thiện hoàn toàn và server admin vẫn chưa được triển khai.

## Xây dựng fontend bằng reactjs

### Tổng quan

Trong quá trình phát triển frontend cho hệ thống web, ReactJS được lựa chọn làm framework chính nhờ vào khả năng xây dựng giao diện người dùng một cách linh hoạt, hiện đại và có khả năng tái sử dụng cao. Kiến trúc frontend được tổ chức theo mô hình component, trong đó mỗi trang chức năng như trang sản phẩm, giỏ hàng, đăng nhập, tìm kiếm… được xây dựng thành các component riêng biệt, đảm nhiệm vai trò hiển thị và tương tác với người dùng. React đóng vai trò trung gian giao tiếp với backend thông qua các lời gọi API HTTP, truyền và nhận dữ liệu động để cập nhật giao diện theo thời gian thực. Đặc biệt, hệ thống bảo mật được triển khai bằng cách sử dụng HTTP-only cookie để lưu trữ token xác thực, đảm bảo an toàn cho người dùng khi sử dụng. Việc kết hợp giữa React và kiến trúc microservice phía backend giúp tách biệt rõ ràng giữa giao diện và xử lý nghiệp vụ, từ đó nâng cao khả năng mở rộng, bảo trì và phát triển độc lập của hệ thống.

### Cơ chế bảo mật token

Trong quá trình xây dựng frontend bằng ReactJS, cơ chế bảo mật token được triển khai thông qua giao thức HTTP với việc lưu trữ token truy cập trong bộ nhớ tạm thời (memory) hoặc các cơ chế tách biệt khỏi client-side storage dễ bị khai thác như localStorage hoặc sessionStorage. Việc sử dụng HTTP-only token là một chiến lược bảo mật phổ biến, trong đó token được gửi và nhận thông qua cookie có thuộc tính HttpOnly, giúp ngăn chặn các cuộc tấn công XSS (Cross-Site Scripting), vì JavaScript phía client không thể truy cập token này. Mặc dù phương pháp này yêu cầu cấu hình đồng bộ giữa frontend và backend để xử lý cookie đúng cách — đặc biệt trong môi trường phân tán theo kiến trúc microservice — nhưng nó mang lại lợi ích lớn trong việc tăng cường bảo vệ thông tin xác thực của người dùng. Cách tiếp cận này góp phần nâng cao tính an toàn của hệ thống, đặc biệt trong bối cảnh các ứng dụng web hiện đại ngày càng dễ trở thành mục tiêu của các hình thức tấn công khai thác trình duyệt.

### Cơ chế lưu ảnh và xây dựng server ảnh

Trong hệ thống được triển khai, các API xử lý ảnh được thiết kế theo hướng mã hóa và truyền tải dữ liệu một cách trực tiếp và hiệu quả giữa client và backend. Cụ thể, khi người dùng tải ảnh lên, ảnh sẽ được mã hóa thành mảng byte (byte[]) tại frontend và gửi kèm trong payload của HTTP request về phía backend. Tại đây, ảnh sẽ được lưu trữ trực tiếp dưới dạng nhị phân trong cơ sở dữ liệu SQL Server, thường thông qua kiểu dữ liệu VARBINARY(MAX).

Khi cần hiển thị ảnh, backend cung cấp một API riêng chuyên dụng, thực hiện truy vấn dữ liệu ảnh từ cơ sở dữ liệu, sau đó trả về ảnh dưới dạng dữ liệu nhị phân kèm header Content-Type phù hợp (ví dụ: image/jpeg hoặc image/png). Phía frontend chỉ cần gán thuộc tính src của thẻ <img> về đúng đường dẫn API đó để trình duyệt tự động hiển thị ảnh mà không cần xử lý thêm.

Cách tiếp cận này giúp đơn giản hóa luồng truyền ảnh, không cần phụ thuộc vào hệ thống lưu trữ tệp tách biệt như file system hoặc cloud storage. Đồng thời, việc lưu ảnh trực tiếp trong cơ sở dữ liệu giúp dễ dàng quản lý, đồng bộ, sao lưu và bảo mật ảnh gắn liền với các thực thể liên quan (như sản phẩm, người dùng). Việc sử dụng API trả ảnh động giúp kiểm soát truy cập tốt hơn, có thể tích hợp dễ dàng các lớp xác thực hoặc phân quyền. Đây là một giải pháp hiệu quả trong các hệ thống vừa và nhỏ, nơi việc quản lý tập trung và toàn vẹn dữ liệu là ưu tiên hàng đầu.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

### Mô hình

Trong dự án này, chúng tôi đã xây dựng thành công một mô hình học sâu nhằm dự đoán hành vi mua sản phẩm của người dùng dựa trên dữ liệu hành vi thực tế. Mô hình sử dụng kiến trúc dựa trên Transformer Encoder kết hợp với Fully Connected Network, huấn luyện trên tập dữ liệu lớn gồm 500,000 dòng, trong đó có sự mất cân bằng nhẹ giữa hai lớp (0 và 1).

Kết quả đánh giá trên tập validation cho thấy:

Accuracy đạt 97.46%, cho thấy mô hình phân loại tổng thể rất chính xác.

F1-score cho lớp dương đạt 96.55%, cho thấy mô hình nhận diện tốt các trường hợp người dùng sẽ mua hàng.

Sự khớp giữa phân phối nhãn thực tế và dự đoán khẳng định mô hình học được xu hướng tổng thể, không bị lệch nhãn.

Ma trận nhầm lẫn cho thấy số lượng lỗi rất thấp và phân bố đều, minh chứng cho khả năng tổng quát hóa mạnh và độ ổn định của mô hình.

Từ đó, mô hình này có thể ứng dụng hiệu quả trong các hệ thống đề xuất sản phẩm, marketing cá nhân hóa, phân tích hành vi khách hàng, giúp cải thiện trải nghiệm người dùng và tăng tỷ lệ chuyển đổi.

### Ứng dụng web

Mặc dù các server về cơ bản đã hoạt động ổn định, hệ thống vẫn chưa được tối ưu hoàn toàn, đặc biệt là chưa được kiểm tra toàn diện và chưa thực hiện các bài kiểm tra stress hoặc kiểm thử hệ thống toàn diện để đảm bảo tính ổn định trong môi trường sản xuất. Một vấn đề đáng chú ý là server Flask, được sử dụng để triển khai mô hình AI, hiện đang gặp khó khăn về hiệu suất do mô hình AI chưa được tối ưu đầy đủ, dẫn đến việc xử lý chậm và tải nặng cho hệ thống.

Về mặt phát triển frontend, mặc dù 6 trong số 8 trang của giao diện người dùng đã hoàn thành, các trang còn lại cần tiếp tục phát triển để đảm bảo tính đầy đủ của giao diện người dùng. Phía admin, chưa có bất kỳ trang quản trị nào được phát triển, điều này ảnh hưởng đến khả năng giám sát và quản lý hệ thống từ phía người quản trị.

## Hướng phát triển

### Mô hình

Dù kết quả hiện tại rất tích cực, vẫn còn nhiều hướng đi để tiếp tục mở rộng và nâng cao hiệu quả hệ thống:

Cá nhân hóa sâu hơn:

* Kết hợp với mô hình recommendation-based như collaborative filtering, hoặc tích hợp embedding người dùng để tạo ra hệ thống đề xuất đa chiều hơn.

Phân tích theo chuỗi thời gian (temporal behavior):

* Mở rộng kiến trúc để nắm bắt chuỗi hành vi theo thời gian, sử dụng các mô hình như Transformer thời gian, giúp hiểu sâu hơn các xu hướng dài hạn của người dùng.

Tối ưu hóa bằng kỹ thuật NLP:

* Nâng cao khả năng hiểu các truy vấn tìm kiếm và bình luận sản phẩm bằng cách sử dụng embedding chuyên biệt (BERT, RoBERTa) cho văn bản.

Đánh giá online (A/B Testing):

* Triển khai mô hình vào môi trường thật và đánh giá hiệu quả bằng các chỉ số kinh doanh như CTR, CR, doanh thu, v.v.

Phát hiện bất thường hoặc hành vi lừa đảo:

* Sử dụng mô hình hiện tại như nền tảng để mở rộng sang bài toán anomaly detection, nhằm phát hiện các hành vi không bình thường.

Tối ưu hóa hiệu năng inference:

* Đơn giản hóa mô hình hoặc áp dụng kỹ thuật quantization/pruning để phục vụ các thiết bị có tài nguyên thấp hơn (mobile, edge).

### Ứng dụng web

Nhìn chung, hệ thống đang trong giai đoạn phát triển với một nền tảng cơ bản đã được hoàn thiện, nhưng vẫn cần tiến hành tối ưu hóa hiệu suất, hoàn thiện các tính năng còn thiếu, và thực hiện kiểm thử toàn diện để đảm bảo hoạt động ổn định và khả năng mở rộng trong môi trường thực tế. Trong các giai đoạn tiếp theo, hướng phát triển sẽ tập trung vào việc hoàn thiện các service còn thiếu, đặc biệt là user\_service và admin\_service. Đồng thời, cần tối ưu hóa các mô hình AI trong server Flask, cải thiện hiệu suất hệ thống và hoàn thiện các trang frontend còn lại. Một trong những ưu tiên quan trọng sẽ là triển khai các bài kiểm tra chức năng và tối ưu hóa các yếu tố khác để đảm bảo hệ thống có thể đáp ứng yêu cầu của người dùng và hoạt động mượt mà khi đưa vào vận hành thực tế.

# DANH MỤC TÀI LIỆU THAM KHẢO

## Tài liệu tham khảo chính (dẫn đến các bài nghiên cứu hoặc báo cáo)

### Thông tin sự hiệu quả của hệ thống gợi ý

Stylitics: "Fashion Product Recommendation Systems and Their Role in Consumer Engagement and Conversion."

* <http://stylitics.com/resources/blog/fashion-product-recommendations/?utm_source=chatgpt.com>

ViSenze: "How AI-driven Product Recommendations Can Grow E-Commerce Revenue in Fashion."

* <http://www.visenze.com/blog/2023/07/19/ace-your-product-recommendations-to-grow-revenue/?utm_source=chatgpt.com>

Grand View Research: "Recommendation Engine Market - Analysis and Forecast 2024."

* <http://www.grandviewresearch.com/industry-analysis/recommendation-engine-market-report?utm_source=chatgpt.com>

The Guardian (Marks & Spencer): Using AI for Personalization in Fashion Retail.

* <http://www.theguardian.com/business/article/2024/sep/05/m-and-s-using-ai-to-advise-shoppers-body-shape-style-preferences?utm_source=chatgpt.com>

Wired (The Yes): AI-Driven Fashion Recommendations: How AI is changing the way people shop for clothes.

* <http://www.wired.com/story/ai-personalised-shopping?utm_source=chatgpt.com>

ArXiv: "Improving Recommendation Systems for E-Commerce: A Case Study in Fashion Retail."

* <http://arxiv.org/abs/1908.08327?utm_source=chatgpt.com>

ArXiv: "Optimizing Fashion Product Recommendations for Profit Maximization."

* <http://arxiv.org/abs/2203.06641?utm_source=chatgpt.com>

Đề xuất sản phẩm chiếm trung bình 31% doanh thu của các trang thương mại điện tử:

* <https://www.retaildive.com/news/stylitics-data-recommendation-engine-sales/562761/>

Tại Amazon, hệ thống gợi ý đóng góp tới 35% doanh thu:

* <https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/how-retailers-can-keep-up-with-consumers>

Tăng tỷ lệ chuyển đổi lên đến 300% nhờ hệ thống gợi ý:

* <https://www.mckinsey.com/industries/retail/our-insights/personalization-in-retail>

Tăng giá trị đơn hàng trung bình (AOV):

* <https://www.visenze.com/press/visual-ai-in-retail-report-2021/>

Tăng 22% giá trị vòng đời khách hàng (CLV) nhờ cá nhân hóa:

* <https://www.salesforce.com/blog/personalized-customer-experiences/>

Giảm tỷ lệ thoát trang và tăng thời gian duyệt web nhờ hệ thống gợi ý:

* <https://www.bigcommerce.com/blog/personalized-recommendations/>

91% người tiêu dùng thích các thương hiệu cá nhân hóa trải nghiệm (Accenture):

* <https://www.accenture.com/us-en/insights/interactive/personalization>

Sự bùng nổ dữ liệu người dùng và ứng dụng AI trong hệ thống đề xuất:

* <https://www.forbes.com/sites/forbestechcouncil/2023/02/06/how-ai-is-transforming-ecommerce-product-recommendations/>

Ứng dụng hệ thống gợi ý đa lĩnh vực: giáo dục, y tế, giải trí, ngân hàng:

* <https://towardsdatascience.com/a-comprehensive-guide-to-recommendation-systems-in-2021-4f8f5cb9b87b>

Boston Consulting Group (BCG): cá nhân hóa tăng gấp đôi ROI:

* <https://www.bcg.com/publications/2021/the-future-of-personalization>

### Nguồn thông tin về phần Ứng dụng thực tế - Các công ty sử dụng CF

Paper "Matrix Factorization Techniques for Recommender Systems" (ACM):

* <https://dl.acm.org/doi/10.1145/1608565.1608614>

Paper "Amazon.com Recommendations: Item-to-Item Collaborative Filtering" (ACM):

* <https://dl.acm.org/doi/10.1145/864220.864230>

Spotify Engineering Blog:

* <https://engineering.atspotify.com/2015/10/what-we-learned-building-discover-weekly/>

Paper "Deep Neural Networks for YouTube Recommendations" (arXiv):

* <https://arxiv.org/abs/1606.07792>

Alibaba paper (Challenge về recommendation):

* <https://tianchi.aliyun.com/competition/entrance/231522/information>

Một số blog từ Alibaba Cloud:

* <https://www.alibabacloud.com/blog/how-alibaba-built-its-recommendation-engine_594095>

Research paper overview:

* <https://research.facebook.com/publications/people-you-may-know/>

LinkedIn Engineering Blog (RecSys conference paper):

* <https://engineering.linkedin.com/blog/2016/06/large-scale-recommendation-systems>

### Nguồn thông tin về phần Ứng dụng thực tế - Các công ty sử dụng CF-based Encode Tranformer

YouTube Recommendation System:

* <https://research.google/pubs/pub45530/>
* (Paper: "Deep Neural Networks for YouTube Recommendations", mô tả cách YouTube sử dụng các mô hình dựa trên sequence và attention.)

Alibaba Recommendation Research:

* <https://arxiv.org/abs/1905.06874>
* (Paper: "Behavior Sequence Transformer for E-commerce Recommendation in Alibaba", trình bày cách Alibaba sử dụng Transformer để model hóa hành vi người dùng.)

Amazon Personalize:

* <https://aws.amazon.com/personalize/>
* (Trang chủ dịch vụ Amazon Personalize, mô tả việc dùng deep learning và transformer-like models để gợi ý.)

TikTok Recommendation System:

* <https://arxiv.org/abs/2008.09200>
* (Paper: "Monolith: Real Time Recommendation System With Collisionless Embedding Table", TikTok áp dụng mô hình attention-based cho recommendation.)

SASRec: Self-Attentive Sequential Recommendation:

* <https://arxiv.org/abs/1808.09781>
* (Paper: SASRec, tiên phong trong việc dùng self-attention cho bài toán recommendation theo chuỗi.)

## Cố vấn hỗ trợ lý luận và phân tích dự án

Trong dự án này, ChatGPT đóng vai trò là một cố vấn hỗ trợ lý luận và phân tích, giúp người dùng trong các giai đoạn phát triển, tối ưu hóa và triển khai các chiến lược và kỹ thuật. Nhiệm vụ của ChatGPT bao gồm:

Hỗ trợ phân tích và thiết kế: ChatGPT cung cấp các gợi ý và phân tích liên quan đến các giải pháp kỹ thuật, cấu trúc dữ liệu, và phương pháp học sâu (Deep Learning) cho các bài toán như hệ thống đề xuất sản phẩm, nhận diện hình ảnh, và các mô hình học máy. ChatGPT giúp xác định các yếu tố quan trọng trong việc xây dựng mô hình, tối ưu hiệu quả và giảm thiểu các vấn đề như overfitting.

Xây dựng ý tưởng và sáng tạo: ChatGPT đóng góp vào việc hình thành các ý tưởng sáng tạo cho dự án, bao gồm các chiến lược gợi ý sản phẩm và việc áp dụng các mô hình học sâu hiện đại như Transformer hay Collaborative Filtering. ChatGPT cũng giúp người dùng đưa ra các phân tích xu hướng ngành và thực tiễn ứng dụng công nghệ vào lĩnh vực thời trang, tối ưu hóa trải nghiệm người dùng.

Hỗ trợ tư vấn kỹ thuật: Trong suốt quá trình phát triển dự án, ChatGPT cung cấp các phân tích chi tiết về mã nguồn, cấu trúc hệ thống, và cách ứng dụng các phương pháp mới vào mô hình học sâu, giúp người dùng tối ưu hóa hiệu suất và cải thiện chất lượng của sản phẩm.

Cung cấp các tài liệu và nghiên cứu: ChatGPT tìm kiếm và cung cấp các tài liệu nghiên cứu, báo cáo, và nghiên cứu trường hợp từ các lĩnh vực liên quan, cung cấp cho người dùng các dẫn chứng và minh họa thực tế để củng cố các quyết định thiết kế và triển khai dự án.

Tư vấn về công nghệ và công cụ: ChatGPT giúp người dùng lựa chọn các công cụ, phần mềm, và nền tảng công nghệ phù hợp với yêu cầu của dự án, đồng thời hướng dẫn sử dụng chúng một cách hiệu quả.

ChatGPT hỗ trợ người dùng từ những bước đầu của dự án cho đến khi hoàn thành, luôn đảm bảo rằng các quyết định kỹ thuật được đưa ra dựa trên các lý thuyết vững chắc và các phương pháp thực tiễn đã được chứng minh.

# LIÊN KẾT NGOÀI

## 1 . Nguồn lưu trữ dự án

* [ambrouse/CF\_based\_encode\_tranformer\_de\_xuat\_san\_pham](https://github.com/ambrouse/CF_based_encode_tranformer_de_xuat_san_pham)

## Nguồn dữ liệu dự án

* <https://drive.google.com/drive/folders/1tCpUZLlx7v_UZVV9qhwSa8HD5du9DdsG?usp=sharing>

## Tài liệu mô tả chức năng người dùng (FSD)

* <https://github.com/ambrouse/CF_based_encode_tranformer_de_xuat_san_pham/blob/47ca45933f9bacc83ee257fe574d3dfdf3e07335/report/M%C3%94%20T%E1%BA%A2%20GIAO%20DI%E1%BB%86N%2C%20CH%E1%BB%A8C%20N%C4%82NG%20CHO%20TRANG%20WEB.docx>

## Tài liệu mô tả database (DDD)

* <https://github.com/ambrouse/CF_based_encode_tranformer_de_xuat_san_pham/blob/47ca45933f9bacc83ee257fe574d3dfdf3e07335/report/M%C3%94%20T%E1%BA%A2%20DATABASE%20TRANG%20WEB.docx>

## Tài liệu mô tả api (API Documentation)

* <https://github.com/ambrouse/CF_based_encode_tranformer_de_xuat_san_pham/blob/47ca45933f9bacc83ee257fe574d3dfdf3e07335/report/Mo_ta_api_trang_web.docx>

## Diagram database

* <https://github.com/ambrouse/CF_based_encode_tranformer_de_xuat_san_pham/blob/47ca45933f9bacc83ee257fe574d3dfdf3e07335/web/database/Untitled%20(1).png>