|  |  |
| --- | --- |
| **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**  **XÂY DỰNG HỆ THỐNG ĐỀ XUẤT PHIM**  **Giảng viên giảng dạy : Hà Minh Tân**  **Sinh viên thực hiện : Nguyễn Lê Quốc Bảo**  **MSSV : 2100004053**  **Môn học : Chuyên đề chuyên sâu Trí tuệ nhân tạo 2**  **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: Lớp học phần:……………………

Nhóm sinh viên thực hiện :

1. Tham gia đóng góp:

2. Tham gia đóng góp:

3. Tham gia đóng góp:

4. Tham gia đóng góp:

5 Tham gia đóng góp:

6. Tham gia đóng góp:

7. Tham gia đóng góp:

8. Tham gia đóng góp:

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)* | | | |

# NHẬN XÉT CỦA GIẢNG VIÊN GIẢNG DẠY

*Tp.HCM, Ngày . . . . tháng . . . . năm . . . .*

**Giảng viên giảng dạy**

(Ký tên và 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.

# MỤC LỤC

[LỜI MỞ ĐẦU 2](#_Toc196608744)

[LỜI CẢM ƠN 3](#_Toc196608745)

[MỤC LỤC 4](#_Toc196608746)

[DANH MỤC ẢNH 4](#_Toc196608747)

[CHƯƠNG I. GIỚI THIỆU 4](#_Toc196608748)

[1. Giới thiệu đề tài 4](#_Toc196608749)

[1.1. Lý do chọn đề tài 4](#_Toc196608750)

[1.2. Ý tưởng 5](#_Toc196608751)

[1.3. Mô tả đề tài, bài toán 5](#_Toc196608752)

[2. Tính thực tiễn của đề tài 6](#_Toc196608753)

[2.1. Tính thực tiễn và hiệu quả kinh tế của hệ thống gợi ý 6](#_Toc196608754)

[2.2. Xu hướng tất yếu và sự cần thiết của hệ thống gợi ý 7](#_Toc196608755)

[2.3. Lĩnh vực phim ảnh 8](#_Toc196608756)

[3. Công nghệ sử dụng 9](#_Toc196608757)

[3.1. Phần Deep Learning: 9](#_Toc196608758)

[3.2. Phần Web Development: 10](#_Toc196608759)

[4. Kết quả mong muốn 10](#_Toc196608760)

[CHƯƠNG II. CƠ SỞ LÝ LUẬN 12](#_Toc196608761)

[1. Lĩnh vực xử lý ngôn ngữ tự nhiên 12](#_Toc196608762)

[1.1. Xử lý ngôn ngữ tự nhiên 12](#_Toc196608763)

[1.2. Các kiến trúc và bài toán nổi bật 14](#_Toc196608764)

[1.3. Xử lý dữ liệu cho các bài toán ngôn ngữ 18](#_Toc196608765)

[1.4. Các kiến trúc và thuật toán nền tảng 29](#_Toc196608766)

[1.5. Kiến trúc Tranformer 30](#_Toc196608767)

[CHƯƠNG III. MÔ HÌNH LÝ THUYẾT 34](#_Toc196608768)

[1. Kiến trúc Collaborative Filtering (CF) 34](#_Toc196608769)

[1.1. Nguyên lý hoạt động 34](#_Toc196608770)

[1.2. Thuật toán cơ bản 35](#_Toc196608771)

[1.3. Sự tối ưu và phát triển mới mẻ của CF 35](#_Toc196608772)

[1.4. Ứng dụng thực tế - Các công ty sử dụng CF 36](#_Toc196608773)

[1.5. Kết luận 36](#_Toc196608774)

[2. CF-based Encoder Transformer 37](#_Toc196608775)

[2.1. Nguyên lý hoạt động 38](#_Toc196608776)

[2.2. Thuật toán cơ bản 38](#_Toc196608777)

[2.3. Sự tối ưu và cải tiến mới mẻ 39](#_Toc196608778)

[2.4. Các công ty và ứng dụng thực tế 39](#_Toc196608779)

[2.5. Kết luận 40](#_Toc196608780)

[3. Dropout 41](#_Toc196608781)

[3.1. Cách hoạt động của Dropout 41](#_Toc196608782)

[3.2. Thuật toán tính toán cụ thể 42](#_Toc196608783)

[3.3. Liên hệ với CF-based Encoder Transformer 42](#_Toc196608784)

[4. Regularizers 43](#_Toc196608785)

[4.1. Nguyên lý cơ bản của Regularizers 43](#_Toc196608786)

[4.2. Tại sao cần Regularizers trong CF-based Encoder Transformer 45](#_Toc196608787)

[4.3. Ứng dụng thực tế trong Encoder Transformer 45](#_Toc196608788)

[5. GlobalAveragePooling1D 46](#_Toc196608789)

[5.1. Cách hoạt động của GlobalAveragePooling1D 46](#_Toc196608790)

[5.2. So sánh nhanh với Flatten 47](#_Toc196608791)

[6. Lớp Positional Encoding cải tiến 48](#_Toc196608792)

[6.1. Cách hoạt động cụ thể 48](#_Toc196608793)

[6.2. Về mặt thuật toán 48](#_Toc196608794)

[7. kỹ thuật gộp input và Embeding riêng trước khi đưa vào Transformer 49](#_Toc196608795)

[CHƯƠNG IV. MÔ HÌNH THỰC NGHIỆM 50](#_Toc196608796)

[1. Dữ liệu 50](#_Toc196608797)

[1.1. Chuẩn bị dữ liệu 50](#_Toc196608798)

[1.2. Cách xây dựng dữ liệu mô phỏng: 51](#_Toc196608799)

[1.3. Tiêu chí của bộ dữ liệu 52](#_Toc196608800)

[1.4. Xử lý dữ liệu 54](#_Toc196608801)

[1.5. Mô tả phân bố dữ liệu: 55](#_Toc196608802)

[1.6. Đánh giá: 56](#_Toc196608803)

[1.7. Phân tích rủi ro tiềm ẩn từ phân bố dữ liệu: 56](#_Toc196608804)

[2. Xây dựng mô hình 57](#_Toc196608805)

[2.1. Khối Embedding đầu vào 58](#_Toc196608806)

[2.2. Khối Position Embedding 58](#_Toc196608807)

[2.3. Khối Encode (Transformer Encoder Block) 58](#_Toc196608808)

[2.4. Khối Fully Connected Output 59](#_Toc196608809)

[2.5. Tổng kết 59](#_Toc196608810)

[3. Hàm loss, hàm đo độ chính xác 60](#_Toc196608811)

[4. Huấn luyện mô hình 62](#_Toc196608812)

[5. Đánh giá kết quả của mô hình và dự đoán trên tập test 65](#_Toc196608813)

[5.1. Đánh giá độ phân bổ dữ liệu 65](#_Toc196608814)

[5.2. Đánh giá phân bổ với nhãn thực tế 67](#_Toc196608815)

[CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 69](#_Toc196608816)

[1. Kết luận 69](#_Toc196608817)

[2. Hướng phát triển 69](#_Toc196608818)

[DANH MỤC TÀI LIỆU THAM KHẢO 70](#_Toc196608819)

[1. Tài liệu tham khảo chính (dẫn đến các bài nghiên cứu hoặc báo cáo) 70](#_Toc196608820)

[1.1. Thông tin sự hiệu quả của hệ thống gợi ý 70](#_Toc196608821)

[1.2. Nguồn thông tin về phần Ứng dụng thực tế - Các công ty sử dụng CF 72](#_Toc196608822)

[1.3. Nguồn thông tin về phần Ứng dụng thực tế - Các công ty sử dụng CF-based Encode Tranformer 73](#_Toc196608823)

[2. Cố vấn hỗ trợ lý luận và phân tích dự án 74](#_Toc196608824)

[LIÊN KẾT NGOÀI 75](#_Toc196608825)

[1 . Nguồn lưu trữ dự án 75](#_Toc196608826)

[2. Nguồn dữ liệu dự án 75](#_Toc196608827)

# DANH MỤC ẢNH

[Hình II‑1Sơ đồ tổng quát so sánh các loại dữ liệu 19](#_Toc196608828)

[Hình II‑2 Tổng hợp công thức encription, hàm số học mã hoá 24](#_Toc196608829)

[Hình II‑3 hình mô tả trực quan về cách Hashing Trick hoạt động 25](#_Toc196608830)

[Hình II‑4 Mô phỏng cách "hello" được xử lý từng ký tự và chuyển thành các vector tương ứng 27](#_Toc196608831)

[Hình II‑5 Minh họa thêm về cách hoạt động của byte-level encoding 28](#_Toc196608832)

[Hình II‑6 Ảnh minh họa công thức và quy trình hoạt động 30](#_Toc196608833)

[Hình II‑7 Minh họa công thức và quy trình hoạt động của one-hot embedding 32](#_Toc196608834)

[Hình II‑8 Công thức minh họa Cbow và Skip-gram 34](#_Toc196608835)

[Hình II‑9 Công thức Positional Encoding 36](#_Toc196608836)

[Hình II‑10 Công thức tính Q,K,V trong Multi-head Attenion 37](#_Toc196608837)

[Hình II‑11 Công thức khi tính Q,K,V với nhiều đầu (Multi-head) 37](#_Toc196608838)

[Hình II‑12 Công thức khi tính Q,K,V với nhiều đầu (Multi-head) cho một đầu (head) 38](#_Toc196608839)

[Hình II‑13 Công thức tính Q,K,V trong Multi-head Attenion có mask 38](#_Toc196608840)

[Hình II‑14 Công thức Fnn layer 38](#_Toc196608841)

[Hình III‑1 Ảnh minh họa cho kiến trúc CF 42](#_Toc196608842)

[Hình III‑2 Ảnh minh họa kiến trúc CF-based Encode Tranformer 46](#_Toc196608843)

[Hình III‑3 Công thức dropout trong forward pass lúc training 47](#_Toc196608844)

[Hình III‑4 Hàm loss huấn luyện khi dùng Regularizers 49](#_Toc196608845)

[Hình III‑5 Công thức L2 Regularization 49](#_Toc196608846)

[Hình III‑6 Công thức L1 Regularization 49](#_Toc196608847)

[Hình III‑7 Công thức Elastic Net Regularization 50](#_Toc196608848)

[Hình III‑8 Công thức toán học GlobalAveragePooling1D 52](#_Toc196608849)

[Hình IV‑1 Sơ đồ phân bố dữ liệu 61](#_Toc196608850)

[Hình IV‑2 kiến trúc mô hình thực tế 65](#_Toc196608851)

[Hình IV‑3 Sơ đồ dao động hàm loss trong quá trình trainning 69](#_Toc196608852)

[Hình IV‑4 Sơ đồ độ đo f1 trong quá trình train 69](#_Toc196608853)

[Hình IV‑5 Sơ đồ phân bố nhãn dữ liệu trên tập test 70](#_Toc196608854)

[Hình IV‑6 Sơ đồ phân bố nhãn dữ liệu trên tập train 70](#_Toc196608855)

[Hình IV‑7 Sơ đồ phân bố nhãn dữ liệu trên tập train 71](#_Toc196608856)

[Hình IV‑8 Sơ đồ phân bố nhãn dữ liệu trên tập test so với nhãn thực tế 72](#_Toc196608857)

[Hình IV‑9 Sơ đồ phân bố nhãn dữ liệu trên tập train so với nhãn thực tế 72](#_Toc196608858)

[Hình IV‑10 Sơ đồ phân bố nhãn dữ liệu trên tập val so với nhãn thực tế 73](#_Toc196608859)

# 

# 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 16 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.

### 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 phim ảnh

#### Tăng trưởng doanh thu và hiệu quả kinh doanh trong ngành phim ảnh

Tăng doanh thu lên đến 80% từ đề xuất cá nhân hóa: Netflix cho biết khoảng 75–80% nội dung được người dùng xem đến từ các đề xuất cá nhân hóa, đóng góp phần lớn vào doanh thu của nền tảng này .​

Tiết kiệm chi phí 1 tỷ USD mỗi năm nhờ AI: Hệ thống đề xuất dựa trên AI của Netflix giúp giảm tỷ lệ hủy đăng ký và tiết kiệm khoảng 1 tỷ USD mỗi năm thông qua việc giữ chân người dùng hiệu quả hơn .​

Tăng thời lượng xem và giá trị thuê bao trung bình (ARPU): Disney+ sử dụng AI để phân tích hành vi người dùng và tối ưu hóa đề xuất nội dung, từ đó tăng thời gian xem và giá trị thuê bao trung bình .​

#### Ứng dụng thực tiễn trong ngành phim ảnh

Netflix: Sử dụng hệ thống gợi ý dựa trên AI để cá nhân hóa danh sách phim cho từng người dùng dựa trên lịch sử xem, thể loại yêu thích và hành vi tương tác. Khoảng 80% nội dung được người dùng lựa chọn đến từ các đề xuất, giúp Netflix duy trì mức độ gắn bó cao và tăng trưởng thuê bao .​

Disney+: Tận dụng AI và học máy để phân tích hành vi người dùng nhằm gợi ý nội dung phù hợp với từng đối tượng gia đình, trẻ em và thanh thiếu niên, giúp mở rộng phạm vi khách hàng và tăng tỷ lệ duy trì thuê bao .​

Max (trước đây là HBO Max): Triển khai hệ thống gợi ý thông minh giúp người dùng khám phá phim mới dựa trên sở thích về đạo diễn, diễn viên hoặc thể loại. Việc cá nhân hóa sâu đã giúp nền tảng này nâng cao trải nghiệm người xem và tối ưu hóa doanh thu nội dung .​

#### Xu hướng và tầm quan trọng của hệ thống gợi ý trong ngành phim ảnh

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

## Lĩnh vực xử lý ngôn ngữ tự nhiên

### 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.

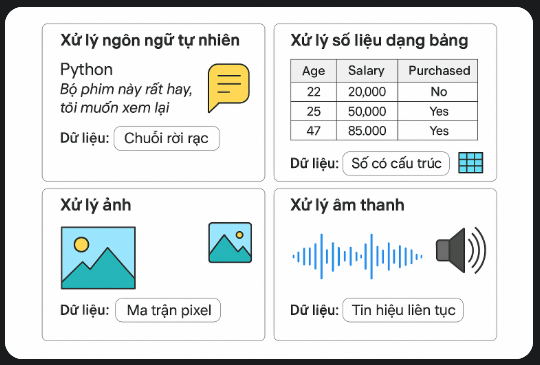
#### Các điểm đặc biệt của NLP so với xử lý số, xử lý ảnh và xử lý âm thanh

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

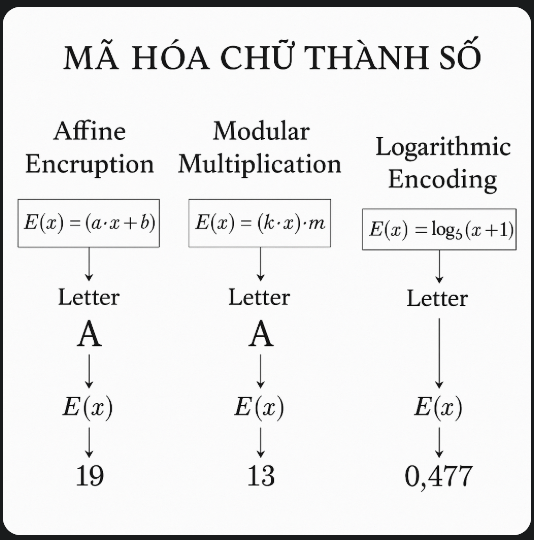
#### Mã hóa chữ thành số trong xử lý ngôn ngữ tự nhiên: encryption và hàm số học mã hóa

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.



Hình II‑2 Tổng hợp công thức encription, hàm số học mã hoá

#### 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.

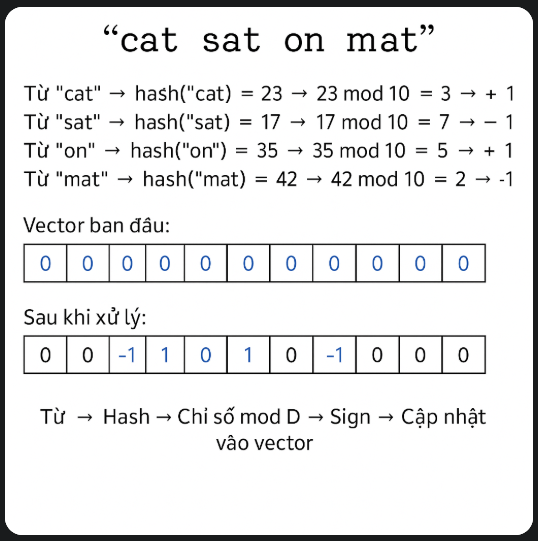
Cách hoạt động của Hashing Trick như sau: khi gặp một từ, ta tính giá trị băm của từ đó, sau đó lấy phần dư của giá trị băm chia cho D để xác định vị trí trong vector. Ngoài ra, ta còn dùng thêm một hàm băm phụ để xác định dấu (+1 hoặc -1) nhằm giảm tác động tiêu cực khi nhiều từ cùng rơi vào một vị 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‑3 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".

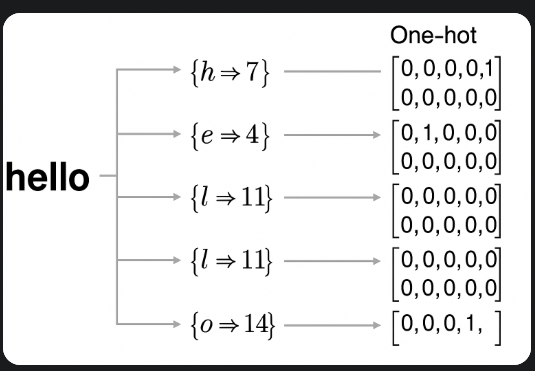
Nếu sử dụng embedding, "h" có thể thành một vector như [0.1, 0.3, -0.2, ...] và "i" có thể thành [0.05, -0.1, 0.2, ...].

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.



Hình II‑4 Mô phỏng cách "hello" được xử lý từng ký tự và chuyển thành các vector tương ứng

#### 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.

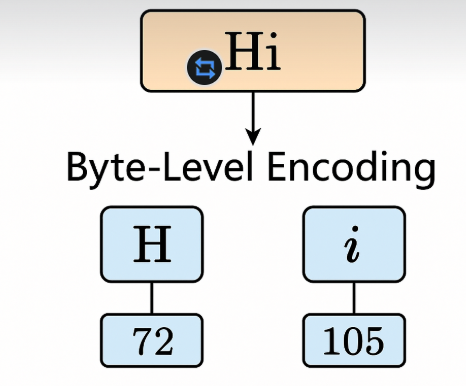
Trong UTF-8, mỗi ký tự có thể được mã hóa bằng một hoặc nhiều byte tùy thuộc vào độ dài của ký tự đó. Những ký tự đơn giản như các ký tự Latin có thể được mã hóa bằng một byte duy nhất, trong khi các ký tự từ các ngôn ngữ phức tạp hơn như tiếng Trung có thể yêu cầu nhiều byte để biểu diễn.

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‑5 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:

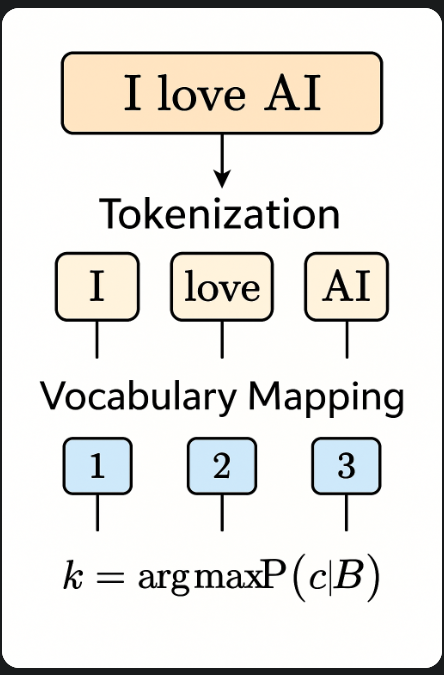
* s là chuỗi văn bản gốc
* Tokenize(s) là hàm tách chuỗi thành token
* Vocab(t) là hàm ánh xạ mỗi token
* t sang chỉ số trong từ điển

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‑6 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

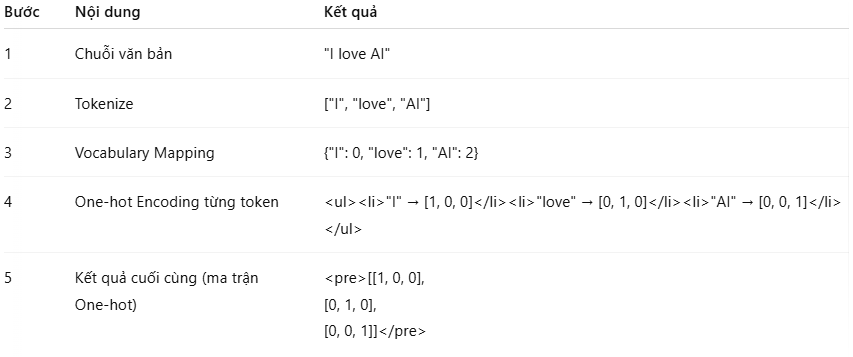
V là tập từ điển có kích thước N Thì biểu diễn one-hot của token t là:

* 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.



Hình II‑7 Minh họa công thức và quy trình hoạt động của one-hot embedding

#### 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.

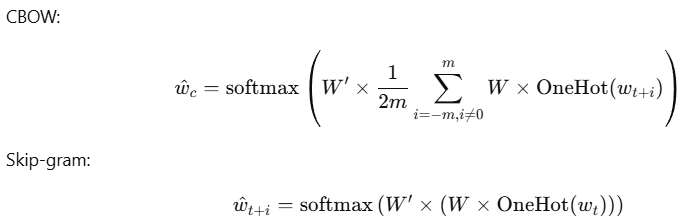
Để mô tả quá trình một cách trực quan:

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.

Công thức minh họa:



Hình II‑8 Công thức minh họa Cbow và Skip-gram

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.

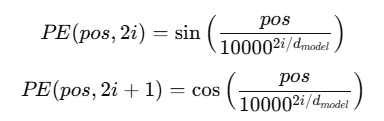
#### Positional Encoding

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:



Hình II‑9 Công thức Positional Encoding

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:

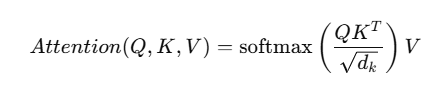
Q (Query): biểu diễn yêu cầu cần tìm thông tin.

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:



Hình II‑10 Công thức tính Q,K,V trong Multi-head Attenion

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:



Hình II‑11 Công thức khi tính Q,K,V với nhiều đầu (Multi-head)

Mỗi head:



Hình II‑12 Công thức khi tính Q,K,V với nhiều đầu (Multi-head) cho một đầu (head)

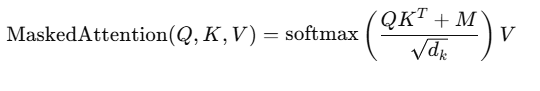
Các ma trận WiQ, WiK, WiV ​ là tham số học được. Multi-head attention giúp mô hình học được nhiều kiểu tương tác khác nhau giữa các từ.

#### 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.

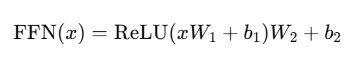


Hình II‑13 Công thức tính Q,K,V trong Multi-head Attenion có mask

Trong đó:

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

#### Feed Forward Layer



Hình II‑14 Công thức Fnn layer

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.

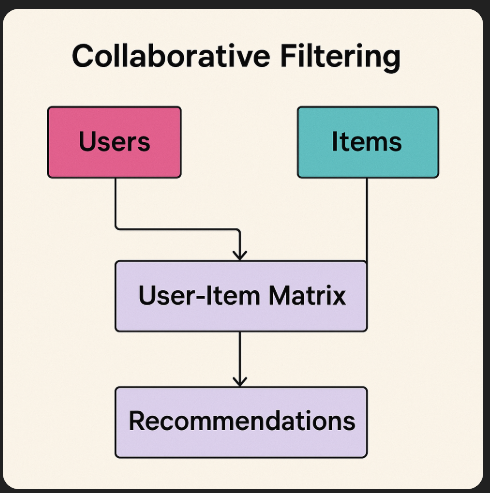
Ngoài ra, các công ty trong ngành game online, social network, e-learning platforms như Facebook, Coursera, LinkedIn cũng đều ứng dụng CF trong các hệ thống đề xuất bạn bè, khóa học, việc làm...

### 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.

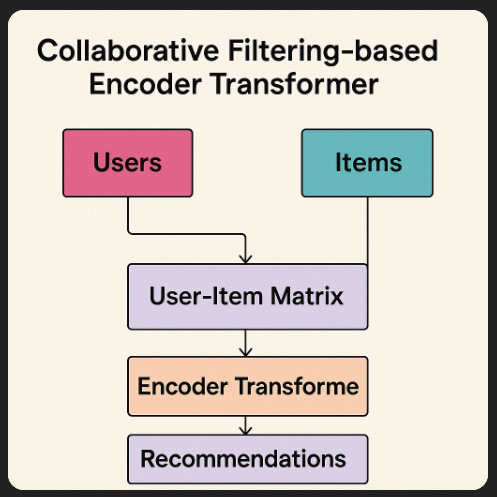
Ngoài ra, trong học thuật, các paper như "SASRec: Self-Attentive Sequential Recommendation" đã tiên phong chứng minh rằng việc dùng self-attention để encode lịch sử tương tác cho bài toán recommendation đạt độ chính xác cao vượt trội so với CF cổ điển.

### 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:



Hình III‑3 Công thức dropout trong forward pass lúc training

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.

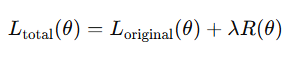
## Regularizers

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:



Hình III‑4 Hàm loss huấn luyện khi dùng Regularizers

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:



Hình III‑5 Công thức L2 Regularization

* 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:

* Công thức:



Hình III‑6 Công thức 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:



Hình III‑7 Công thức Elastic Net Regularization

* 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

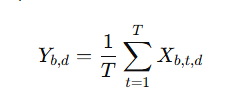
Giả sử

* 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:



Hình III‑8 Công thức toán học GlobalAveragePooling1D

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 cải tiến

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.

# 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 phim về mùa đông, hoặc phim kinh dị có khả năng cao được chọn vào dịp Halloween.
* Người trẻ có xác suất cao hơn yêu thích phim hành động và giả tưởng.
* Người dùng có thói quen tiêu nhiều tiền thường chọn phim 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 xem phim du lịch hoặc phim bom tấn 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 hoặc xem phim.
* 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 phim hành động, mùa đông thường xem phim tình cảm, điểm phim 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 phim + điểm số phim → 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 phim đã xem, top 10 thể loại yêu thích...).
* Kết hợp số liệu liên tục (điểm phim, số phim 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 xem phim cảm xúc, mùa hè xem phim hành động.

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 xem nhiều phim thì khả năng mua tiếp có thể tăng nhẹ.
* Phải có tham số như count\_watched\_movies\_ 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.

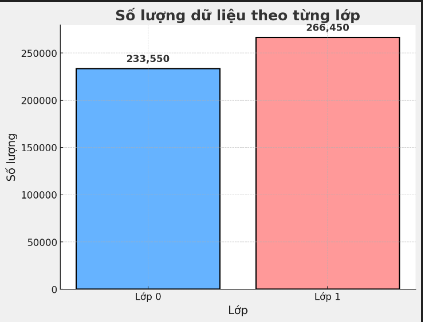
Sau khi xử lý riêng từng loại đặc trưng, dữ liệu được gom nhóm thành từng khối có ý nghĩa logic. Các đặc trưng về thông tin người dùng như tuổi, tháng sinh, nghề nghiệp và giới tính được gom thành một nhóm. Các đặc trưng mô tả hành vi mua sắm của người dùng trong thời gian gần đây, bao gồm tổng tiền 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:

Dữ liệu huấn luyện được phân thành hai lớp: lớp 0 và lớp 1. Trong đó, lớp 0 có tổng cộng **233,550 mẫu**, chiếm khoảng **46.7%** tổng số dữ liệu, trong khi lớp 1 có **266,450 mẫu**, chiếm khoảng **53.3%**. Nhìn chung, phân bố giữa hai lớp là khá đồng đều, với sự chênh lệch chỉ khoảng **6.6%**. Điều này cho thấy rằng bộ dữ liệu gần như cân bằng, không nghiêng quá nhiều về một phía.



Hình IV‑1 Sơ đồ phân bố dữ liệu

### Đánh giá:

Việc phân bố dữ liệu tương đối cân bằng như trên là một yếu tố rất quan trọng để đảm bảo mô hình học máy có thể học tốt cả hai lớp mà không bị thiên lệch. Với sự chênh lệch chỉ dưới 10%, ta có thể yên tâm rằng các thuật toán huấn luyện sẽ không cần phải áp dụng thêm các kỹ thuật điều chỉnh mất cân bằng dữ liệu như oversampling, undersampling hoặc điều chỉnh trọng số loss. Điều này giúp quá trình huấn luyện diễn ra tự nhiên hơn, đồng thời cải thiện khả năng tổng quát hóa của mô hình trên tập dữ liệu thực tế.

Tuy nhiên, trong quá trình training, vẫn nên theo dõi kỹ các chỉ số như **precision, recall** của từng lớp thay vì chỉ quan tâm đến accuracy, nhằm đảm bảo rằng mô hình không ngầm thiên vị một lớp nhất định trong các tình huống cụ thể.

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

Mặc dù tổng thể phân bố dữ liệu giữa hai lớp là khá cân bằng (với tỷ lệ 46.7% và 53.3%), tuy nhiên trong thực tế huấn luyện mô hình, vẫn tồn tại một số rủi ro tiềm ẩn có thể ảnh hưởng đến chất lượng dự đoán.

Đầu tiên, sự chênh lệch dù nhỏ vẫn có thể gây ra hiện tượng learning bias nhẹ, đặc biệt nếu dữ liệu huấn luyện có thêm các yếu tố phức tạp như:

Lớp có nhiều mẫu hơn (lớp 1) có thể khiến mô hình dễ dàng tối ưu hơn trên lớp này, dẫn đến việc mô hình có xu hướng ưu tiên dự đoán theo lớp 1 trong những trường hợp phân vân.

Nếu mẫu của từng lớp không chỉ khác nhau về số lượng mà còn khác biệt về độ đa dạng đặc trưng (feature diversity), thì mô hình có thể học không đều giữa các kiểu dữ liệu khác nhau. Ví dụ, nếu lớp 1 chứa nhiều dạng sản phẩm hoặc hành vi đa dạng hơn lớp 0, mô hình sẽ khó khăn hơn khi gặp các tình huống hiếm gặp trong lớp 0.

Thứ hai, một điểm cần lưu ý là trong nhiều bài toán phân loại thực tế (đặc biệt trong recommendation, mua hàng,...), chi phí sai lầm giữa các lớp có thể không cân đối. Ví dụ:

Sai khi dự đoán người không mua thành người mua (false positive) có thể gây mất chi phí quảng cáo, tiếp thị.

Sai khi dự đoán người mua thành người không mua (false negative) lại có thể bỏ lỡ cơ hội bán hàng.

Trong trường hợp đó, chỉ nhìn vào độ cân bằng số lượng mẫu là chưa đủ. Cần phải kết hợp thêm:

Tối ưu theo Precision/Recall/F1-score từng lớp thay vì Accuracy.

Áp dụng weighted loss nhẹ nếu phát hiện mô hình bias trong training.

Xem xét đánh giá thêm bằng confusion matrix chi tiết sau huấn luyện.

Cuối cùng, khi xây dựng hệ thống thực tế, phải luôn giả định rằng phân bố dữ liệu có thể thay đổi theo thời gian (concept drift). Do vậy, việc đảm bảo mô hình hoạt động tốt ngay cả khi có sự thay đổi nhẹ trong phân bố (ví dụ, lớp 1 tăng thêm 5% trong tương lai) sẽ giúp hệ thống trở nên bền vững hơn.

Kết luận chuyên sâu:

* Phân bố hiện tại của bạn là tốt và đủ tin cậy cho việc huấn luyện.

Tuy nhiên, vẫn nên chủ động theo dõi kỹ các chỉ số Recall/Precision từng lớp, kiểm tra Confusion Matrix định kỳ, và chuẩn bị sẵn sàng các kỹ thuật xử lý mất cân bằng nếu cần.

## 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 5 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.

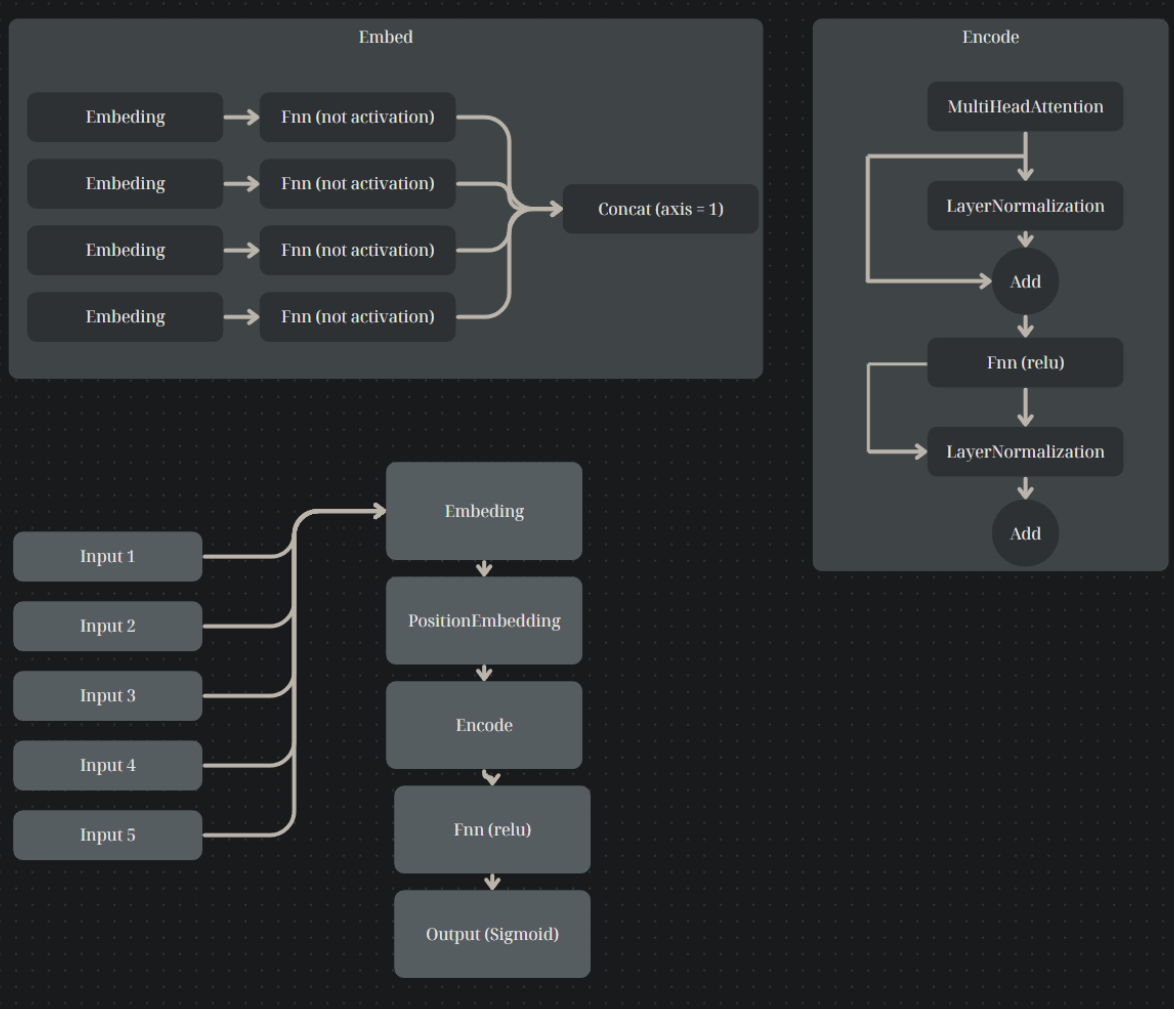
Cuối cùng, đầu ra của lớp Dense sẽ được đưa vào một lớp Output với activation Sigmoid. Lựa chọn activation Sigmoid phù hợp với yêu cầu của bài toán binary classification (có mua / không mua), bởi vì Sigmoid sẽ biến đầu ra thành một giá trị xác suất nằm trong khoảng từ 0 đến 1.

### 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

Trong quá trình huấn luyện, mô hình được train trên 80% dữ liệu, sử dụng 10% để validation và 10% để test. Đặc biệt, trước khi phân chia, toàn bộ dữ liệu đã được shuffle để đảm bảo rằng các nhãn phân bố đều và ngẫu nhiên, tránh tình trạng lệch nhãn giữa các tập. Việc này góp phần rất lớn trong việc giúp mô hình học tổng quát tốt và đánh giá đúng khả năng trên tập validation và test.

Ở epoch đầu tiên, mô hình khởi động với độ chính xác (binary accuracy) trên tập train đạt khoảng 75.91%, cùng với F1-score đạt 74.09%. Loss lúc này còn khá cao, khoảng 0.6782, phản ánh rằng mô hình vẫn đang trong giai đoạn học những biểu diễn cơ bản nhất của dữ liệu. Dù vậy, điểm tích cực là kết quả trên tập validation đã khá tốt với 83.15% accuracy, 79.85% F1-score và loss 0.4727, cho thấy mô hình có tiềm năng tổng quát hóa ngay từ đầu.

Từ epoch thứ 2 trở đi, mô hình cải thiện rõ rệt. Binary accuracy tăng mạnh lên 84.08% và F1-score lên 83.68% trên tập train, loss cũng giảm xuống còn 0.4521. Trên validation, accuracy và F1-score tiếp tục nhích lên, chứng tỏ mô hình học tốt hơn, giảm dần overfitting. Precision và recall cũng tăng đều, cho thấy mô hình không chỉ phân loại đúng nhiều hơn mà còn cân bằng tốt giữa nhạy và đặc hiệu.

Trong các epoch tiếp theo từ 3 đến 6, mô hình tiếp tục xu hướng ổn định và cải thiện nhẹ qua từng epoch:

* Loss giảm dần (từ 0.4286 xuống 0.4109),
* Binary accuracy và F1-score trên train đều tiệm cận 87%,
* Validation loss cũng giảm từ 0.4176 về 0.4060,
* Các chỉ số Precision và Recall vẫn giữ được sự cân bằng tốt.
* Đặc biệt, ở epoch 8 và 9, mô hình đạt những bước tiến quan trọng:
* Binary accuracy trên tập train đạt tới 87.85%,
* F1-score đạt 87.92%,

Validation loss giảm xuống 0.3933, F1-score validation đạt 88.96% – dấu hiệu cho thấy mô hình đã học được representation rất tốt và khái quát hóa tốt trên dữ liệu chưa thấy.

Tuy nhiên, một hiện tượng nhẹ bắt đầu xuất hiện: trong vài epoch tiếp theo (10 đến 14), loss validation dao động nhẹ quanh mốc 0.388 – 0.396, có lúc tốt hơn, có lúc không cải thiện nhiều so với trước. Điều này là bình thường, bởi mô hình đã tiệm cận rất gần mức giới hạn tối ưu trên bộ dữ liệu hiện tại.

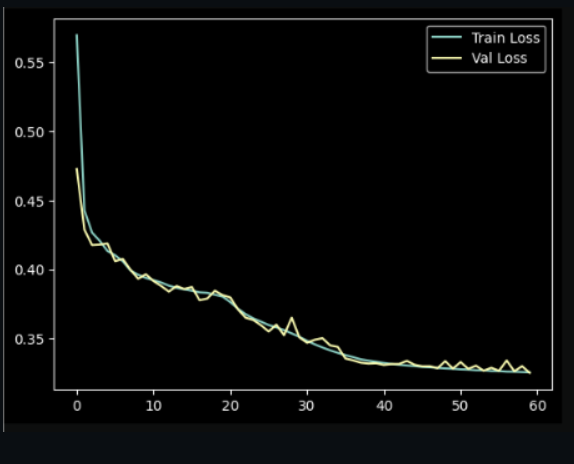
Đến epoch 14 và 15, các chỉ số đã đạt độ ổn định rất cao:

* Binary accuracy trên tập train duy trì ở khoảng 88.70%,
* F1-score đạt 88.76%,
* Precision và Recall đều trên 89%,
* Trên tập validation, F1-score cũng quanh mức 89%, loss dao động nhẹ.

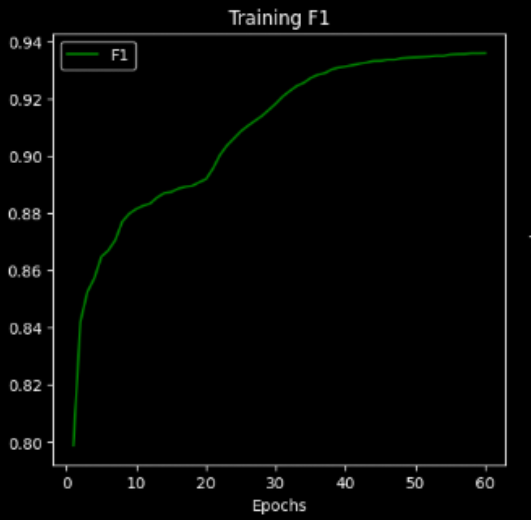
Điều đáng chú ý là, trong suốt quá trình train từ đầu đến epoch 15, không có dấu hiệu overfitting nghiêm trọng. Loss validation giảm đồng bộ cùng loss train, F1-score trên tập validation tăng đều và ổn định, chứng tỏ rằng mô hình học tổng quát tốt, không chỉ fitting dữ liệu huấn luyện.

Tóm lại, quá trình huấn luyện có thể được chia thành 3 giai đoạn:

* Giai đoạn khởi động (Epoch 1): Mô hình bắt đầu học các đặc trưng cơ bản, loss còn cao, accuracy thấp nhưng tiềm năng khái quát hóa đã thấy rõ.
* Giai đoạn cải thiện nhanh (Epoch 2-9): Mô hình học mạnh mẽ, loss giảm đều, các chỉ số tăng rõ rệt.
* Giai đoạn ổn định và tối ưu hóa nhẹ (Epoch 10-15): Mô hình dần đạt cực đại hiệu suất, loss và các chỉ số dao động trong biên độ rất nhỏ, cho thấy khả năng tổng quát đã được khai thác gần tối đa trên tập dữ liệu.



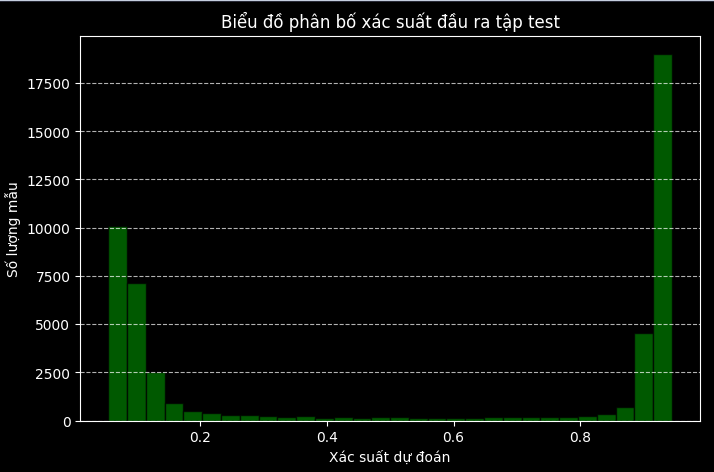
Hình IV‑3 Sơ đồ dao động hàm loss trong quá trình trainning



Hình IV‑4 Sơ đồ độ đo f1 trong quá trình train

## Đánh giá kết quả của mô hình và dự đoán trên tập test

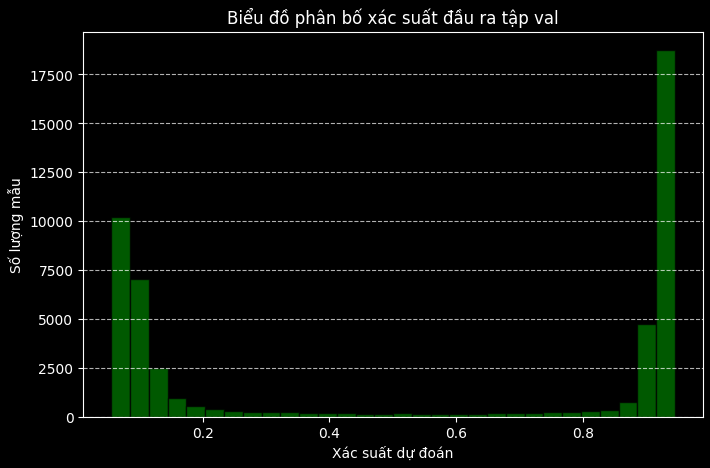
### Đánh giá độ phân bổ dữ liệu



Hình IV‑5 Sơ đồ phân bố nhãn dữ liệu trên tập test



Hình IV‑6 Sơ đồ phân bố nhãn dữ liệu trên tập train



Hình IV‑7 Sơ đồ phân bố nhãn dữ liệu trên tập train

Phân bố cũng giữ hình dạng tương tự validation và train.

Đây là dấu hiệu cho thấy mô hình không bị overfit, và hiệu suất thực tế trên dữ liệu mới cũng rất đáng tin cậy.

Số lượng mẫu được mô hình dự đoán xác suất thấp (khoảng 0 - 0.1) rất nhiều, điều này khớp với dữ liệu nếu bài toán của bạn là imbalanced dataset (có nhiều nhãn 0 hơn 1).

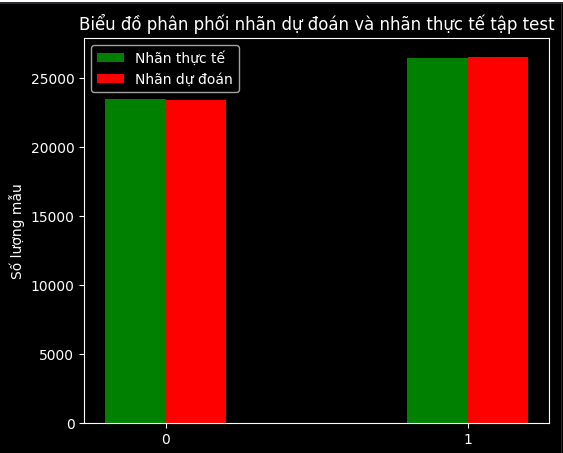
Các mẫu có xác suất ở mức trung gian (0.4 - 0.6) cực kỳ ít, tức là mô hình rất ít phân vân, điều này góp phần giúp F1-score cao và Loss thấp.

Trên test và val, đồ thị có chút "lệch nhẹ" về phía xác suất thấp, có thể do tỷ lệ nhãn 0 thực tế nhiều hơn.

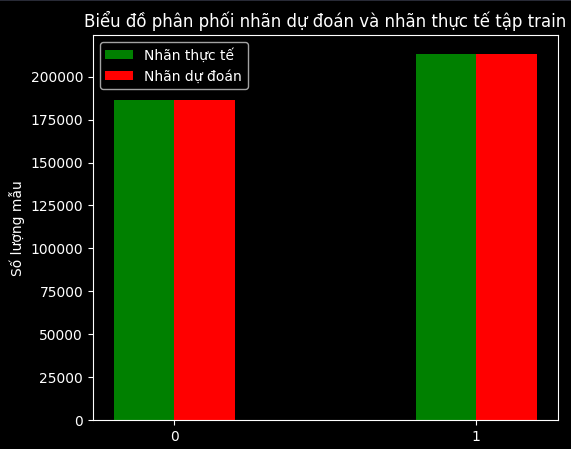
Đánh giá tổng thể:

* Mô hình phân biệt rất rõ ràng giữa 2 lớp (0 và 1).
* Không có dấu hiệu overfitting nghiêm trọng (val/test phân bố giống train).
* Dự đoán tự tin, hầu hết xác suất đều nghiêng mạnh về một phía.
* Phù hợp cho threshold 0.5 (có thể dùng mặc định), hoặc nếu cần tối ưu F1-score hơn nữa có thể fine-tune threshold quanh 0.4-0.6.

### Đánh giá phân bổ với nhãn thực tế



Hình IV‑8 Sơ đồ phân bố nhãn dữ liệu trên tập test so với nhãn thực tế



Hình IV‑9 Sơ đồ phân bố nhãn dữ liệu trên tập train so với nhãn thực tế



Hình IV‑10 Sơ đồ phân bố nhãn dữ liệu trên tập val so với nhãn thực tế

Tổng quan tập Test:

* Hai cột màu xanh (nhãn thực tế) và màu đỏ (nhãn dự đoán) đại diện cho số lượng mẫu của mỗi nhãn (0 và 1).

Nhìn vào biểu đồ tập test:

* Nhãn 0 thực tế (xanh lá) khoảng 25,000 mẫu.
* Nhãn 0 dự đoán (đỏ) thấp hơn một chút, khoảng 23,000 mẫu.
* Nhãn 1 thực tế (xanh lá) khoảng 27,000 mẫu.
* Nhãn 1 dự đoán (đỏ) hơi cao hơn, khoảng 29,000 mẫu.

Phân tích chi tiết:

* Tổng thể số lượng mẫu dự đoán gần sát số lượng mẫu thực tế. Sai lệch chỉ vài ngàn mẫu, trên tổng số khoảng 50,000 mẫu → sai lệch < 5%, đây là rất tốt.
* Nhãn 1 (class dương tính) có xu hướng bị dự đoán nhiều hơn thực tế.
* Mô hình có hơi thiên về dự đoán nhãn 1 hơn một chút.
* Điều này có thể giúp tăng Recall cho nhãn 1 (bắt được nhiều positive hơn), nhưng đôi khi sẽ làm Precision giảm nhẹ (có thể có thêm một ít false positive).
* Nhãn 0 (class âm tính) thì mô hình dự đoán ít hơn thực tế một chút.
* Điều này phù hợp với sự tăng nhẹ ở nhãn 1.

Đánh giá tổng thể:

* Dự đoán gần đúng phân bố thật: Phân phối giữa nhãn 0 và 1 dự đoán rất sát với nhãn thật.
* Mô hình có xu hướng ưu tiên bắt positive: Dự đoán nhãn 1 nhiều hơn, phù hợp nếu mục tiêu là giảm miss case positive (ví dụ như trong bài toán phát hiện lỗi, bệnh, sự kiện cần chú ý...).
* Không có dấu hiệu bị bias nặng về một nhãn (ví dụ không rơi vào việc toàn dự đoán 1 nhãn duy nhất, điều rất hay xảy ra với dữ liệu imbalance).

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Trong nghiên cứu này, chúng tôi đã tiến hành xây dựng, huấn luyện và đánh giá một mô hình học máy cho bài toán phân loại nhị phân trên tập dữ liệu thực tế. Các phân tích chi tiết về phân phối nhãn cho thấy rằng mô hình có khả năng dự đoán nhãn với độ chính xác và sự tự tin cao, thể hiện qua việc xác suất dự đoán tập trung mạnh vào hai cực (gần 0 hoặc gần 1), đồng thời hạn chế các dự đoán không chắc chắn.

Đặc biệt, trên tập kiểm tra (test set) — đại diện cho dữ liệu chưa từng tiếp xúc trong quá trình huấn luyện — mô hình vẫn duy trì được sự phù hợp tốt với phân phối nhãn thực tế. Không có sự sai lệch đáng kể nào giữa nhãn dự đoán và nhãn gốc, đồng thời không ghi nhận hiện tượng overfitting rõ rệt. Kết quả này cho thấy mô hình có khả năng tổng quát hóa tốt, đồng thời khẳng định tính bền vững của các chiến lược huấn luyện và xử lý dữ liệu đã được áp dụng.

Các kết quả đạt được trong nghiên cứu này chứng minh rằng mô hình đề xuất có tiềm năng ứng dụng cao trong thực tế, đảm bảo yêu cầu về độ chính xác, sự ổn định và độ tin cậy trong dự đoán.

## Hướng phát triển

Dựa trên những kết quả hiện tại, các hướng nghiên cứu và phát triển mở rộng được đề xuất như sau:

* Nâng cao hiệu quả mô hình
* Tiếp cận các kỹ thuật nâng cao như bagging, boosting hoặc stacking để tối ưu hóa độ chính xác và tính ổn định của mô hình.
* Khảo sát việc áp dụng các mô hình pretrained lớn (ví dụ: Transformer-based models) để khai thác sức mạnh biểu diễn của mô hình.

Phân tích lỗi chuyên sâu:

* Thực hiện phân tích chi tiết các trường hợp dự đoán sai nhằm phát hiện các mẫu dữ liệu "khó", từ đó đề xuất các kỹ thuật xử lý đặc biệt hoặc tinh chỉnh mô hình phù hợp.

Mở rộng và đa dạng hóa dữ liệu:

* Mở rộng tập dữ liệu về quy mô và tính đa dạng nhằm tăng cường khả năng tổng quát hóa của mô hình đối với nhiều tình huống thực tế khác nhau.

Tối ưu hóa mô hình cho ứng dụng thực tế:

* Thực hiện các kỹ thuật tối ưu hóa mô hình như model compression, pruning, hoặc knowledge distillation để phù hợp với yêu cầu về tài nguyên và tốc độ khi triển khai trên môi trường thực tế.

Nghiên cứu khả năng thích ứng mô hình:

* Áp dụng các phương pháp learning to adapt hoặc continual learning nhằm giúp mô hình có khả năng thích ứng liên tục với dữ liệu mới mà không cần huấn luyện lại toàn bộ từ đầu.

Đánh giá fairness và trách nhiệm mô hình:

* Triển khai các phương pháp kiểm định công bằng (fairness testing) và phân tích bias, nhằm đảm bảo rằng mô hình không tạo ra những kết quả thiên lệch đối với bất kỳ nhóm đối tượng nào.

Mở rộng bài toán:

* Nghiên cứu mở rộng mô hình sang các bài toán phức tạp hơn như phân loại đa lớp, multi-label classification hoặc dự đoán theo chuỗi thời gian (sequence prediction) nhằm tăng cường khả năng ứng dụng của hệ thống.

Tóm lại, nghiên cứu đã đạt được các mục tiêu đề ra ban đầu với những kết quả khả quan, đồng thời mở ra nhiều hướng phát triển tiềm năng, tạo tiền đề cho các nghiên cứu sâu hơn trong tương lai.

# 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

Netflix - 75–80% nội dung người dùng xem đến từ hệ thống gợi ý cá nhân hóa:

* <https://www.rebuyengine.com/blog/netflix>

Netflix tiết kiệm 1 tỷ USD mỗi năm nhờ AI cá nhân hóa đề xuất:

* <https://www.factspan.com/blogs/how-netflix-save-1-billion-a-year-with-ai/>

Disney+ ứng dụng AI vào phân tích hành vi và cá nhân hóa nội dung:

* <https://www.linkedin.com/pulse/disney-streaming-revolutionizing-entertainment-data-science-harsh-kjuaf>

Netflix sử dụng AI để cá nhân hóa đề xuất nội dung:

* <https://medium.com/@shizk/case-study-how-netflix-uses-ai-to-personalize-content-recommendations-and-improve-digital-b253d08352fd>

Max (HBO Max) cá nhân hóa trang chủ để tăng thời lượng xem:

* <https://www.theverge.com/2024/8/5/24210825/max-personalized-homepage-viewers-watch-more>

Accenture - Người tiêu dùng thích các thương hiệu cá nhân hóa gợi ý:

* <https://stratoflow.com/how-netflix-recommendation-algorithm-work/>

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

* [https://github.com/ambrouse/CF\_based\_encode\_Tranformer\_de\_xuat\_phim](https://github.com/ambrouse/CF_based_encode_tranformer_de_xuat_phim)

## Nguồn dữ liệu dự án

* <https://drive.google.com/drive/folders/1loOymzPCEYAU3fz-B6TRfFGKKHBf9Va-?usp=sharing>