

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**BÀI GIẢNG HỌC PHẦN
HỆ GỢI Ý**

Tác giả: TS. Ngô Văn Linh

Mục lục

Mục lục	3
1 GIỚI THIỆU HỆ GỌI Ý	11
Giới thiệu	11
1.1 Vai trò và tầm quan trọng của hệ gợi ý	12
1.1.1 Mục đích và tiêu chí thành công của hệ thống gợi ý	12
1.2 Các phương pháp gợi ý chính	18
1.2.1 Lọc cộng tác	19
1.2.2 Lọc dựa trên nội dung	19
1.2.3 Gợi ý dựa trên tri thức	20
1.2.4 Phương pháp lai	20
1.3 Các thách thức trong hệ gợi ý	21
1.4 Ứng dụng trong thực tế và thương mại điện tử	22
1.5 Xu hướng phát triển hệ gợi ý hiện đại	22
Kết luận	22
2 GỢI Ý DỰA TRÊN LỌC CỘNG TÁC CƠ BẢN	24
2.1 Mở đầu	24
2.2 Cơ sở lý thuyết	25
2.2.1 Ma trận người dùng – sản phẩm	25
2.2.2 Chuẩn hoá dữ liệu	25
2.2.3 Độ tương đồng giữa vector	26
2.2.4 Phân loại lọc cộng tác	27
2.3 Lọc cộng tác dựa trên bộ nhớ (Memory-based CF)	27
2.3.1 Lọc cộng tác dựa trên người dùng (User-based CF)	27

2.3.2	Lọc cộng tác dựa trên sản phẩm (Item-based CF)	28
2.3.3	Clustering-based CF	28
2.3.4	Ví dụ minh họa chi tiết	29
2.4	Dộ thừa dữ liệu và vấn đề cold-start	31
2.5	Lọc cộng tác dựa trên mô hình: Phân tích ma trận cho hệ gợi ý	33
2.5.1	Giới thiệu và mô hình cơ bản	34
2.5.2	Cập nhật nghiệm đóng: Alternating Least Squares (ALS) . .	35
2.5.3	Phiên bản cho phản hồi ngầm: Weighted ALS (WALS)	36
2.5.4	Ưu/nhược điểm của ALS (nghiệm đóng)	37
2.5.5	Cập nhật theo gradient (SGD)	37
2.5.6	Dộ phức tạp tính toán	38
2.5.7	Tổng kết và thực hành	38
2.6	Ứng dụng thực tiễn	39
2.6.1	Amazon: Gợi ý sản phẩm thương mại điện tử	39
2.6.2	Netflix: Gợi ý phim cá nhân hoá	39
2.6.3	YouTube: Gợi ý video	39
2.6.4	Mạng xã hội: Facebook, TikTok	40
2.6.5	Thương mại điện tử Việt Nam	40
2.7	Hạn chế và thách thức	40
2.7.1	Cold-start	40
2.7.2	Dữ liệu thừa	40
2.7.3	Khả năng mở rộng	41
2.7.4	Thiếu khả năng giải thích	41
2.7.5	Đa dạng và độ mới	41
3	GỢI Ý DỰA TRÊN LỌC CỘNG TÁC NÂNG CAO	42
3.1	Lọc cộng tác dựa trên học máy cơ bản	42
3.1.1	Ý tưởng chính	42
3.1.2	Quy trình học máy cơ bản	43
3.1.3	Ưu điểm và hạn chế	44
3.1.4	Kết luận	44
3.2	Lọc cộng tác nâng cao: Học biểu diễn ẩn	44

MỤC LỤC	5
3.2.1 Phân tích ma trận (Matrix Factorization)	44
3.2.2 Học sâu trong phân tích ma trận	45
3.2.3 Neural Matrix Factorization (NeuMF)	46
3.2.4 Deep Matrix Factorization (DMF)	48
3.2.5 LightGCN	52
3.3 Dữ liệu Explicit và Implicit Feedback	57
3.3.1 Explicit Feedback	58
3.3.2 Implicit Feedback	58
3.3.3 Weighted Matrix Factorization (WMF) [11]	59
3.3.4 Bayesian Personalized Ranking (BPR) [20]	61
3.4 Sử dụng đồng thời Implicit và Explicit Feedback	65
3.4.1 Tổng quan	65
3.4.2 Co-Rating [17]	67
3.4.3 Phương pháp NMTR [6]	69
3.4.4 Mô hình ITE (Implicit to Explicit Feedback [26])	71
4 GỢI Ý DỰA TRÊN NỘI DUNG	77
4.1 Tổng quan gợi ý dựa trên nội dung (Content-Based Recommendation)	77
4.2 Biểu diễn Nội dung và Sự Tương đồng Nội dung	78
4.2.1 Mô hình không gian-véc-tơ và TF-IDF	79
4.2.2 Mô hình vector và TF-IDF	80
4.2.3 Tìm kiếm Dựa trên Sự Tương Đồng	80
4.3 Thảo luận (Discussion)	84
4.3.1 Đánh giá so sánh (Comparative evaluation)	84
4.3.2 Hạn chế (Limitations)	85
4.4 Tóm tắt (Summary)	85
5 GỢI Ý DỰA TRÊN TRI THỨC	87
5.1 Giới thiệu	87
5.2 Biểu diễn tri thức và suy luận	89
5.2.1 Ví dụ minh họa: Laptop	89
5.2.2 Giải thích ví dụ	91

5.3	Ràng buộc	91
5.4	Trường hợp và độ tương đồng	93
5.4.1	Tính toán độ tương đồng cục bộ	93
5.5	Tương tác với hệ thống gợi ý dựa trên ràng buộc	94
5.5.1	Thiết lập giá trị mặc định (Defaults)	95
5.5.2	Xử lý yêu cầu không thỏa mãn	95
5.6	Tương tác với hệ thống gợi ý dựa trên case	95
5.6.1	Phản hồi chỉnh sửa (Critiquing)	96
5.6.2	Các bước chính của hệ thống critiquing	96
5.6.3	Phản hồi chỉnh sửa hợp chất (Compound critiquing)	97
5.6.4	Phản hồi chỉnh sửa động (Dynamic critiquing)	98
6	GỢI Ý DỰA TRÊN PHIÊN	100
6.1	Giới thiệu về gợi ý dựa trên phiên (Session-based Recommendation)	100
6.2	Gợi ý dựa trên độ tương đồng item-to-item	101
6.2.1	Giới thiệu	101
6.2.2	Biểu diễn item	102
6.2.3	Độ đo tương đồng	102
6.2.4	Quy trình gợi ý	103
6.2.5	Dánh giá	103
6.2.6	Ví dụ minh họa	103
6.2.7	Kết luận	104
6.3	Phát hiện các luật kết hợp (Association Rule Mining)	104
6.3.1	Giới thiệu	104
6.3.2	Các định nghĩa cơ bản	104
6.3.3	Bài toán phát hiện luật kết hợp	105
6.3.4	Nguyên lý và giải thuật Apriori	105
6.3.5	Sinh ra luật kết hợp từ tập mục thường xuyên	106
6.3.6	Sinh luật hiệu quả	106
6.3.7	Kết luận	106
6.4	Tiếp cận dựa trên RNN: mô tả chi tiết và các mở rộng	107
6.4.1	Tổng quan	107

6.4.2	Session-based Recommendations with Recurrent Neural Networks [10]	110
6.4.3	SASRec: Self-Attentive Sequential Recommendation [14]	112
6.5	BERT4Rec: Bidirectional Encoder Representations for Sequential Recommendation	114
6.5.1	Giới thiệu	114
6.5.2	Chuẩn bị dữ liệu huấn luyện	115
6.5.3	Kiến trúc mô hình	115
6.5.4	Hàm mất mát	116
6.5.5	Ưu điểm và hạn chế	116
6.5.6	Kết luận	116
6.5.7	TiSASRec: Time Interval Aware Self-Attention for Sequential Recommendation [16]	117
6.5.8	So sánh SASRec, BERT4Rec và TiSASRec	119
7	GỢI Ý DỰA TRÊN LAI GHÉP	121
7.1	Giới thiệu về hệ gợi ý lai	121
7.2	Thiết kế lai trong hệ gợi ý	122
7.3	Thiết kế lai đơn khôi	123
7.3.1	Hệ gợi ý lai kết hợp đặc trưng	123
7.3.2	Hệ gợi ý lai bổ sung đặc trưng	124
7.4	Thiết kế lai song song	125
7.4.1	Thiết kế lai song song	125
7.4.2	Lai trộn (Mixed hybrids)	126
7.4.3	Lai có trọng số (Weighted hybrids)	126
7.4.4	Lai chuyển đổi (Switching hybrids)	127
7.5	Các thiết kế lai nâng cao	128
7.5.1	Lai theo chuỗi xử lý (Pipelined Hybridization)	128
7.5.2	Lai theo tầng (Cascade Hybridization)	128
7.5.3	Lai theo siêu mức (Meta-level Hybridization)	129
8	ĐÁNH GIÁ HỆ THỐNG GỢI Ý	130
8.1	Giới thiệu	130

8.2	Các đặc điểm chung của nghiên cứu đánh giá	131
8.2.1	Tính mô tả và lắp lại	131
8.2.2	Tính hợp lệ	131
8.2.3	Dộ tin cậy (Reliability)	131
8.2.4	Tính nhạy (Sensibility)	131
8.3	Đối tượng nghiên cứu	132
8.3.1	Người dùng thực	132
8.3.2	Dữ liệu lịch sử hoặc tổng hợp	132
8.3.3	Dộ thưa dữ liệu (Sparsity)	132
8.3.4	Ví dụ tập dữ liệu phổ biến	132
8.4	Hạn chế của các phương pháp đánh giá	133
8.5	Phương pháp nghiên cứu (Research Methods)	133
8.5.1	Thiết kế thí nghiệm (Experimental Design)	133
8.5.2	Thiết kế bán thí nghiệm (Quasi-Experimental Design)	134
8.5.3	Thiết kế phi thí nghiệm (Nonexperimental Design)	134
8.5.4	Nghiên cứu tình huống (Case Study)	134
8.6	Thiết lập đánh giá (Evaluation Settings)	134
8.7	Các độ đo phổ biến trong đánh giá hệ gợi ý dựa trên dữ liệu lịch sử	135
8.7.1	Dánh giá hệ thống gợi ý theo truy xuất thông tin	135
8.7.2	Dộ đo AUC (Area Under the ROC Curve)	136
8.7.3	Dộ đo MRR (Mean Reciprocal Rank)	137
8.7.4	Dộ đo NDCG (Normalized Discounted Cumulative Gain)	138
8.7.5	Dộ đo MAP (Mean Average Precision)	139
8.7.6	Ví dụ minh họa các độ đo đánh giá	140
9	ỨNG DỤNG DEEP LEARNING: GỢI Ý TIN TỨC	143
9.1	Hệ thống gợi ý tin tức và học sâu	143
9.1.1	Tổng quan về hệ thống gợi ý tin tức	144
9.1.2	Kiến trúc tổng quát của hệ thống gợi ý học sâu	144
9.1.3	Các mô hình điển hình	145
9.1.4	Tập dữ liệu MIND	146
9.1.5	Tiền xử lý dữ liệu MIND và thống kê	146

9.1.6	Xu hướng nghiên cứu và tiềm năng	147
9.2	Mô hình gợi ý đa góc nhìn MAML [27]	149
9.2.1	Kiến trúc tổng thể	149
9.2.2	Bộ mã hoá tin tức (News Encoder)	150
9.2.3	Bộ mã hoá người dùng (User Encoder)	150
9.2.4	Bộ dự đoán click (Click Predictor)	151
9.2.5	Hàm mất mát (Loss Function)	151
9.2.6	Tóm tắt	151
9.3	Mô hình chú ý cá nhân hóa NPA [28]	151
9.3.1	Giới thiệu	151
9.3.2	Kiến trúc tổng thể	152
9.3.3	Bộ mã hoá tin tức (News Encoder)	152
9.3.4	Bộ mã hoá người dùng (User Encoder)	153
9.3.5	Bộ dự đoán click (Click Predictor)	153
9.3.6	Hàm mất mát (Loss Function)	153
9.3.7	Tóm tắt	153
9.4	Mô hình gợi ý với cơ chế tự chú ý: NRMS [29]	154
9.4.1	Giới thiệu	154
9.4.2	Kiến trúc mô hình	154
9.4.3	Hàm mất mát (Loss function)	156
9.4.4	Tóm tắt	156
9.5	Mô hình dựa gợi ý dựa trên biểu diễn bộ nhớ dài hạn và ngắn hạn LSTUR [1]	156
9.5.1	Giới thiệu	156
9.5.2	Kiến trúc tổng thể	157
9.5.3	Bộ mã hoá tin tức (News Encoder)	157
9.5.4	Bộ mã hoá người dùng (User Encoder)	158
9.5.5	Bộ dự đoán click (Click Predictor)	158
9.5.6	Hàm mất mát (Loss Function)	158
9.5.7	Tóm tắt	159
9.6	So sánh các mô hình MAML, NRMS, NPA và LSTUR	159
9.6.1	Tổng quan	159

9.6.2	So sánh chi tiết	159
9.6.3	Nhận xét	159
10	ỨNG DỤNG DEEP LEARNING: GỢI Ý DỰA TRÊN HỌC TƯƠNG PHẢN	161
10.1	Học tương phản (Contrastive Learning)	161
10.1.1	Quy trình học tương phản	162
10.1.2	Mô hình SimCLR [5]	163
10.1.3	Mô hình SimCSE [7]	165
10.2	Học tương phản trong hệ gợi ý	167
10.2.1	Nguyên lý cơ bản	167
10.2.2	Tối ưu hoá biểu diễn người dùng qua các <i>views</i>	168
10.2.3	Các cách tạo views cho người dùng	168
10.2.4	Hàm mất mát tương phản	168
10.2.5	Lợi ích trong hệ gợi ý	169
10.3	Học tương phản cho gợi ý chuỗi CL4Rec [31]	169
10.3.1	Động lực	169
10.3.2	Chuẩn bị dữ liệu huấn luyện	169
10.3.3	Data Augmentation cho chuỗi phiên	170
10.3.4	Kiến trúc mô hình	171
10.3.5	Hàm mất mát	172
10.3.6	Ưu điểm	172
10.4	Tăng cường dữ liệu trong học tương phản CoSeRec [18]	172
10.4.1	Động lực	172
10.4.2	Chiến lược tăng cường dữ liệu trong CoSeRec (Hình 10.5) . .	173
10.4.3	Kiến trúc mô hình	175
10.4.4	Hàm mất mát tương phản	175
10.4.5	Hàm mất mát kết hợp	176
10.4.6	Ưu điểm	176
10.5	Các phương pháp mở rộng học tương phản trong hệ gợi ý	176
10.5.1	Học tương phản có tính đổi biến cho gợi ý tuần tự ECL-SR [35]	176
10.5.2	Học tương phản với gợi ý dựa trên đồ thị LightGCL [3] . . .	178

Chương 1

GIỚI THIỆU HỆ GỢI Ý

Giới thiệu	11
1.1 Vai trò và tầm quan trọng của hệ gợi ý	12
1.2 Các phương pháp gợi ý chính	18
1.3 Các thách thức trong hệ gợi ý	21
1.4 Ứng dụng trong thực tế và thương mại điện tử	22
1.5 Xu hướng phát triển hệ gợi ý hiện đại	22
Kết luận	22

Giới thiệu

Trong kỷ nguyên số, lượng thông tin mà con người phải tiếp cận hằng ngày ngày càng trở nên khổng lồ. Trên các nền tảng thương mại điện tử, hàng triệu sản phẩm được trưng bày để phục vụ nhu cầu mua sắm. Trên các ứng dụng nghe nhạc và xem phim trực tuyến, hàng trăm nghìn bản nhạc và bộ phim được cập nhật mỗi ngày. Người dùng đứng trước một biển thông tin rộng lớn, nhưng khả năng tìm kiếm, lọc lựa và ra quyết định của họ lại có hạn. Hiện tượng này được gọi là quá tải thông tin, và nó tạo ra nhu cầu cấp thiết về những công cụ có khả năng hỗ trợ người dùng tìm thấy nội dung phù hợp một cách nhanh chóng và chính xác.

Hệ thống gợi ý [13, 21, 4], hay còn gọi là recommender systems, ra đời để giải quyết vấn đề này. Đây là một tập hợp các phương pháp dựa trên trí tuệ nhân tạo và khai phá dữ liệu, có chức năng phân tích hành vi, sở thích, nhu cầu của người dùng và từ đó đưa ra các gợi ý phù hợp. Về bản chất, hệ thống gợi ý giống như một trợ

lý ảo, theo dõi những gì người dùng đã quan tâm trong quá khứ và dự đoán những gì họ có thể thích trong tương lai.

Ngày nay, hệ gợi ý đã trở thành phần trung tâm trong nhiều nền tảng trực tuyến. Các công ty lớn như Amazon, Netflix, Spotify hay TikTok đều dựa rất nhiều vào hệ thống gợi ý để giữ chân khách hàng và thúc đẩy kinh doanh. Chẳng hạn, Netflix công bố rằng phần lớn nội dung mà người dùng xem đến từ các đề xuất của hệ thống gợi ý, chứ không phải từ việc tìm kiếm trực tiếp. Điều này cho thấy sức mạnh và vai trò chiến lược của hệ gợi ý trong bối cảnh hiện nay.

1.1 Vai trò và tầm quan trọng của hệ gợi ý

Hệ thống gợi ý đóng vai trò quan trọng đối với cả người dùng lẫn doanh nghiệp. Đối với người dùng, hệ gợi ý giúp tiết kiệm đáng kể thời gian tìm kiếm và mang lại trải nghiệm cá nhân hóa. Thay vì phải duyệt qua hàng trăm sản phẩm, người dùng chỉ cần dựa vào danh sách đề xuất đã được chọn lọc sẵn theo sở thích. Điều này không chỉ giúp họ thoải mái hơn trong quá trình mua sắm hay giải trí, mà còn tạo cảm giác được thấu hiểu. Ngoài ra, nhờ gợi ý, người dùng còn có cơ hội khám phá những sản phẩm hoặc dịch vụ mà bình thường họ chưa từng nghĩ đến.

Đối với doanh nghiệp, hệ gợi ý giúp tăng doanh thu và lợi nhuận. Khi khách hàng được tiếp cận đúng sản phẩm mà họ cần, khả năng họ nhấp chuột, thêm vào giỏ hàng và tiến hành mua sắm sẽ cao hơn. Ngoài ra, hệ gợi ý cũng góp phần giữ chân khách hàng, bởi trải nghiệm cá nhân hóa khiến họ quay trở lại nền tảng nhiều lần hơn. Một yếu tố nữa không thể bỏ qua là khả năng mở rộng danh mục tiêu thụ: nhờ cơ chế gợi ý long tail, doanh nghiệp không chỉ bán được các sản phẩm phổ biến mà còn đẩy mạnh doanh số từ các sản phẩm ít người biết đến.

Tóm lại, hệ thống gợi ý vừa giúp người dùng vượt qua rào cản thông tin, vừa mang lại lợi ích kinh doanh rõ rệt. Đây là lý do khiến nó trở thành một thành phần không thể thiếu trong hầu hết các dịch vụ trực tuyến hiện nay.

1.1.1 Mục đích và tiêu chí thành công của hệ thống gợi ý

Hệ thống gợi ý (*recommender system*) được triển khai với nhiều mục tiêu khác nhau tùy theo *domain*, nhóm người dùng mục tiêu và yêu cầu kinh doanh hoặc xã hội của tổ chức. Do vậy, việc xác định tiêu chí thành công cho một hệ thống gợi ý phải là một quá trình phân tích và lựa chọn có chủ đích — không tồn tại một bộ tiêu chí đánh giá chuẩn áp dụng được cho mọi kịch bản. Phần này cung cấp phân tích chuyên sâu theo từng khía cạnh chính: truy hồi thông tin, gợi ý (khám

phá/long-tail), dự đoán, tương tác người dùng và khả năng chuyển đổi. Mỗi khía cạnh được thảo luận với các mục tiêu, tiêu chí đánh giá phù hợp, xung đột tiềm ẩn giữa các mục tiêu và gợi ý thực tiễn cho thiết kế thí nghiệm và triển khai.

Quan điểm tổng quát: đa mục tiêu và phụ thuộc theo domain

Hệ thống gợi ý thường chịu yêu cầu tối ưu hoá nhiều mục tiêu khác nhau, trong đó có những mục tiêu trực tiếp phục vụ người dùng (ví dụ: giảm chi phí tìm kiếm, tăng sự hài lòng) và những mục tiêu mang tính tổ chức/kinh doanh (ví dụ: tăng doanh thu, giữ chân khách hàng). Các mục tiêu này có thể mâu thuẫn: một chiến lược tối ưu hóa tính đa dạng có thể làm giảm độ chính xác thuần túy; chính sách tối ưu hóa doanh thu có thể làm giảm trải nghiệm khám phá của người dùng. Vì vậy, trước khi xác định tiêu chí đánh giá, nhóm thiết kế cần làm rõ thứ tự ưu tiên (prioritization) giữa các mục tiêu, hoặc thiết kế hàm mục tiêu tổng hợp phù hợp với chính sách sản phẩm.

Bối cảnh domain đóng vai trò quyết định trong quá trình lựa chọn tiêu chí:

- **Dịch vụ tin tức, nội dung tức thời:** ưu tiên mục tiêu giữ chân, tốc độ cung cấp nội dung phù hợp và đảm bảo tính cập nhật.
- **Sàn thương mại điện tử:** ưu tiên chuyển đổi và doanh thu, nhưng vẫn cần giữ cân bằng để tránh giảm chất lượng trải nghiệm mua.
- **Nền tảng giải trí (video, âm nhạc):** tập trung vào thời lượng xem, đa dạng nội dung và khám phá để tăng mức độ tương tác lâu dài.
- **Ứng dụng học thuật hoặc cổng tri thức:** ưu tiên độ chính xác, tính có thể giải thích và hỗ trợ tìm kiếm mục tiêu.

Quy trình lý tưởng là khởi động với một ma trận mục tiêu rõ ràng, xác định chỉ số chính (primary KPI) và chỉ số phụ (secondary KPI), sau đó thiết kế pipeline đánh giá gồm offline tests, kiểm định counterfactual và thử nghiệm online.

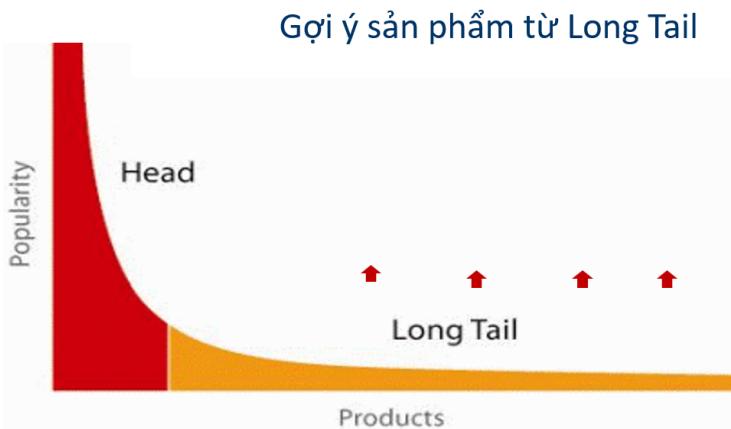
1. Khía cạnh truy hồi thông tin (Information Retrieval)

Mục tiêu phân tích Khía cạnh truy hồi thông tin của hệ thống gợi ý hướng đến tối ưu hoá chi phí tìm kiếm của người dùng khi họ có ý định rõ ràng về item mong muốn. Hệ thống ở vai trò lọc và sắp xếp các kết quả một cách nhanh chóng, giúp người dùng tìm thấy "known-item" hoặc các lựa chọn phù hợp nhất trong ít thao tác.

Tiêu chí thành công và phân tích Những tiêu chí phù hợp bao gồm độ chính xác của top-k, xếp hạng của item mục tiêu, và các chỉ số đo công sức tìm kiếm (số lần tương tác, thời gian hoàn thành nhiệm vụ). Trong phân tích cần lưu ý:

- **Độ tin cậy theo ngữ cảnh:** Truy hồi không chỉ đòi hỏi kết quả chính xác mà còn đòi hỏi tính ổn định theo thời gian — những thay đổi thuật toán nhỏ không nên dẫn đến mất mát đột ngột về khả năng tìm kiếm cho người dùng quen thuộc.
- **Khả năng tuỳ chỉnh:** Người dùng có nhu cầu khác nhau về độ rộng kết quả (breadth) so với độ sâu (depth) — giao diện nên cho phép điều khiển (filters, facets) để điều chỉnh hành vi hệ thống.
- **Độ tương tác tối thiểu:** Với các nhiệm vụ truy hồi, giảm số bước/tương tác cần thiết để đạt được mục tiêu là tiêu chí then chốt; do đó đánh giá UX task-based phải song hành với các đánh giá offline.

Thực nghiệm Đối với domain tập trung vào truy hồi, cần tiến hành testing với các bộ truy vấn thực tế, kèm ground-truth do chuyên gia kiểm định hoặc dựa trên hành vi lịch sử. User study dạng "information-seeking task" là quan trọng để đo lường chi phí tìm kiếm và mức độ hài lòng.



Hình 1.1: Minh họa hiện tượng long-tail trong gợi ý.

2. Khía cạnh gợi ý (Recommendation) – Khám phá và Long Tail

Mục tiêu phân tích Khía cạnh gợi ý hướng tới việc giới thiệu các item người dùng chưa biết tới, đặc biệt tập trung vào long-tail items (Hình 1.1) — những mục

ít phổ biến nhưng có thể phù hợp với sở thích cá nhân. Mục tiêu này nhấn mạnh khả năng mở rộng tầm nhìn cho người dùng, tăng tính đa dạng và hỗ trợ trải nghiệm khám phá.

Tiêu chí thành công và xung đột mục tiêu Phân tích tiêu chí ở khía cạnh này cần cân nhắc các tiêu chí sau:

- **Coverage và Long-tail exposure:** tần suất và tỷ lệ items thuộc long-tail xuất hiện trong đề xuất, trên toàn bộ người dùng và trên từng người dùng.
- **Novelty và Serendipity:** đánh giá mức độ "mới lạ" và bất ngờ có lợi mà đề xuất đem lại — những yếu tố này khó đo lường thuần số và thường yêu cầu thu thập dữ liệu định tính.
- **Giá trị luân chuyển:** long-tail items có thể không mang lại chuyển đổi ngay lập tức, nhưng đóng góp tới retention và sự hài lòng dài hạn; cần theo dõi các chỉ số dài hạn chứ không chỉ tương tác tức thời.

Xung đột điển hình: tăng exposure cho long-tail có thể làm giảm immediate accuracy (người dùng đợi khi nhận thấy nội dung "không liên quan"). Do đó, chiến lược cần có kiểm soát — ví dụ, kết hợp exploitation (đề xuất an toàn) với một phần exploration (thử nghiệm long-tail có kiểm soát), hoặc áp dụng cá nhân hóa tỉ lệ khám phá dựa trên lịch sử tương tác của từng user.

Thực nghiệm Dánh giá hiệu quả long-tail đòi hỏi kết hợp: phân tích log để đo coverage và distribution of exposure; thí nghiệm A/B để quan sát ảnh hưởng tới retention và engagement; và user interview để hiểu độ hài lòng khi khám phá nội dung mới. Do long-tail thường ảnh hưởng theo thời gian, nên cần các thử nghiệm có thời gian đủ dài để quan sát hiệu ứng muộn.

3. Khía cạnh dự đoán (Prediction)

Mục tiêu phân tích Khía cạnh dự đoán tập trung vào khả năng ước lượng mức độ người dùng sẽ thích item (rating hoặc likelihood). Đây là khuôn khổ truyền thống và dễ đo lường về mặt kỹ thuật, thích hợp cho việc so sánh thuật toán.

Tiêu chí thành công và cân nhắc Các chỉ số tiêu biểu cho khía cạnh này thường là các chỉ số lỗi dự đoán và hiệu suất xếp hạng. Tuy nhiên, trong phần phân tích cần nhấn mạnh:

- **Sự tương quan với KPI sản phẩm:** cải thiện các chỉ số dự đoán không luôn tương ứng trực tiếp với cải thiện KPIs kinh doanh; phân tích cần kiểm chứng mối liên hệ này thông qua thử nghiệm thực tế.
- **Sự nhạy cảm theo phân khúc người dùng:** mô hình có thể hoạt động tốt trên tập người dùng có nhiều tương tác nhưng kém với người dùng mới; cân nhắc đánh giá theo phân đoạn.
- **Độ công bằng và thiên vị:** mô hình dự đoán có thể cung cấp các item phổ biến và bỏ qua nhóm ít được tiếp cận; phân tích cần kiểm tra bias theo metadata (category, creator, vùng địa lý).

Thực nghiệm Bên cạnh các thử nghiệm offline tiêu chuẩn, cần áp dụng các phương pháp ước lượng counterfactual để đánh giá tác động khi không thể triển khai A/B testing trực tiếp, như technique dựa trên log policy và phương pháp giảm sai lệch trong ước tính.

4. Khía cạnh tương tác (Interaction)

Mục tiêu phân tích Tương tác người dùng không chỉ đơn thuần là số click hay thời lượng — nó liên quan tới trải nghiệm cảm xúc: cảm giác được hiểu, được tôn trọng quyền kiểm soát, và cảm thấy an tâm khi sử dụng. Hệ thống có thể đóng vai trò giáo dục, định hướng sở thích người dùng, và cần khả năng diễn giải để thuyết phục và xây dựng lòng tin.

Tiêu chí thành công và thách thức Các chỉ số liên quan đến tương tác bao gồm điểm hài lòng từ khảo sát, tần suất quay lại, và các chỉ số engagement có chiều sâu. Trong phân tích cần chú ý:

- **Explainability không phải chỉ là tính kỹ thuật:** Việc diễn giải cho user cần phù hợp ngữ cảnh — một lời giải thích quá kỹ thuật có thể gây rối, trong khi giải thích quá đơn giản có thể mất tính chính xác.
- **Quyền kiểm soát của người dùng:** cơ chế feedback (like/dislike, preference sliders) giúp người dùng cảm thấy họ có thể điều chỉnh đề xuất, và đồng thời cung cấp tín hiệu rõ ràng cho hệ thống.
- **Đo lường cảm xúc:** "good feeling" là một khái niệm đa chiều; các phương pháp phù hợp gồm khảo sát định kỳ, diary studies, và phân tích hành vi hậu A/B (ví dụ: liệu người dùng có quay lại ngay sau khi thấy gợi ý?).

Thực nghiệm Để phân tích khía cạnh tương tác, cần kết hợp data-driven metrics và qualitative research. Các thử nghiệm nên đo không chỉ sự thay đổi trong action metrics mà còn trong subjective metrics (độ hài lòng, cảm nhận về tính minh bạch). Những insights này thường giúp tinh chỉnh các thành phần giao diện người dùng và chính sách hiển thị đề xuất.

5. Khả năng chuyển đổi (Conversion)

Mục tiêu phân tích Khả năng chuyển đổi tập trung vào tối ưu các bước cuối của phễu (funnel) — từ hiển thị tới click, từ click tới mua/hoàn thành mục tiêu. Đây thường là mục tiêu chính trong các ứng dụng thương mại và dịch vụ trực tuyến.

Tiêu chí thành công và lưu ý Các chỉ số chuyển đổi (CTR, CR, doanh thu) là trực tiếp và dễ hiểu cho bên kinh doanh. Tuy nhiên, khi phân tích cần nhấn mạnh:

- **Chi phí và lợi ích ngắn hạn vs dài hạn:** tối ưu cho doanh thu tức thời có thể gây tổn hại tới retention; cần phân tích trade-off giữa lift tức thời và tác động dài hạn.
- **Trực quan hóa và phân tích nguyên nhân:** khi CTR thay đổi, cần phân tích sâu để xác định nguyên nhân (thay đổi thuật toán, seasonality, thay đổi interface).
- **Phân khúc theo giá trị:** tối ưu hoá dựa trên trọng số kinh tế (ví dụ: lợi nhuận biên trên mỗi item) thường hiệu quả hơn tối ưu thuần CTR.

Thực nghiệm A/B testing được coi là tiêu chuẩn vàng để đo chuyển đổi, nhưng cần thiết kế cẩn trọng (sample size, duration, kiểm soát yếu tố ngoại lai). Khi không thể A/B trực tiếp, các phương pháp khai thác lịch sử và mô hình hóa causal inference cần được sử dụng.

Vấn đề xuyên suốt: cold-start, exploration-exploitation, fairness và pháp lý

Cold-start Đối với user hoặc item mới, các chỉ số accuracy/coverage có thể rất khác biệt so với những phần còn lại của hệ thống. Phân tích cần tách riêng performance theo nhóm "cold-start" và "warm" để đưa ra chiến lược thích hợp: onboarding, content-based seeding, hay ưu tiên khảo sát sở thích ban đầu.

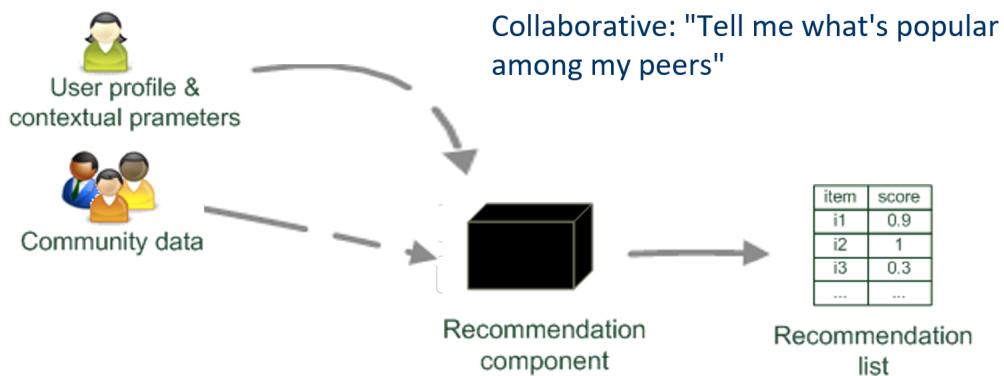
Exploration vs. Exploitation Chiến lược khám phá cần được đưa vào hệ thống một cách có kiểm soát. Việc đưa vào explore items để tăng long-tail exposure phải dựa trên chính sách có thể cân bằng rủi ro cho trải nghiệm người dùng.

Fairness, Bias và tuân thủ pháp lý Hệ thống gợi ý có thể khuếch đại những bất bình đẳng hiện có. Phân tích tiêu chí thành công nên bao gồm các ràng buộc về công bằng và tuân thủ quy định (ví dụ: khả năng giải thích theo yêu cầu pháp luật).Thêm vào đó, cần chỉ rõ cách đo lường bias theo các thuộc tính quan trọng (danh mục, tác giả, khu vực) và chính sách giảm thiểu.

Việc thiết kế và đánh giá hệ thống gợi ý phải bắt đầu từ một phân tích cẩn trọng về mục tiêu domain và sự ưu tiên giữa các chỉ tiêu. Không tồn tại một metric đơn lẻ nào có thể đại diện cho "thành công" chung cho mọi trạng thái; thay vào đó, cần một bộ các chỉ số phân tầng và một pipeline đánh giá kết hợp cả offline, counterfactual, online và qualitative studies. Bằng cách minh bạch về mục tiêu, phân đoạn người dùng và phương pháp thử nghiệm, nhà nghiên cứu và kỹ sư có thể đưa ra quyết định thiết kế phù hợp, cân bằng giữa trải nghiệm người dùng và hiệu quả kinh doanh.

1.2 Các phương pháp gợi ý chính

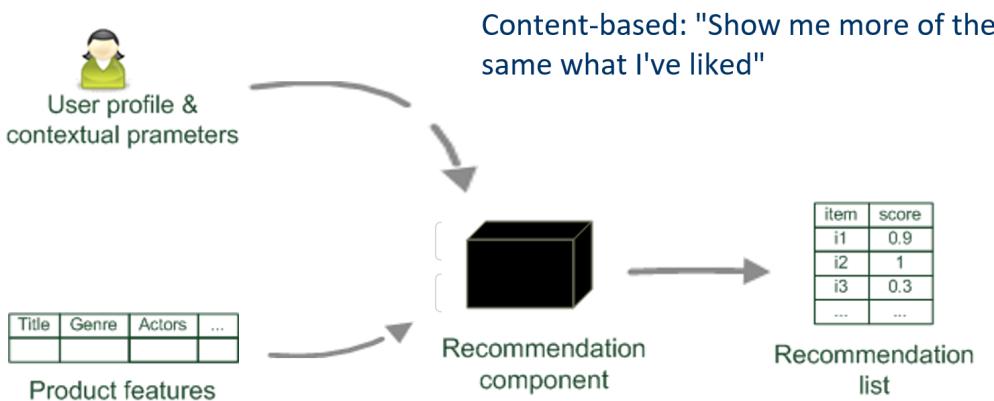
Có bốn phương pháp phổ biến trong việc xây dựng hệ gợi ý, bao gồm lọc cộng tác, lọc dựa trên nội dung, gợi ý dựa trên tri thức và phương pháp lai.



Hình 1.2: Minh họa lọc cộng tác.

1.2.1 Lọc cộng tác

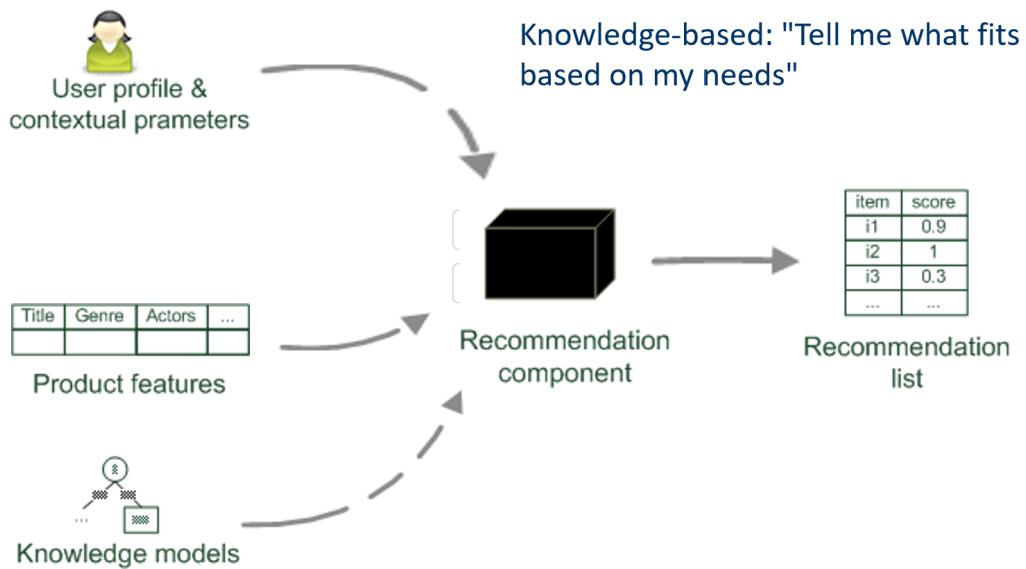
Lọc cộng tác là phương pháp được ứng dụng rộng rãi nhất. Nguyên lý của nó dựa trên giả định rằng những người có hành vi và sở thích tương tự nhau sẽ cùng thích các sản phẩm giống nhau (Hình 1.2). Phương pháp này có thể triển khai theo hướng dựa trên người dùng hoặc dựa trên sản phẩm. Trong cách tiếp cận dựa trên người dùng, hệ thống sẽ tìm những cá nhân có sở thích gần giống nhau, rồi gợi ý sản phẩm mà người này đã quan tâm cho người kia. Trong cách tiếp cận dựa trên sản phẩm, hệ thống tìm những sản phẩm thường được người dùng đánh giá cùng nhau, sau đó gợi ý các sản phẩm tương tự cho những ai đã chọn sản phẩm gốc. Lọc cộng tác có ưu điểm là không cần thông tin chi tiết về sản phẩm, tuy nhiên nó gặp khó khăn khi xuất hiện người dùng mới hoặc sản phẩm mới chưa có dữ liệu.



Hình 1.3: Minh họa lọc dựa trên nội dung.

1.2.2 Lọc dựa trên nội dung

Lọc dựa trên nội dung thì tập trung vào các đặc tính của sản phẩm (Hình 1.3). Hệ thống xây dựng hồ sơ người dùng dựa trên các sản phẩm họ đã thích, rồi so khớp với những sản phẩm có đặc điểm tương đồng. Ví dụ, nếu một người thường xem phim hành động, hệ thống sẽ đề xuất thêm nhiều phim thuộc thể loại này. Ưu điểm của phương pháp này là có thể gợi ý ngay cả khi sản phẩm mới chưa có ai đánh giá. Tuy nhiên, nó thường thiếu đa dạng vì chỉ gợi ý quanh một nhóm nội dung quen thuộc.



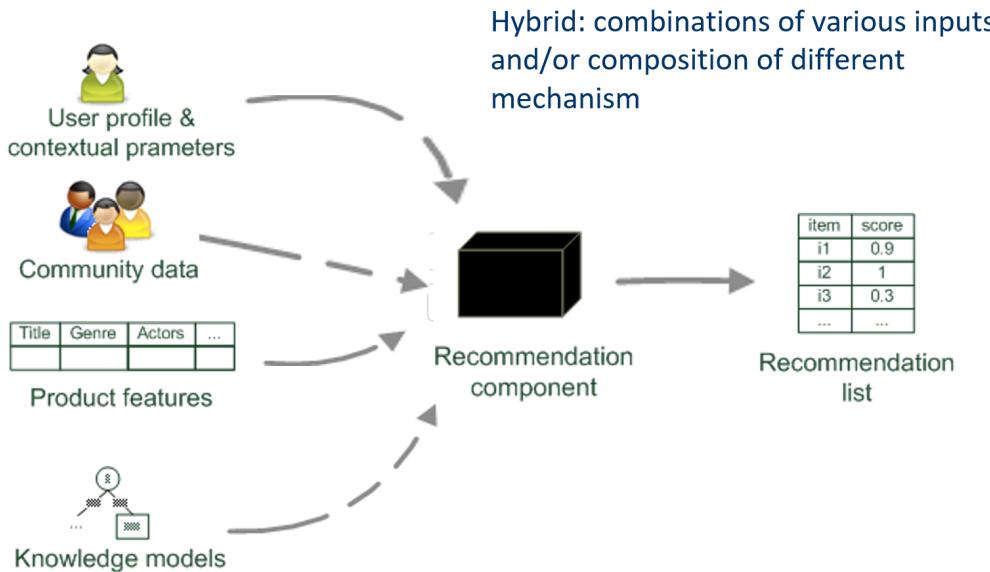
Hình 1.4: Minh họa gợi ý dựa trên tri thức.

1.2.3 Gợi ý dựa trên tri thức

Gợi ý dựa trên tri thức là phương pháp không phụ thuộc vào dữ liệu lịch sử, mà dựa vào nhu cầu cụ thể của người dùng và cơ sở tri thức đã được xây dựng (Hình 1.4). Người dùng đưa ra yêu cầu rõ ràng, chẳng hạn muốn mua một chiếc điện thoại có dung lượng pin cao và giá dưới mười triệu, hệ thống sẽ sử dụng dữ liệu tri thức để tìm sản phẩm phù hợp nhất. Phương pháp này có ưu điểm là tránh được vấn đề cold-start, nhưng nhược điểm là cần duy trì một cơ sở tri thức phong phú và chính xác.

1.2.4 Phương pháp lai

Phương pháp lai kết hợp nhiều kỹ thuật với nhau nhằm tận dụng ưu điểm và hạn chế nhược điểm (1.5). Chẳng hạn, Netflix kết hợp lọc cộng tác với lọc nội dung, nhờ đó vừa tận dụng được dữ liệu người dùng, vừa khai thác thông tin chi tiết về sản phẩm. Các hệ gợi ý hiện đại thường áp dụng mô hình lai để đạt được sự cân bằng giữa độ chính xác và tính đa dạng.



Hình 1.5: Minh họa gợi ý dựa trên lai ghép các chiến lược.

1.3 Các thách thức trong hệ gợi ý

Mặc dù mang lại nhiều lợi ích, hệ gợi ý cũng phải đối mặt với những thách thức lớn. Một trong những vấn đề phổ biến nhất là cold-start, khi hệ thống không có dữ liệu về người dùng mới hoặc sản phẩm mới. Trong trường hợp này, hệ thống khó đưa ra gợi ý chính xác.

Một vấn đề khác là dữ liệu thừa. Thông thường, mỗi người dùng chỉ đánh giá một số ít sản phẩm trong toàn bộ danh mục, dẫn đến ma trận dữ liệu rất loãng. Điều này khiến việc tính toán tương đồng và dự đoán trở nên kém chính xác.

Ngoài ra, việc thu thập dữ liệu người dùng cũng đặt ra thách thức về bảo mật và quyền riêng tư. Người dùng ngày càng quan tâm đến việc dữ liệu cá nhân của họ được sử dụng ra sao, và các hệ gợi ý cần có cơ chế minh bạch để bảo vệ thông tin này.

Cuối cùng, khả năng giải thích cũng là một vấn đề. Người dùng thường muốn biết tại sao hệ thống lại gợi ý một sản phẩm nào đó. Nếu hệ thống không thể đưa ra lời giải thích hợp lý, người dùng sẽ khó đặt niềm tin vào các đề xuất.

1.4 Ứng dụng trong thực tế và thương mại điện tử

Hệ thống gợi ý đã được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Trong thương mại điện tử, Amazon nổi tiếng với các gợi ý sản phẩm liên quan và gợi ý sản phẩm thường mua kèm. Trong lĩnh vực giải trí, Netflix sử dụng gợi ý để cung cấp danh sách phim “dành riêng cho bạn”, trong khi Spotify tạo playlist cá nhân hóa dựa trên thói quen nghe nhạc. Ngay cả trong mạng xã hội, Facebook hay TikTok cũng dựa vào hệ gợi ý để hiển thị bạn bè, nhóm cộng đồng hoặc video ngắn phù hợp nhất với từng người dùng.

Những ứng dụng này cho thấy hệ gợi ý đã trở thành một phần không thể thiếu trong trải nghiệm số. Chúng góp phần định hình hành vi tiêu dùng và làm thay đổi cách người dùng tiếp cận thông tin.

1.5 Xu hướng phát triển hệ gợi ý hiện đại

Hệ thống gợi ý đang không ngừng phát triển theo nhiều hướng mới. Một xu hướng nổi bật là gợi ý theo ngữ cảnh, tức là hệ thống tính đến các yếu tố như thời gian, địa điểm, thiết bị hay thậm chí tâm trạng của người dùng khi đưa ra đề xuất. Xu hướng thứ hai là gợi ý hội thoại, trong đó người dùng có thể tương tác trực tiếp với hệ thống qua chatbot để chỉnh sửa gợi ý. Ngoài ra, còn có hướng phát triển gợi ý cho nhóm, nhằm phục vụ nhiều người dùng cùng lúc, chẳng hạn gợi ý bộ phim cho cả gia đình.

Đặc biệt, với sự phát triển của trí tuệ nhân tạo và học sâu, các mô hình gợi ý ngày càng có khả năng khai thác dữ liệu phức tạp hơn, bao gồm cả văn bản, hình ảnh và âm thanh. Điều này mở ra tiềm năng to lớn để tạo ra những hệ gợi ý thông minh, chính xác và cá nhân hóa ở mức độ chưa từng có.

Kết luận

Hệ thống gợi ý là công cụ thiết yếu trong thế giới số hiện đại. Nó giúp người dùng vượt qua tình trạng quá tải thông tin, tìm thấy sản phẩm phù hợp và khám phá thêm nhiều nội dung mới. Đồng thời, nó cũng mang lại lợi ích lớn cho doanh nghiệp thông qua việc tăng doanh thu, giữ chân khách hàng và mở rộng danh mục sản phẩm.

Bốn phương pháp chính – lọc cộng tác, lọc dựa trên nội dung, gợi ý dựa trên tri thức và phương pháp lai – mỗi phương pháp đều có ưu và nhược điểm riêng, và sự

kết hợp thông minh các phương pháp này tạo nên sức mạnh vượt trội. Trong tương lai, với sự hỗ trợ của trí tuệ nhân tạo, dữ liệu lớn và học sâu, hệ thống gợi ý sẽ tiếp tục phát triển mạnh mẽ, ngày càng thông minh hơn, cá nhân hóa hơn và đóng vai trò trung tâm trong mọi dịch vụ trực tuyến.

Chương 2

GỢI Ý DỰA TRÊN LỌC CỘNG TÁC CƠ BẢN

2.1	Mở đầu	24
2.2	Cơ sở lý thuyết	25
2.3	Lọc cộng tác dựa trên bộ nhớ (Memory-based CF)	27
2.4	Độ thưa dữ liệu và vấn đề cold-start	31
2.5	Lọc cộng tác dựa trên mô hình: Phân tích ma trận cho hệ gợi ý	33
2.6	Ứng dụng thực tiễn	39
2.7	Hạn chế và thách thức	40

2.1 Mở đầu

Trong kỷ nguyên dữ liệu lớn và thương mại điện tử, hệ thống gợi ý đóng vai trò quan trọng trong việc cá nhân hóa trải nghiệm người dùng. Một trong những phương pháp cốt lõi và lâu đời nhất được sử dụng trong các hệ thống gợi ý chính là **lọc cộng tác** (Collaborative Filtering – CF).

Ý tưởng cơ bản của CF dựa trên giả định rằng: *nếu hai người dùng có hành vi hoặc sở thích giống nhau trong quá khứ, họ cũng sẽ có xu hướng đánh giá hoặc quan tâm đến những mục giống nhau trong tương lai.* Từ giả định này, CF sử dụng dữ liệu tương tác của cộng đồng để đưa ra dự đoán và gợi ý.

Phương pháp CF đã được áp dụng thành công trong nhiều hệ thống quy mô lớn:

từ gợi ý sản phẩm của Amazon, gợi ý phim của Netflix, đến gợi ý bạn bè hoặc nội dung trên mạng xã hội như Facebook, YouTube. Nhờ khả năng học từ dữ liệu người dùng thực tế mà không cần phân tích chi tiết nội dung sản phẩm, CF có thể tạo ra các gợi ý bất ngờ, mới lạ, giúp người dùng khám phá những sản phẩm mà họ chưa từng nghĩ đến.

Tuy nhiên, CF cũng tồn tại nhiều thách thức: hiện tượng dữ liệu thưa (sparse data), vấn đề người dùng hoặc sản phẩm mới (cold-start), cũng như khó khăn trong việc mở rộng hệ thống khi số lượng người dùng và sản phẩm tăng mạnh. Vì vậy, nghiên cứu về CF không chỉ dừng lại ở các thuật toán cơ bản mà còn mở rộng sang nhiều hướng phát triển hiện đại như phân rã ma trận (matrix factorization), học sâu (deep learning), và phương pháp lai (hybrid methods).

Trong chương này, chúng ta sẽ tìm hiểu chi tiết về cơ sở lý thuyết của CF, các thuật toán chính, ví dụ minh họa, ứng dụng thực tiễn, ưu điểm và hạn chế, cũng như các cải tiến hiện đại.

2.2 Cơ sở lý thuyết

2.2.1 Ma trận người dùng – sản phẩm

Trung tâm của CF là một ma trận gọi là **ma trận người dùng – sản phẩm** R . Giả sử có m người dùng và n sản phẩm, ta xây dựng ma trận $R \in \mathbb{R}^{m \times n}$, trong đó mỗi phần tử $R_{u,i}$ thể hiện hành vi của người dùng u đối với sản phẩm i .

Các dạng biểu diễn của $R_{u,i}$ có thể bao gồm:

- Điểm đánh giá (rating), ví dụ trong thang điểm 1–5.
- Giá trị nhị phân (0/1) thể hiện hành vi: có/không mua, có/không xem.
- Các dạng khác như số lần click, thời gian xem, hoặc số lượt nghe.

Một đặc điểm quan trọng của ma trận này là **tính thưa**. Thông thường, mỗi người dùng chỉ đánh giá một phần rất nhỏ trong toàn bộ tập sản phẩm. Do đó, phần lớn các ô trong ma trận R là trống (không có dữ liệu). Chính sự thưa thớt này làm tăng độ khó của bài toán dự đoán.

2.2.2 Chuẩn hóa dữ liệu

Trước khi tính toán độ tương đồng, thường cần chuẩn hóa dữ liệu để loại bỏ các khác biệt cá nhân trong thang đánh giá. Ví dụ, có người thường chấm điểm rất cao

(4–5), có người chấm điểm rất thấp (1–2), dù sở thích của họ có thể giống nhau. Để giảm sai lệch, người ta thường sử dụng:

- Chuẩn hoá theo trung bình: trừ đi giá trị trung bình \bar{r}_u của mỗi người dùng.
- Chuẩn hoá theo độ lệch chuẩn: chia thêm cho σ_u để đưa về cùng thang đo.

Ví dụ: nếu $u1$ thường cho điểm cao (luôn 4–5) còn $u2$ cho điểm thấp (2–3), nhưng xu hướng họ đánh giá cao cùng sản phẩm vẫn giống nhau. Sau chuẩn hoá, sự tương đồng này sẽ hiện rõ.

2.2.3 Độ tương đồng giữa vector

Để so sánh hai người dùng (hoặc hai sản phẩm), ta biểu diễn họ dưới dạng vector gồm các đánh giá tương ứng. Hai thước đo phổ biến nhất là **Cosine similarity** và **Pearson correlation**.

Cosine similarity

Cho hai vector x và y , tương đồng cosine được định nghĩa:

$$\text{sim}_{\text{cos}}(x, y) = \frac{\sum_{s \in S_{xy}} x_s y_s}{\sqrt{\sum_{s \in S_{xy}} x_s^2} \sqrt{\sum_{s \in S_{xy}} y_s^2}}, \quad (2.1)$$

trong đó S_{xy} là tập các sản phẩm mà cả hai đều đã đánh giá.

Pearson correlation

Hệ số tương quan Pearson được định nghĩa:

$$\text{sim}_{\text{Pearson}}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}}. \quad (2.2)$$

Các độ đo khác

Ngoài cosine và Pearson, có thể dùng:

- **Khoảng cách Euclid:** $\text{dist}(x, y) = \sqrt{\sum (x_s - y_s)^2}$.
- **Jaccard similarity** (cho dữ liệu nhị phân):

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Mỗi độ đo có ưu và nhược điểm, lựa chọn phụ thuộc vào loại dữ liệu và mục tiêu ứng dụng.

2.2.4 Phân loại lọc cộng tác

Có nhiều cách phân loại các phương pháp lọc cộng tác, trong đó nổi bật nhất là phân chia theo cách xây dựng mô hình (memory-based hoặc model-based).

Memory-based: sử dụng trực tiếp ma trận R để tính toán độ tương đồng và dự đoán. Bao gồm User-User kNN và Item-Item kNN. Phương pháp này đơn giản, trực quan, nhưng kém hiệu quả khi dữ liệu lớn.

Model-based: xây dựng mô hình dự đoán dựa trên dữ liệu R . Ví dụ: phân rã ma trận, mô hình xác suất, hoặc học sâu. Ưu điểm là có khả năng khái quát và mở rộng tốt, nhưng phức tạp hơn về mặt thuật toán.

2.3 Lọc cộng tác dựa trên bộ nhớ (Memory-based CF)

Trong phần này chúng ta sẽ xem xét các phương pháp lọc cộng tác theo bộ nhớ.

2.3.1 Lọc cộng tác dựa trên người dùng (User-based CF)

Trong User-based CF, ta so sánh các người dùng với nhau để tìm ra những cá nhân có hành vi tương đồng. Giả định ở đây là: *nếu người dùng u có sở thích tương tự người dùng v , và v đã thích sản phẩm i , thì khả năng cao u cũng sẽ thích i .*

Cách tiếp cận này yêu cầu:

1. Xác định tập láng giềng N_u của người dùng u , gồm những người có độ tương đồng cao với u .
2. Dự đoán điểm đánh giá $\hat{r}_{u,i}$ bằng cách tổng hợp đánh giá của các láng giềng đối với sản phẩm i .

Trong User-based CF với k-nearest neighbors (kNN), ta chọn ra k người dùng có độ tương đồng cao nhất với người dùng u . Công thức dự đoán đánh giá cho sản phẩm i :

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N_u} \text{sim}(u, v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in N_u} |\text{sim}(u, v)|}. \quad (2.3)$$

Trong đó:

- \bar{r}_u : trung bình đánh giá của người dùng u .
- N_u : tập k lảng giềng gần nhất của u .
- $\text{sim}(u, v)$: độ tương đồng giữa u và v .

Công thức này giúp loại bỏ sai lệch do mỗi người có xu hướng cho điểm khác nhau.

Ưu điểm: dễ hiểu, dễ cài đặt. Nhược điểm: khó mở rộng khi số người dùng lớn, và dễ bị nhiễu bởi các hành vi bất thường.

2.3.2 Lọc cộng tác dựa trên sản phẩm (Item-based CF)

Khác với User-based, Item-based CF so sánh các sản phẩm với nhau. Giả định: *nếu sản phẩm i và j được nhiều người dùng cùng đánh giá tương tự, thì một người dùng đã thích i cũng có khả năng thích j .*

Quy trình:

1. Xác định tập lảng giềng N_i của sản phẩm i , gồm những sản phẩm có độ tương đồng cao.
2. Với một người dùng u , dự đoán điểm đánh giá $\hat{r}_{u,i}$ dựa trên các sản phẩm tương tự j mà u đã đánh giá.

Trong Item-based CF, ta quan tâm đến tương đồng giữa sản phẩm. Với một sản phẩm i , tìm tập N_i gồm k sản phẩm tương tự. Công thức dự đoán:

$$\hat{r}_{u,i} = \frac{\sum_{j \in N_i} \text{sim}(i, j) r_{u,j}}{\sum_{j \in N_i} |\text{sim}(i, j)|}. \quad (2.4)$$

Trong đó $r_{u,j}$ là đánh giá của người dùng u với sản phẩm j .

Ưu điểm của Item-based kNN là ma trận tương đồng giữa các sản phẩm ít thay đổi theo thời gian (ổn định hơn người dùng), do đó tính toán một lần có thể dùng lâu dài.

2.3.3 Clustering-based CF

Một cách tiếp cận khác là phân cụm người dùng (hoặc sản phẩm) thành các nhóm tương tự nhau. Sau đó, dự đoán dựa trên trung bình nhóm. Ví dụ, nếu người dùng u thuộc cụm C , thì điểm dự đoán $\hat{r}_{u,i}$ có thể lấy từ trung bình đánh giá của cụm C đối với sản phẩm i .

Phương pháp này giảm chi phí tính toán, nhưng có thể mất độ chính xác do gom nhóm.

2.3.4 Ví dụ minh họa chi tiết

Để hiểu rõ cơ chế hoạt động của lọc cộng tác, chúng ta cùng xét một ví dụ nhỏ với ma trận người dùng – sản phẩm. Giả sử có 4 người dùng ($u1, u2, u3, u4$) và 5 sản phẩm ($i1, i2, i3, i4, i5$). Ma trận đánh giá R như sau:

	$i1$	$i2$	$i3$	$i4$	$i5$
$u1$	5	3	4	4	?
$u2$	3	1	2	3	3
$u3$	4	3	4	3	5
$u4$	3	3	1	5	4

Bảng 2.1: Ma trận đánh giá giả định (dấu ? là giá trị cần dự đoán).

Nhiệm vụ: dự đoán điểm đánh giá mà người dùng $u1$ sẽ cho sản phẩm $i5$.

Tính tương đồng giữa người dùng

Cosine similarity giữa $u1$ và $u3$

Xét các sản phẩm chung mà cả $u1$ và $u3$ đều đã đánh giá: $\{i1, i2, i3, i4\}$.

Vector:

$$u1 = (5, 3, 4, 4), \quad u3 = (4, 3, 4, 3).$$

Khi đó:

$$\text{sim}_{\cos}(u1, u3) = \frac{5 \cdot 4 + 3 \cdot 3 + 4 \cdot 4 + 4 \cdot 3}{\sqrt{5^2 + 3^2 + 4^2 + 4^2} \cdot \sqrt{4^2 + 3^2 + 4^2 + 3^2}}.$$

Tính toán:

$$= \frac{20 + 9 + 16 + 12}{\sqrt{66} \cdot \sqrt{50}} = \frac{57}{\sqrt{3300}} \approx 0.99.$$

Như vậy $u1$ và $u3$ có mức tương đồng rất cao.

Pearson correlation giữa $u1$ và $u2$

Trước hết tính trung bình:

$$\bar{r}_{u1} = \frac{5 + 3 + 4 + 4}{4} = 4, \quad \bar{r}_{u2} = \frac{3 + 1 + 2 + 3}{4} = 2.25.$$

Sử dụng công thức:

$$\text{sim}_{\text{Pearson}}(u1, u2) = \frac{\sum_{i=1}^4 (r_{u1,i} - \bar{r}_{u1})(r_{u2,i} - \bar{r}_{u2})}{\sqrt{\sum(r_{u1,i} - \bar{r}_{u1})^2} \cdot \sqrt{\sum(r_{u2,i} - \bar{r}_{u2})^2}}.$$

Tính tử số:

$$(5 - 4)(3 - 2.25) + (3 - 4)(1 - 2.25) + (4 - 4)(2 - 2.25) + (4 - 4)(3 - 2.25)$$

$$= 1 \cdot 0.75 + (-1) \cdot (-1.25) + 0 \cdot (-0.25) + 0 \cdot 0.75 = 0.75 + 1.25 = 2.$$

Tính mẫu số:

$$\sqrt{(1^2 + (-1)^2 + 0^2 + 0^2)} \cdot \sqrt{(0.75^2 + (-1.25)^2 + (-0.25)^2 + 0.75^2)}.$$

$$= \sqrt{2} \cdot \sqrt{0.5625 + 1.5625 + 0.0625 + 0.5625} = \sqrt{2} \cdot \sqrt{2.75} \approx 1.414 \cdot 1.658 = 2.345.$$

Vậy:

$$\text{sim}_{\text{Pearson}}(u1, u2) \approx \frac{2}{2.345} \approx 0.85.$$

Dự đoán User-based CF

Giả sử chọn $u3$ và $u2$ làm láng giềng gần nhất của $u1$. Điểm dự đoán cho $i5$:

$$\hat{r}_{u1,i5} = \bar{r}_{u1} + \frac{\text{sim}(u1, u3)(r_{u3,i5} - \bar{r}_{u3}) + \text{sim}(u1, u2)(r_{u2,i5} - \bar{r}_{u2})}{|\text{sim}(u1, u3)| + |\text{sim}(u1, u2)|}.$$

Thay số:

$$\bar{r}_{u1} = 4, \quad \bar{r}_{u3} = \frac{4+3+4+3+5}{5} = 3.8, \quad \bar{r}_{u2} = 2.25.$$

$$= 4 + \frac{0.99(5 - 3.8) + 0.85(3 - 2.25)}{0.99 + 0.85}.$$

$$= 4 + \frac{0.99 \cdot 1.2 + 0.85 \cdot 0.75}{1.84}.$$

$$= 4 + \frac{1.188 + 0.638}{1.84} = 4 + \frac{1.826}{1.84} \approx 4.99.$$

Kết quả: $u1$ được dự đoán sẽ cho $i5$ số điểm gần 5.

Dự đoán Item-based CF

Ta tính tương đồng giữa i_5 với các sản phẩm mà u_1 đã đánh giá. Giả sử i_5 có tương đồng cao nhất với i_3 và i_4 . Khi đó:

$$\hat{r}_{u1,i5} = \frac{\text{sim}(i_5, i_3) \cdot r_{u1,i3} + \text{sim}(i_5, i_4) \cdot r_{u1,i4}}{|\text{sim}(i_5, i_3)| + |\text{sim}(i_5, i_4)|}.$$

Ví dụ, nếu $\text{sim}(i_5, i_3) = 0.9$, $\text{sim}(i_5, i_4) = 0.7$, $r_{u1,i3} = 4$, $r_{u1,i4} = 4$, thì:

$$\hat{r}_{u1,i5} = \frac{0.9 \cdot 4 + 0.7 \cdot 4}{0.9 + 0.7} = \frac{3.6 + 2.8}{1.6} = 6.4/1.6 = 4.0.$$

Như vậy, Item-based CF dự đoán u_1 sẽ cho i_5 điểm khoảng 4.0, thấp hơn so với User-based CF.

So sánh kết quả

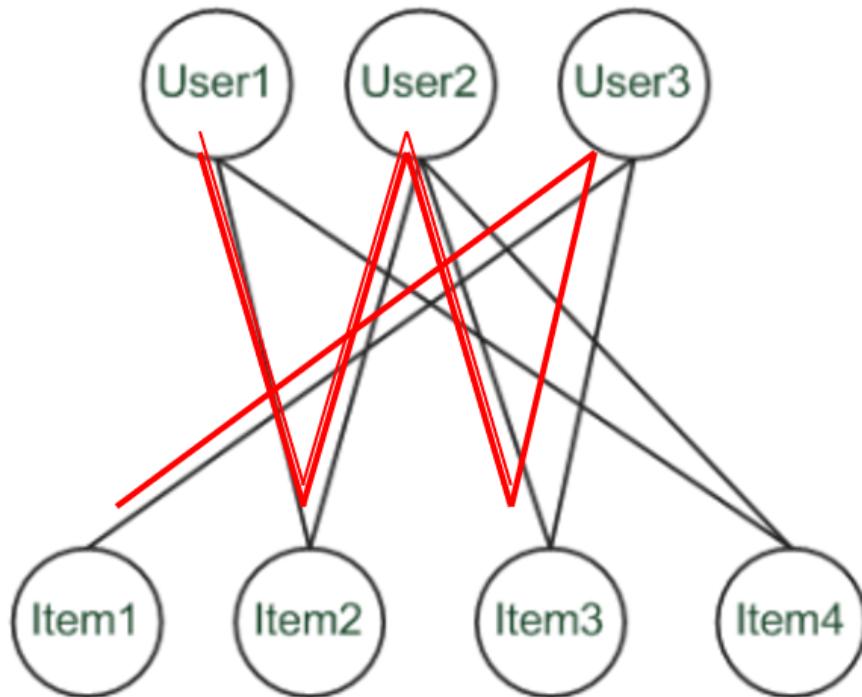
Trong ví dụ này, User-based CF dự đoán ≈ 4.99 (rất cao), trong khi Item-based CF dự đoán ≈ 4.0 . Trên thực tế, tùy vào đặc tính dữ liệu, một trong hai phương pháp có thể chính xác hơn. Đây cũng là lý do tại sao nhiều hệ thống kết hợp cả hai cách tiếp cận.

2.4 Độ thừa dữ liệu và vấn đề cold-start

Trong các ma trận đánh giá được minh họa ở ví dụ trước, hầu như mọi cặp người dùng – sản phẩm đều có giá trị. Tuy nhiên, trong các ứng dụng thực tế, ma trận đánh giá thường rất thừa, bởi vì người dùng chỉ đánh giá hoặc mua một tỷ lệ nhỏ trong toàn bộ danh mục sản phẩm. Thách thức đặt ra là làm thế nào để đưa ra dự đoán chính xác khi số lượng đánh giá sẵn có còn hạn chế. Một cách tiếp cận đơn giản là khai thác thêm thông tin bổ sung về người dùng, chẳng hạn như giới tính, độ tuổi, học vấn, sở thích hay các dữ liệu nhân khẩu học khác. Tập hợp những người dùng tương tự (làng giềng) khi đó không chỉ dựa trên phân tích đánh giá tường minh hoặc ngầm, mà còn kết hợp với thông tin bên ngoài. Tuy nhiên, khi sử dụng các thông tin này, hệ thống sẽ không còn thuần tuý dựa trên cộng tác nữa và phát sinh những vấn đề về cách thu thập, kết hợp các nguồn dữ liệu khác nhau. Mặc dù vậy, ở giai đoạn khởi động của một hệ thống gợi ý, những kỹ thuật này có thể hữu ích để đạt được số lượng người dùng tối hạn.

Nhiều phương pháp đã được đề xuất để xử lý vấn đề khởi động lạnh (cold-start) và dữ liệu thừa [34, 12]. Một ví dụ điển hình là phương pháp dựa trên đồ thị [12].

Ý tưởng chính của phương pháp này là khai thác tính chất “bắc cầu” trong sở thích của người dùng, từ đó bổ sung thêm thông tin vào ma trận đánh giá. Thay vì chỉ xem xét mối quan hệ trực tiếp trong ma trận, phương pháp này biểu diễn dữ liệu thành một đồ thị hai phía (người dùng – sản phẩm) và sử dụng các đường đi trong đồ thị để suy luận mối liên kết gián tiếp. Ví dụ, trong gợi ý cộng tác thông thường, ta chỉ xét các đường đi dài ba bước (người dùng – sản phẩm – người dùng – sản phẩm). Tuy nhiên, khi dữ liệu thưa, số lượng đường đi này rất ít, nên việc mở rộng sang đường đi dài hơn (năm bước, bảy bước, ...) giúp phát hiện thêm nhiều liên kết gián tiếp và cải thiện chất lượng gợi ý (Hình 2.1). Ngoài ra, một loại thông tin thứ ba (“sản phẩm tương tự được đánh giá bởi những người dùng tương tự”) cũng được kết hợp trong hàm dự đoán, và các kết quả ban đầu cho thấy độ chính xác được cải thiện đáng kể trong các tập dữ liệu thưa.



Hình 2.1: Minh họa lan truyền sở thích trên đồ thị tương tác.

Vấn đề khởi động lạnh có thể được xem như một trường hợp đặc biệt của hiện tượng thưa dữ liệu. Nó bao gồm hai tình huống: (a) làm thế nào để gợi ý cho người dùng mới chưa từng đánh giá sản phẩm nào, và (b) làm thế nào để xử lý các sản phẩm mới chưa có ai mua hoặc đánh giá. Các cách tiếp cận lai, tận dụng thêm thông

tin bên ngoài, là giải pháp phổ biến. Ngoài ra, một chiến lược khác là yêu cầu người dùng mới đưa ra một số lượng đánh giá tối thiểu trước khi được sử dụng hệ thống. Việc lựa chọn các sản phẩm để hỏi có thể được tối ưu dựa trên lý thuyết thông tin, nhằm thu thập được nhiều thông tin nhất từ ít đánh giá nhất. Một số thuật toán, chẳng hạn Eigentaste, cũng áp dụng chiến lược này bằng cách yêu cầu người dùng đánh giá một tập chuẩn sản phẩm để khởi tạo hồ sơ.

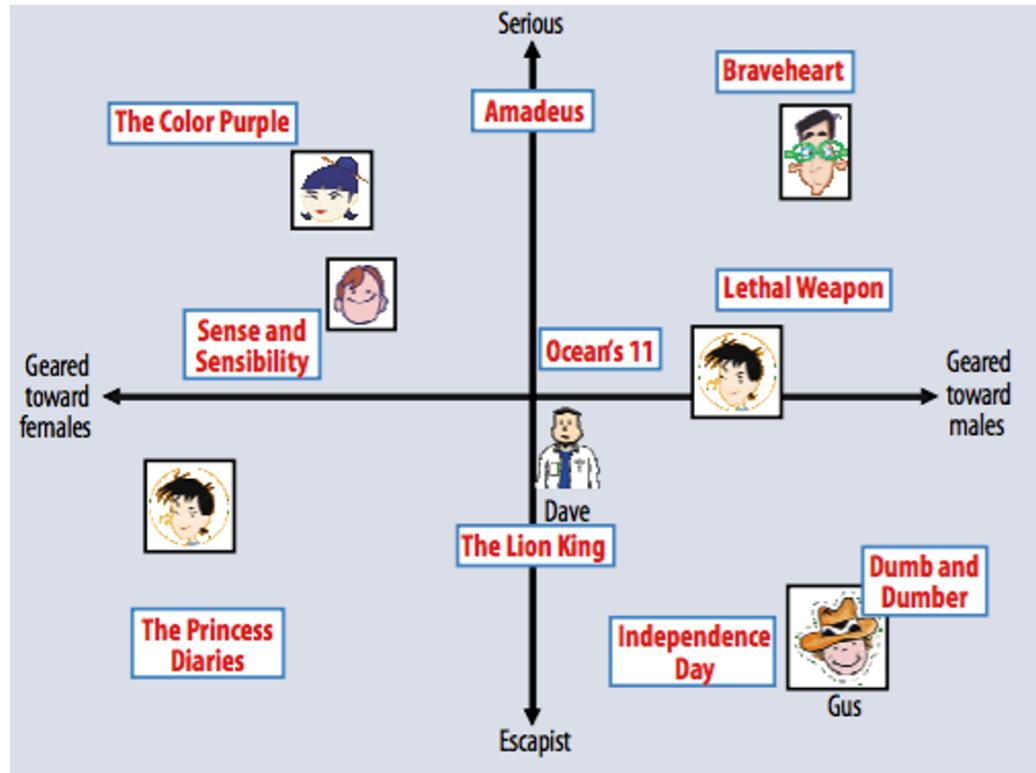
Để giảm chi phí tính toán khi phân tích các mối liên kết xa, ma trận đánh giá có thể được biến đổi thành đồ thị hai phía và áp dụng kỹ thuật kích hoạt lan truyền (spreading activation). Phương pháp này cho phép duyệt đồ thị hiệu quả và khai thác các mối liên hệ gián tiếp. So với các thuật toán gợi ý cộng tác truyền thống, phương pháp này cho thấy chất lượng dự đoán tăng rõ rệt, đặc biệt khi dữ liệu rất thưa hoặc khi xử lý người dùng mới. Tuy nhiên, khi mật độ dữ liệu đạt tới một ngưỡng nhất định, chất lượng có thể suy giảm so với các thuật toán cơ bản, và chi phí tính toán của việc xét mối liên hệ xa vẫn còn là một hạn chế khi áp dụng trên quy mô lớn.

Một kỹ thuật khác là “bỏ phiếu mặc định”. Thay vì chỉ tính toán độ tương tự dựa trên các sản phẩm mà cả hai người dùng đều đã đánh giá, hệ thống gán thêm các giá trị mặc định cho những sản phẩm mà chỉ một trong hai người dùng đã đánh giá. Cách làm này giúp triệt tiêu tác động của những điểm tương đồng khác biệt ngẫu nhiên khi số lượng đánh giá chung quá ít, từ đó cải thiện chất lượng dự đoán trên tập dữ liệu thưa.

Gần đây, một hướng tiếp cận khác dựa trên quan sát rằng hầu hết các hệ gợi ý cộng tác chỉ khai thác một phần thông tin trong ma trận đánh giá, hoặc là sự tương đồng giữa người dùng, hoặc là sự tương đồng giữa sản phẩm. Bằng cách kết hợp cả hai loại thông tin này, hệ thống có thể tăng độ chính xác trong dự đoán, đặc biệt khi đối mặt với vấn đề dữ liệu thưa.

2.5 Lọc cộng tác dựa trên mô hình: Phân tích ma trận cho hệ gợi ý

Phân tích ma trận (matrix factorization -MF) [15] là một trong những phương pháp cơ bản trong lọc cộng tác dựa trên mô hình. Phân tích ma trận hướng tới học biểu diễn user và item trên cùng không gian đặc trưng (minh họa trong Hình 2.2).



Hình 2.2: Minh họa user và item được biểu diễn trên cùng không gian đặc trưng.

2.5.1 Giới thiệu và mô hình cơ bản

Giả sử có m người dùng và n sản phẩm. Tập các cặp quan sát được ký hiệu $\Omega \subseteq \{1, \dots, m\} \times \{1, \dots, n\}$. Với mỗi cặp $(u, i) \in \Omega$ ta quan sát được giá trị đánh giá (rating) $r_{ui} \in \mathbb{R}$ (hoặc một dạng phản hồi tương tự). Mục tiêu là ước lượng các giá trị r_{ui} chưa quan sát.

Trong mô hình phân tích ma trận (matrix factorization) ta biểu diễn mỗi người dùng u bằng véc-tơ nhân tố

$$\mathbf{w}_u \in \mathbb{R}^r,$$

và mỗi sản phẩm i bằng véc-tơ nhân tố

$$\mathbf{h}_i \in \mathbb{R}^r,$$

với $r \ll \min(m, n)$ là số chiều tiềm ẩn (latent factors). Giá trị dự đoán cho cặp (u, i) được mô tả bằng tích vô hướng

$$v_{ui} = \mathbf{w}_u^\top \mathbf{h}_i.$$

Bài toán học là tìm ma trận $W = [\mathbf{w}_1^\top; \dots; \mathbf{w}_m^\top] \in \mathbb{R}^{m \times r}$ và $H = [\mathbf{h}_1^\top; \dots; \mathbf{h}_n^\top] \in \mathbb{R}^{n \times r}$ sao cho $R \approx WH^\top$ trên phần tử quan sát được.

Một biểu diễn mục tiêu tối giản là tối thiểu hóa lỗi bình phương có điều chuẩn:

$$J(W, H) = \sum_{(u,i) \in \Omega} (r_{ui} - \mathbf{w}_u^\top \mathbf{h}_i)^2 + \lambda \left(\sum_{u=1}^m \|\mathbf{w}_u\|_2^2 + \sum_{i=1}^n \|\mathbf{h}_i\|_2^2 \right),$$

trong đó $\lambda > 0$ là hệ số điều chỉnh nhằm giảm quá khớp.

Trong thực tế thường đưa thêm thành phần thiên lệch (biases): trung bình toàn cục μ , thiên lệch người dùng b_u và thiên lệch item b_i , khi đó mô hình dự đoán là

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{w}_u^\top \mathbf{h}_i,$$

và hàm mất mát tương ứng mở rộng bằng cách cộng các điều chỉnh cho b_u, b_i .

2.5.2 Cập nhật nghiệm đóng: Alternating Least Squares (ALS)

Hàm mục tiêu $J(W, H)$ là hàm lỗi theo từng khối tham số riêng lẻ (theo W khi H cố định, và theo H khi W cố định). Ý tưởng của phương pháp Alternating Least Squares (ALS) [24] là tối ưu luân phiên: cố định H rồi giải bài toán tối ưu cho từng \mathbf{w}_u (độc lập giữa các u), sau đó cố định W và giải cho từng \mathbf{h}_i .

Ta trình bày nghiệm đóng cho trường hợp không có bias (phiên bản mở rộng với bias sẽ trình bày bên dưới). Với H cố định, bài toán đối với mỗi người dùng u là:

$$\min_{\mathbf{w}_u} \sum_{i \in \Omega_u} (r_{ui} - \mathbf{w}_u^\top \mathbf{h}_i)^2 + \lambda \|\mathbf{w}_u\|_2^2,$$

trong đó $\Omega_u = \{i \mid (u, i) \in \Omega\}$ là tập các item mà user u đã đánh giá.

Viết ma trận con $H_{\Omega_u} \in \mathbb{R}^{|\Omega_u| \times r}$ sao cho mỗi hàng của H_{Ω_u} là véc-tơ \mathbf{h}_i^\top với $i \in \Omega_u$, và vector $\mathbf{r}_u \in \mathbb{R}^{|\Omega_u|}$ chứa các r_{ui} . Hàm mục tiêu theo \mathbf{w}_u là

$$\|\mathbf{r}_u - H_{\Omega_u} \mathbf{w}_u\|_2^2 + \lambda \|\mathbf{w}_u\|_2^2.$$

Lấy đạo hàm theo \mathbf{w}_u và đặt bằng 0 thu được phương trình tuyến tính (normal equation):

$$(H_{\Omega_u}^\top H_{\Omega_u} + \lambda I_r) \mathbf{w}_u = H_{\Omega_u}^\top \mathbf{r}_u.$$

Do ma trận bên trái là một ma trận dương xác định (thêm λI đảm bảo dương xác định), ta có nghiệm duy nhất:

$$\mathbf{w}_u = (H_{\Omega_u}^\top H_{\Omega_u} + \lambda I_r)^{-1} H_{\Omega_u}^\top \mathbf{r}_u.$$

Tương tự, khi cố định W , với mỗi item i có tập người dùng $\Omega^i = \{u \mid (u, i) \in \Omega\}$, ta xây dựng ma trận $W_{\Omega^i} \in \mathbb{R}^{|\Omega^i| \times r}$ (hàng là \mathbf{w}_u^\top), và vector $\mathbf{r}^i \in \mathbb{R}^{|\Omega^i|}$ chứa các r_{ui} . Nghiệm cho \mathbf{h}_i là:

$$(W_{\Omega^i}^\top W_{\Omega^i} + \lambda I_r) \mathbf{h}_i = W_{\Omega^i}^\top \mathbf{r}^i,$$

hay

$$\mathbf{h}_i = (W_{\Omega^i}^\top W_{\Omega^i} + \lambda I_r)^{-1} W_{\Omega^i}^\top \mathbf{r}^i.$$

Thuật toán ALS luân phiên giải các hệ tuyến tính trên cho mọi u và mọi i cho tới khi hội tụ (hoặc đạt số vòng lặp tối đa). Trong cài đặt lớn, thay vì tính nghịch đảo đầy đủ, người ta giải hệ bằng phương pháp Cholesky vì r thường nhỏ; đồng thời có thể song song hóa cập nhật cho nhiều \mathbf{w}_u (hoặc nhiều \mathbf{h}_i) độc lập.

Mở rộng: ALS với bias Nếu mô hình có bias μ, b_u, b_i , ta có thể xử lý bằng hai cách: (i) tối ưu riêng các bias (chẳng hạn bằng việc ước lượng μ, b_u, b_i trước rồi cố định khi học W, H), hoặc (ii) ghép bias vào véc-tơ nhân tố bằng cách mở rộng chiều thêm 1: định nghĩa $\tilde{\mathbf{w}}_u = [\mathbf{w}_u^\top, b_u]^\top \in \mathbb{R}^{r+1}$ và $\tilde{\mathbf{h}}_i = [\mathbf{h}_i^\top, 1]^\top \in \mathbb{R}^{r+1}$, khi đó

$$\mu + b_u + b_i + \mathbf{w}_u^\top \mathbf{h}_i = \tilde{\mathbf{w}}_u^\top \tilde{\mathbf{h}}_i + \text{const.}$$

và các công thức ALS tương tự áp dụng cho véc-tơ mở rộng (cần điều chỉnh điều chuẩn để không quá ưu hóa phần bias).

2.5.3 Phiên bản cho phản hồi ngầm: Weighted ALS (WALS)

Với dữ liệu phản hồi ngầm (implicit feedback) như lượt xem, số lần click, Hu et al. đề xuất mô hình có trọng số tin cậy. Ta chuyển sang biến nhị phân $p_{ui} \in \{0, 1\}$ biểu diễn sở thích (1 nếu user u từng tương tác item i , ngược lại 0), và gán trọng số tin cậy $c_{ui} \geq 1$ tăng theo mức tương tác (ví dụ $c_{ui} = 1 + \alpha \cdot \text{count}_{ui}$). Hàm mục tiêu là

$$\min_{X, Y} \sum_{u, i} c_{ui} (p_{ui} - x_u^\top y_i)^2 + \lambda \left(\sum_u \|x_u\|_2^2 + \sum_i \|y_i\|_2^2 \right).$$

Với ký hiệu $X = [x_1^\top; \dots; x_m^\top]$ và $Y = [y_1^\top; \dots; y_n^\top]$, ALS dẫn tới hệ:

$$(Y^\top C_u Y + \lambda I) x_u = Y^\top C_u p_u,$$

với $C_u = \text{diag}(c_{u1}, \dots, c_{un})$ và $p_u = (p_{u1}, \dots, p_{un})^\top$. Giải pháp tương tự cho y_i . Vì C_u là ma trận đường chéo, tính toán có thể tối ưu bằng cách tách hai phần và chỉ xét các phần tử có trọng số lớn.

2.5.4 Ưu/nhược điểm của ALS (nghiệm đóng)

- ALS cho nghiệm ngon cho từng khối và dễ song song hóa: có thể cập nhật đồng thời nhiều \mathbf{w}_u vì các hệ tuyến tính độc lập.
- Phù hợp cho dữ liệu lớn khi triển khai phân tán (MapReduce, Spark).
- Nhược điểm: mỗi bước cần giải hệ kích thước r cho mọi user/item; nếu r lớn hoặc mật độ tương tác cao, chi phí có thể lớn.

2.5.5 Cập nhật theo gradient (SGD)

Một cách học khác là tối ưu trực tiếp $J(W, H)$ bằng hạ gradient ngẫu nhiên (stochastic gradient descent — SGD). Ta tối ưu theo từng quan sát $(u, i) \in \Omega$ bằng cách cập nhật các tham số theo bước âm gradient.

Mô hình không bias Đặt sai số dự đoán tại cặp (u, i) là

$$e_{ui} = r_{ui} - \mathbf{w}_u^\top \mathbf{h}_i.$$

Đạo hàm riêng của J theo \mathbf{w}_u và \mathbf{h}_i là

$$\frac{\partial J}{\partial \mathbf{w}_u} = -2 \sum_{i \in \Omega_u} e_{ui} \mathbf{h}_i + 2\lambda \mathbf{w}_u, \quad \frac{\partial J}{\partial \mathbf{h}_i} = -2 \sum_{u \in \Omega^i} e_{ui} \mathbf{w}_u + 2\lambda \mathbf{h}_i.$$

Cập nhật SGD cho một bước trên cặp (u, i) với tốc độ học $\eta > 0$ là:

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \eta(e_{ui} \mathbf{h}_i - \lambda \mathbf{w}_u),$$

$$\mathbf{h}_i \leftarrow \mathbf{h}_i + \eta(e_{ui} \mathbf{w}_u - \lambda \mathbf{h}_i).$$

Lặp qua tất cả $(u, i) \in \Omega$ (có thể theo ngẫu nhiên) và nhiều epoch cho tới hội tụ.

Mô hình có bias Khi bao gồm bias μ, b_u, b_i , đặt

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{w}_u^\top \mathbf{h}_i, \quad e_{ui} = r_{ui} - \hat{r}_{ui}.$$

Cập nhật SGD (với cùng điều chuẩn) là:

$$b_u \leftarrow b_u + \eta(e_{ui} - \lambda b_u), \quad b_i \leftarrow b_i + \eta(e_{ui} - \lambda b_i),$$

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \eta(e_{ui} \mathbf{h}_i - \lambda \mathbf{w}_u), \quad \mathbf{h}_i \leftarrow \mathbf{h}_i + \eta(e_{ui} \mathbf{w}_u - \lambda \mathbf{h}_i).$$

Lưu ý thực tiễn với SGD

- Khởi tạo: các véc-tơ $\mathbf{w}_u, \mathbf{h}_i$ thường khởi tạo ngẫu nhiên nhỏ (ví dụ phân phối chuẩn với phương sai nhỏ) hoặc bằng phân phối đều.
- Tốc độ học η : có thể cố định hoặc giảm dần theo epoch; giá trị quá lớn gây dao động, quá nhỏ gây chậm hội tụ.
- Shuffle dữ liệu: nên xáo trộn thứ tự cặp (u, i) giữa các epoch để tăng hiệu quả SGD.
- Early stopping: dừng khi lỗi trên tập kiểm định không cải thiện.
- Điều chỉnh λ : cần tune qua validation.

2.5.6 Độ phức tạp tính toán

Với SGD, mỗi bước cập nhật cho cặp (u, i) có chi phí $\mathcal{O}(r)$ (tích vô hướng + cập nhật hai véc-tơ). Với $|\Omega|$ phần tử quan sát, mỗi epoch chi phí $\mathcal{O}(r|\Omega|)$. Với ALS, chi phí cho mỗi người dùng là giải hệ $r \times r$ (thường $\mathcal{O}(r^3)$ nếu dùng nghịch đảo, nhưng thực tế dùng Cholesky có chi phí $\mathcal{O}(r^3)$ cho mỗi u hay i ; tuy nhiên vì r nhỏ, và có thể song song hóa, ALS thường hiệu quả trên dữ liệu lớn).

2.5.7 Tổng kết và thực hành

Tóm tắt: phương pháp phân tích ma trận cung cấp khung hiệu quả cho mô hình hoá cấu trúc ẩn trong dữ liệu tương tác, với hai lối huấn luyện chính: ALS (nghiệm đóng cho từng khối — phù hợp cho triển khai song song/thuận tiện cho dữ liệu lớn) và SGD (cập nhật mẫu-tiếp-mẫu — đơn giản, dễ triển khai, linh hoạt với nhiều dạng hàm mất mát). Trong thực tế, ta nên:

1. Chuẩn hoá dữ liệu (loại bỏ bias hoặc học bias riêng).
2. Lựa chọn r, λ, η bằng cross-validation.
3. Sử dụng early stopping và đánh giá bằng các chỉ số thích hợp (RMSE cho rating prediction, Precision@K/Recall@K/NDCG cho đề xuất Top-K).
4. Khi dữ liệu là phản hồi ngầm, cân nhắc dùng WALS hoặc các tiêu chí xếp hạng (BPR) thay vì tối ưu RMSE thuần túy.

2.6 Ứng dụng thực tiễn

Phương pháp lọc cộng tác đã được triển khai rộng rãi trong nhiều lĩnh vực. Dưới đây là một số ví dụ điển hình.

2.6.1 Amazon: Gợi ý sản phẩm thương mại điện tử

Amazon là một trong những công ty đầu tiên áp dụng Item-based Collaborative Filtering ở quy mô lớn. Mỗi khi người dùng xem một sản phẩm, hệ thống sẽ hiển thị phần “Customers who bought this item also bought”.

Ví dụ, nếu nhiều người dùng mua cùng lúc một chiếc máy ảnh và một thẻ nhớ, thì khi một khách hàng mới mua máy ảnh, hệ thống sẽ gợi ý thêm thẻ nhớ. Đây chính là ứng dụng thực tế của Item-based CF, nơi sự tương đồng giữa các sản phẩm được tính toán từ dữ liệu mua hàng của cộng đồng.

Ưu điểm của cách tiếp cận này là tính ổn định: quan hệ giữa các sản phẩm ít thay đổi hơn so với hành vi người dùng. Do đó, mô hình có thể được tính toán trước và tái sử dụng trong thời gian dài, tiết kiệm chi phí.

2.6.2 Netflix: Gợi ý phim cá nhân hóa

Netflix sử dụng CF để dự đoán phim mà người dùng sẽ thích. Thay vì chỉ dựa trên thông tin về thể loại hoặc diễn viên (content-based), Netflix tập trung vào hành vi cộng đồng: nếu người dùng *A* và *B* cùng thích một tập phim nào đó, họ có khả năng cùng thích những bộ phim khác.

Đặc biệt, Netflix không chỉ sử dụng User-based hay Item-based cơ bản mà còn triển khai Matrix Factorization, giúp tìm ra các yếu tố ẩn (latent factors). Ví dụ, một yếu tố ẩn có thể biểu thị sự ưa thích phim hành động, một yếu tố khác thể hiện sự quan tâm tới phim lãng mạn. Khi đó, hồ sơ người dùng và đặc trưng phim đều được biểu diễn trong cùng một không gian ẩn, cho phép gợi ý chính xác và cá nhân hóa hơn.

2.6.3 YouTube: Gợi ý video

YouTube kết hợp nhiều phương pháp, trong đó CF đóng vai trò nền tảng. Khi một người dùng xem và thích một video, hệ thống sẽ tìm các video tương tự được cộng đồng có hành vi gần giống cũng xem. Điều này giải thích vì sao sau khi xem một video âm nhạc, bạn thường nhận được gợi ý các bài hát cùng thể loại hoặc của cùng ca sĩ.

YouTube còn kết hợp thêm tín hiệu từ thời gian xem (watch time), lượt thích (like), chia sẻ, và cả vị trí địa lý, nhưng bản chất của việc phát hiện mối liên hệ giữa video vẫn dựa nhiều trên CF.

2.6.4 Mạng xã hội: Facebook, TikTok

Trên Facebook, hệ thống gợi ý bạn bè (People You May Know) sử dụng CF: nếu *A* và *B* cùng có nhiều bạn chung, xác suất cao là họ sẽ quen nhau. Tương tự, TikTok đề xuất video dựa trên sở thích cộng đồng, nơi thuật toán CF được triển khai trên quy mô hàng tỷ người dùng.

2.6.5 Thương mại điện tử Việt Nam

Các sàn thương mại điện tử như Shopee, Tiki, Lazada cũng ứng dụng CF để gợi ý sản phẩm phù hợp với hành vi mua sắm tại thị trường Việt Nam. Ví dụ, khi người dùng thêm vào giỏ hàng một chiếc laptop, hệ thống có thể gợi ý chuột, bàn phím, hoặc túi chống sốc – những sản phẩm mà cộng đồng thường mua kèm.

2.7 Hạn chế và thách thức

Mặc dù hiệu quả, CF vẫn tồn tại nhiều vấn đề cần giải quyết.

2.7.1 Cold-start

Khi xuất hiện một người dùng mới hoặc sản phẩm mới chưa có đánh giá, CF khó đưa ra gợi ý chính xác. Đây là nhược điểm cơ bản vì CF phụ thuộc vào dữ liệu quá khứ. Các cách khắc phục thường là:

- Kết hợp thông tin nội dung (content-based).
- Thu thập dữ liệu ban đầu qua khảo sát hoặc tương tác nhỏ.

2.7.2 Dữ liệu thưa

Trong các hệ thống lớn, mỗi người dùng chỉ đánh giá một số rất ít sản phẩm. Ví dụ, trên Netflix, một người dùng có thể chỉ xem vài chục bộ phim trong khi toàn bộ hệ thống có hàng chục nghìn phim. Khi đó, ma trận *R* rất thưa thớt, làm giảm độ chính xác khi tính tương đồng.

2.7.3 Khả năng mở rộng

Khi số người dùng và sản phẩm lên tới hàng triệu, việc tính toán toàn bộ ma trận tương đồng trở nên tốn kém. Giải pháp thường là:

- Dùng kỹ thuật giảm chiều (matrix factorization).
- Dùng thuật toán phân tán trên hệ thống Big Data như Hadoop, Spark.

2.7.4 Thiếu khả năng giải thích

CF thường đưa ra dự đoán mà không giải thích rõ ràng lý do. Người dùng có thể thắc mắc tại sao họ lại được gợi ý sản phẩm này. Điều này đặc biệt quan trọng trong các hệ thống cần minh bạch (ví dụ: gợi ý tin tức, y tế).

2.7.5 Đa dạng và độ mới

CF có xu hướng tập trung vào những sản phẩm phổ biến (popular items) vì chúng có nhiều dữ liệu. Điều này dẫn tới mất tính đa dạng, hạn chế khả năng khám phá sản phẩm mới. Do đó, cần có cơ chế cân bằng giữa *khả năng dự đoán* và *khả năng khám phá*.

Chương 3

GỢI Ý DỰA TRÊN LỌC CỘNG TÁC NÂNG CAO

3.1	Lọc cộng tác dựa trên học máy cơ bản	42
3.2	Lọc cộng tác nâng cao: Học biểu diễn ẩn	44
3.3	Dữ liệu Explicit và Implicit Feedback	57
3.4	Sử dụng đồng thời Implicit và Explicit Feedback	65

3.1 Lọc cộng tác dựa trên học máy cơ bản

Lọc cộng tác (Collaborative Filtering – CF) là một trong những phương pháp phổ biến nhất trong hệ thống gợi ý, dựa trên giả định rằng những người dùng có hành vi hoặc sở thích tương đồng trong quá khứ sẽ có xu hướng có sở thích tương tự trong tương lai. Trong số các kỹ thuật lọc cộng tác, **lọc cộng tác dựa trên học máy cơ bản** (Machine Learning-based CF) sử dụng các mô hình học máy để dự đoán đánh giá hoặc mức độ thích của người dùng đối với các sản phẩm, bộ phim, bài hát, hay bất kỳ item nào trong hệ thống.

3.1.1 Ý tưởng chính

Ý tưởng cốt lõi của mô hình học máy cho lọc cộng tác là **biểu diễn người dùng và item dưới dạng vector**, sao cho máy tính có thể xử lý và học các mẫu ẩn trong dữ liệu. Cụ thể:

- Mỗi **người dùng** được ánh xạ thành một vector $\mathbf{u} = [u_1, u_2, \dots, u_n]$, trong đó n là số lượng item trong hệ thống.
- Mỗi **chiều của vector** tương ứng với một item, và giá trị tại chiều đó thể hiện mức độ thích hoặc đánh giá của user đối với item đó.

Trong bối cảnh này, mỗi **item** có thể được coi là một **thuộc tính nhãn** (label) riêng biệt. Việc dự đoán mức độ thích của người dùng đối với một item trở thành bài toán **phân loại hoặc hồi quy**, tùy thuộc vào dữ liệu đầu ra:

- Nếu dữ liệu là nhãn rời rạc (ví dụ: thích/không thích), bài toán trở thành **phân loại nhị phân**.
- Nếu dữ liệu là đánh giá số (ví dụ: 1–5 sao), bài toán trở thành **bài toán hồi quy**.

Một cách trực tiếp để triển khai mô hình này là xây dựng **một bộ phân loại hoặc mô hình hồi quy riêng cho từng item**. Với m item, hệ thống sẽ có m mô hình riêng biệt, mỗi mô hình học mối quan hệ giữa các vector người dùng và nhãn tương ứng của item đó.

3.1.2 Quy trình học máy cơ bản

Quy trình xây dựng hệ thống lọc cộng tác dựa trên học máy cơ bản có thể được tóm tắt như sau:

1. **Tiền xử lý dữ liệu:** Chuyển dữ liệu đánh giá của user thành ma trận người dùng – item, trong đó mỗi dòng là vector người dùng và mỗi cột là một item.
2. **Xây dựng mô hình:** Với mỗi item, huấn luyện một mô hình học máy để dự đoán nhãn (hoặc giá trị đánh giá) dựa trên vector người dùng. Các mô hình phổ biến bao gồm logistic regression, decision tree, SVM, hoặc các mô hình hồi quy tuyến tính.
3. **Dự đoán:** Khi cần gợi ý item cho một user mới, đưa vector người dùng vào các mô hình tương ứng để dự đoán mức độ thích của họ đối với từng item.
4. **Xếp hạng và gợi ý:** Sắp xếp các item dựa trên dự đoán và gợi ý những item có điểm số cao nhất.

3.1.3 Ưu điểm và hạn chế

Ưu điểm:

- Đơn giản, dễ hiểu và dễ triển khai.
- Mỗi mô hình item riêng biệt có thể được tối ưu hóa độc lập, giúp linh hoạt trong việc cập nhật dữ liệu mới.

Hạn chế:

- Không xử lý tốt khi ma trận người dùng – item **rất thưa** (sparse).
- Khó mở rộng cho hệ thống có hàng triệu item do phải huấn luyện quá nhiều mô hình.
- Không khai thác mối quan hệ tiềm ẩn giữa các item (khác với các phương pháp dựa trên ma trận phân rã hay embedding).

3.1.4 Kết luận

Mô hình lọc cộng tác dựa trên học máy cơ bản cung cấp một cách tiếp cận trực quan để dự đoán sở thích của người dùng thông qua việc biểu diễn dữ liệu dạng vector và áp dụng các mô hình phân loại/hồi quy. Mặc dù có giới hạn về khả năng mở rộng và độ thưa của dữ liệu, đây vẫn là một nền tảng quan trọng để phát triển các phương pháp gợi ý phức tạp hơn, bao gồm cả mô hình embedding và các kỹ thuật học sâu.

3.2 Lọc cộng tác nâng cao: Học biểu diễn ẩn

Trong các hệ thống gợi ý hiện đại, việc học các biểu diễn ẩn (latent representation) cho người dùng và item là một kỹ thuật rất quan trọng. Thay vì làm việc trực tiếp với ma trận người dùng – item thưa thớt, các phương pháp nâng cao học các vector biểu diễn ẩn để mô hình có thể dự đoán sở thích của người dùng một cách hiệu quả hơn.

3.2.1 Phân tích ma trận (Matrix Factorization)

Phân tích ma trận [15] là một trong những phương pháp học biểu diễn ẩn phổ biến nhất. Ý tưởng chính là phân rã ma trận đánh giá $R \in \mathbb{R}^{m \times n}$ thành tích của hai ma trận mật độ thấp:

$$R \approx UV^\top \quad (3.1)$$

Trong đó:

- $U \in \mathbb{R}^{m \times k}$ là ma trận biểu diễn ẩn của m người dùng.
- $V \in \mathbb{R}^{n \times k}$ là ma trận biểu diễn ẩn của n item.
- $k \ll m, n$ là số chiều của không gian ẩn.

Việc học U và V có thể được thực hiện thông qua tối ưu hóa hàm lỗi:

$$\min_{U,V} \sum_{(i,j) \in \mathcal{K}} (R_{ij} - U_i^\top V_j)^2 + \lambda(\|U_i\|^2 + \|V_j\|^2) \quad (3.2)$$

Trong đó \mathcal{K} là tập các cặp (i, j) có đánh giá thực sự và λ là tham số điều chỉnh.

3.2.2 Học sâu trong phân tích ma trận

Mối liên hệ với mạng nơ-ron: Phân tích ma trận có thể được xem như một **mạng nơ-ron nhân tạo đơn giản** với một lớp ẩn. Cụ thể, vector người dùng U_i và vector item V_j được nhân để dự đoán rating, tương đương với một mạng nơ-ron có *dot product* ở lớp đầu ra.

Một số phương pháp nâng cao kết hợp học sâu để học biểu diễn ẩn hiệu quả hơn, có khả năng khai thác các tương tác phi tuyến giữa người dùng và item:

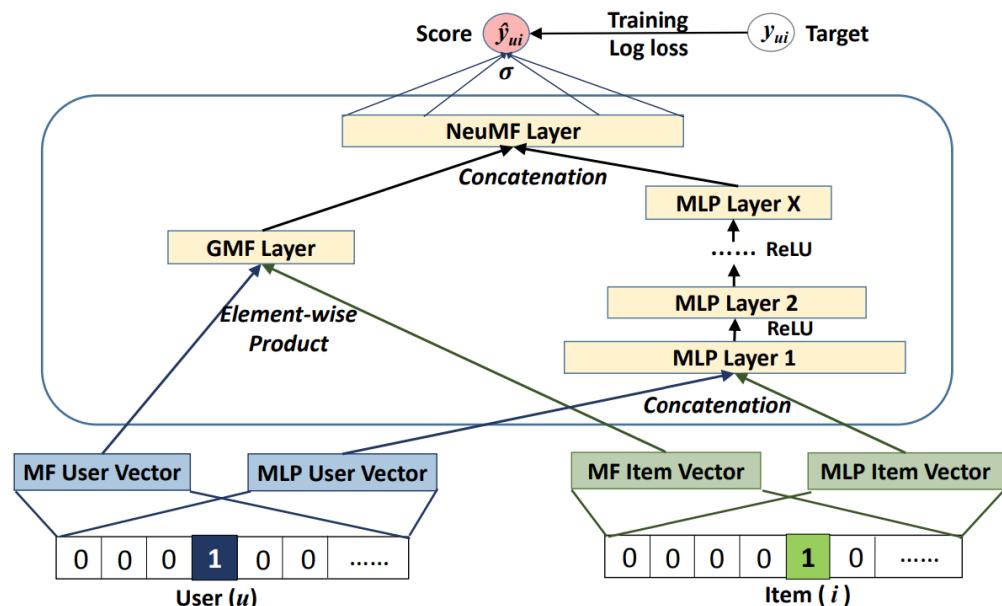
- **Neural Matrix Factorization (NeuMF) [9]:** Kết hợp matrix factorization truyền thống với mạng nơ-ron đa tầng để học tương tác phi tuyến giữa người dùng và item.
- **Deep Matrix Factorization [32]:** Sử dụng các lớp ẩn sâu để học biểu diễn ẩn phức tạp cho cả người dùng và item.
- **LightGCN [8]:** Sử dụng đồ thị người dùng – item và Graph Convolutional Network (GCN) nhẹ để học biểu diễn ẩn, giúp cải thiện hiệu quả gợi ý trong các hệ thống lớn.

Việc học biểu diễn ẩn cho người dùng và item là bước tiến quan trọng so với các mô hình lọc cộng tác cơ bản. Các kỹ thuật phân tích ma trận và các phương pháp học sâu không chỉ xử lý ma trận thưa tốt hơn mà còn có khả năng khai thác các tương tác phi tuyến, từ đó nâng cao độ chính xác của hệ thống gợi ý.

3.2.3 Neural Matrix Factorization (NeuMF)

Neural Matrix Factorization (NeuMF) [9] là một phương pháp gợi ý dựa trên học sâu, kết hợp sức mạnh của **Matrix Factorization (MF) truyền thống** và **mạng nơ-ron đa tầng (MLP)** để học các biểu diễn ẩn (latent representation) phức tạp cho người dùng và item. NeuMF cho phép khai thác các tương tác phi tuyến giữa người dùng và item, nâng cao khả năng dự đoán so với MF thuần túy.

Kiến trúc tổng quan



Hình 3.1: Minh họa kiến trúc của NeuMF.

NeuMF kết hợp hai thành phần chính (Hình 3.1):

1. **Generalized Matrix Factorization (GMF):** phần mở rộng của MF truyền thống, dự đoán rating bằng tích vô hướng (dot product) giữa vector người dùng và item.
2. **Multi-Layer Perceptron (MLP):** một mạng nơ-ron đa tầng, học các tương tác phi tuyến giữa người dùng và item.

Các vector biểu diễn ẩn của user và item được học riêng cho GMF và MLP. Sau đó, các biểu diễn này được kết hợp ở tầng đầu ra để đưa ra dự đoán cuối cùng.

Biểu diễn toán học

1. Generalized Matrix Factorization (GMF) Giả sử $U_i \in \mathbb{R}^{k_1}$ và $V_j \in \mathbb{R}^{k_1}$ lần lượt là vector biểu diễn ẩn của người dùng i và item j trong GMF. Dự đoán rating phi tuyến tính được tính bằng:

$$\hat{y}_{ij}^{GMF} = f(U_i^\top V_j) \quad (3.3)$$

Trong đó $f(\cdot)$ thường là hàm sigmoid nếu dữ liệu nhãn là nhị phân, hoặc identity nếu rating là liên tục.

2. Multi-Layer Perceptron (MLP) Vector người dùng $U'_i \in \mathbb{R}^{k_2}$ và item $V'_j \in \mathbb{R}^{k_2}$ được kết hợp qua các lớp nơ-ron để học các tương tác phi tuyến:

$$\mathbf{h}_1 = \sigma(W_1[U'_i; V'_j] + b_1) \quad (3.4)$$

$$\mathbf{h}_l = \sigma(W_l \mathbf{h}_{l-1} + b_l), \quad l = 2, \dots, L \quad (3.5)$$

Trong đó:

- $[U'_i; V'_j]$ là vector ghép của user và item.
- σ là hàm kích hoạt phi tuyến, ví dụ ReLU.
- W_l, b_l là trọng số và bias của lớp thứ l .

3. Kết hợp GMF và MLP Tầng đầu ra của NeuMF kết hợp GMF và MLP:

$$\hat{y}_{ij} = \sigma(\mathbf{h}_{out}^\top [U_i \odot V_j; \mathbf{h}_L]) \quad (3.6)$$

Trong đó:

- \odot là tích Hadamard (tích từng phần) giữa vector GMF.
- $[\cdot; \cdot]$ là ghép vector GMF và output của MLP.
- \mathbf{h}_{out} là vector trọng số cuối cùng của tầng đầu ra. \mathbf{h}_{out} có thể được xây dựng bằng một mạng nơ-ron sâu MLP để tăng khả năng học biểu diễn của mạng.

Huấn luyện

NeuMF được huấn luyện để tối thiểu hóa hàm mất mát, thường là **binary cross-entropy** cho dữ liệu nhị phân hoặc **MSE** cho dữ liệu rating liên tục:

$$\mathcal{L} = \sum_{(i,j) \in \mathcal{K}} \ell(\hat{y}_{ij}, y_{ij}) + \lambda(\|U_i\|^2 + \|V_j\|^2 + \text{trọng số MLP}) \quad (3.7)$$

Trong đó:

- \mathcal{K} là tập các cặp user-item có đánh giá.
- $\ell(\cdot)$ là hàm mất mát thích hợp.
- λ là hệ số điều chỉnh.

Quá trình huấn luyện thường sử dụng **stochastic gradient descent (SGD)** hoặc các biến thể như **Adam**.

Ưu điểm

- Học được cả các tương tác tuyến tính (GMF) và phi tuyến (MLP).
- Nâng cao độ chính xác so với MF truyền thống.
- Linh hoạt trong việc áp dụng cho dữ liệu nhị phân hoặc rating liên tục.

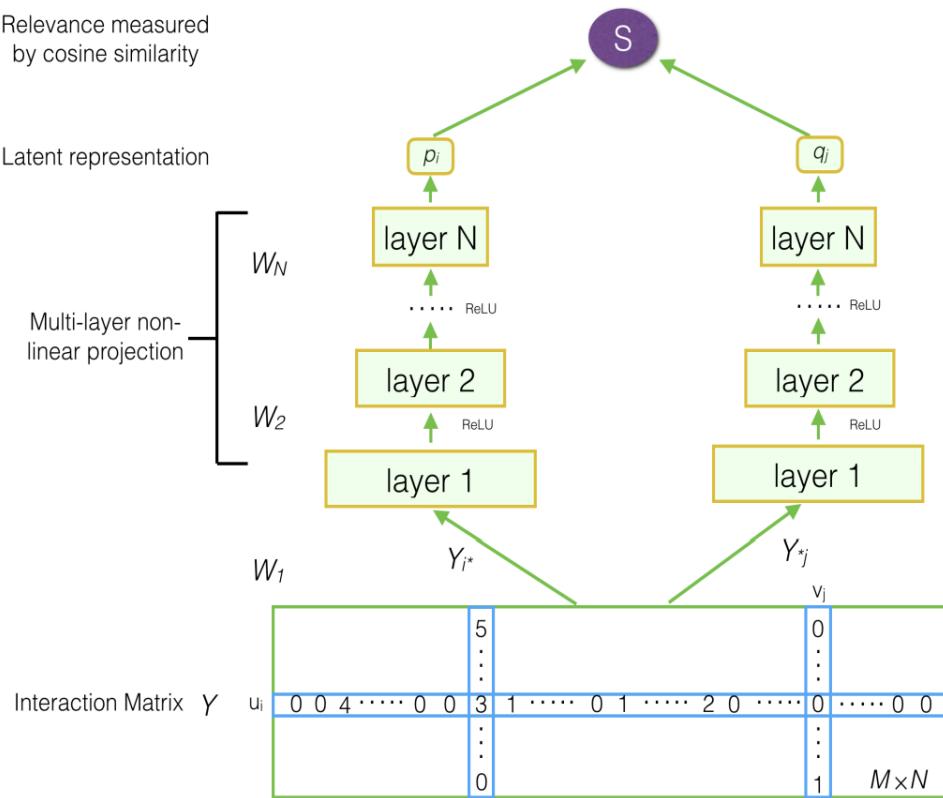
Hạn chế

- Số lượng tham số lớn, yêu cầu dữ liệu huấn luyện nhiều.
- Huấn luyện phức tạp hơn so với MF đơn giản.

3.2.4 Deep Matrix Factorization (DMF)

Deep Matrix Factorization (Deep MF) [32] là một phương pháp gợi ý dựa trên học sâu, mở rộng tư tưởng của Matrix Factorization bằng cách học các biểu diễn ẩn (latent representations) phi tuyến cho người dùng và item từ *toàn bộ* hàng/cột của ma trận tương tác. Thay vì khởi tạo trực tiếp các embedding cho user/item, DMF dùng hai mạng riêng (“user-tower” và “item-tower”) để chiết xuất biểu diễn sâu từ hàng (row) tương tác của user và cột (column) tương tác của item, sau đó đo độ tương đồng trong không gian ẩn để dự đoán tương tác.

Kiến trúc tổng quan



Hình 3.2: Minh họa kiến trúc DMF: hai towers (user/item) học biểu diễn sâu từ hàng/cột ma trận tương tác, score được tính bằng cosine similarity giữa hai biểu diễn cuối cùng.

Kiến trúc DMF (Hình 3.2) gồm hai thành phần chính:

1. **User tower:** một mạng feed-forward nhiều lớp nhận hàng Y_{i*} của ma trận tương tác làm input và sinh vector ẩn p_i .
2. **Item tower:** một mạng feed-forward nhiều lớp nhận cột Y_{*j} của ma trận tương tác làm input và sinh vector ẩn q_j .

Hai towers có thể có cấu trúc tương tự nhưng có tham số riêng. Mức độ tương tác giữa user i và item j được tính bằng hàm similarity (paper gốc sử dụng cosine similarity) trên hai vector ẩn này.

Tiền xử lý đầu vào (Ma trận tương tác)

Ma trận đầu vào Y có thể được xây dựng theo nhiều cách tùy dữ liệu:

- **Implicit binarized:** $Y_{ij} = 1$ nếu tương tác được quan sát, ngược lại 0.
- **Explicit-aware:** $Y_{ij} = R_{ij}$ nếu rating R_{ij} có sẵn, và 0 nếu chưa quan sát (có thể chuẩn hoá rating về khoảng $[0, 1]$).

DMF thường dùng ma trận kết hợp (observed ratings giữ giá trị, ô chưa quan sát là 0) để tận dụng đồng thời thông tin explicit và implicit.

Biểu diễn toán học

1. Forward pass của từng tower Gọi $x_i^{(u)} = Y_{i*} \in \mathbb{R}^N$ là hàng tương tác của user i (với N items), và $x_j^{(v)} = Y_{*j} \in \mathbb{R}^M$ là cột tương tác của item j (với M users). Mỗi tower là một mạng nhiều lớp:

$$\begin{aligned} p_i &= \text{UserTower}(x_i^{(u)}; \Theta_U), \\ q_j &= \text{ItemTower}(x_j^{(v)}; \Theta_I). \end{aligned}$$

Cụ thể, với L lớp và hàm kích hoạt $f(\cdot)$ (ví dụ ReLU), các bước nội bộ có dạng:

$$\begin{aligned} h^{(1)} &= f(W^{(1)}x + b^{(1)}), \\ h^{(t)} &= f(W^{(t)}h^{(t-1)} + b^{(t)}), \quad t = 2, \dots, L, \end{aligned}$$

và vector ẩn cuối cùng $h^{(L)}$ chính là p_i hoặc q_j tương ứng.

2. Hàm score DMF sử dụng cosine similarity làm hàm score dự đoán:

$$\hat{y}_{ij} = \text{cosine}(p_i, q_j) = \frac{p_i^\top q_j}{\|p_i\| \|q_j\|}.$$

Cosine tập trung vào hướng của các biểu diễn ẩn, giảm phụ thuộc vào chuẩn của vector, phù hợp khi các biểu diễn được học qua nhiều lớp phi tuyến.

Hàm mất mát (Loss)

Để kết hợp thông tin explicit và implicit, DMF đề xuất sử dụng một dạng *normalized cross-entropy* (NCE) theo chế độ pointwise, kèm negative sampling từ

các ô chưa quan sát. Sau khi chuẩn hoá giá trị rating $Y_{ij} \in [0, 1]$ (chia cho giá trị rating tối đa nếu cần), loss tổng quát có thể viết:

$$\mathcal{L} = - \sum_{(i,j) \in T} [y_{ij} \log(\hat{y}'_{ij}) + (1 - y_{ij}) \log(1 - \hat{y}'_{ij})] + \lambda \Omega(\Theta),$$

trong đó:

- T là tập huấn luyện bao gồm các cặp dương (observed) và các sample âm (negative sampling) từ các ô chưa quan sát.
- \hat{y}'_{ij} là giá trị dự đoán đã được map/clip vào khoảng $(0, 1)$ (vì cosine có thể trả về giá trị ngoài khoảng này), thường bằng một phép affine + sigmoid hoặc bằng clip + affine để đảm bảo ổn định cho log.
- $\Omega(\Theta)$ là thành phần regularization (ví dụ L_2) trên tham số mô hình và λ là hệ số điều chỉnh.

Cách tiếp cận này cho phép tác động lớn hơn của các ô có rating cao (khi dùng giá trị rating đã chuẩn hoá) đồng thời học từ các tương tác implicit thông qua negative sampling.

Huấn luyện

- Mô hình được huấn luyện end-to-end bằng **backpropagation** và tối ưu bằng các thuật toán tối ưu bậc thấp như **Adam** hoặc SGD với momentum.
- Do ma trận Y rất thưa, thực tế huấn luyện sử dụng **mini-batch** và **negative sampling**: với mỗi positive (quan sát), lấy một số negative ngẫu nhiên từ các ô 0 để cân bằng dữ liệu.
- Regularization (weight decay, dropout) và early-stopping thường được sử dụng để tránh overfitting. Tầng đầu tiên của mỗi tower thường là một projection tuyến tính giảm chiều để xử lý input chiều cao (sparse \rightarrow dense).
- Các siêu tham số cần điều chỉnh: số lớp L , kích thước latent, tỉ lệ negative sampling, learning rate, hệ số regularization.

Ưu điểm

- Học được biểu diễn phi tuyến (non-linear) cho user và item dựa trên *tất cả* mô hình tương tác (hàng/cột), giúp nắm bắt hồ sơ global tốt hơn so với MF tuyến tính.

- Kết hợp linh hoạt thông tin explicit và implicit thông qua thiết kế đầu vào và loss.
- Mô hình end-to-end: biểu diễn và hàm score được tối ưu cùng nhau.

Hạn chế

- **Chi phí tính toán và bộ nhớ:** input dimension bằng số item (hoặc số user) gây vấn đề với hệ thống lớn, đòi hỏi biểu diễn sparse và tầng projection hiệu quả.
- **Cold-start:** user hoặc item mới (hàng/cột toàn zeros) cung cấp rất ít thông tin cho tower tương ứng; cần kết hợp features content-based hoặc onboarding.
- **Tuning phức tạp:** nhiều siêu tham số (số lớp, kích thước, negative ratio) và cần thử nghiệm để đạt hiệu suất tốt.

3.2.5 LightGCN

LightGCN [8] là một mô hình gợi ý dựa trên đồ thị (graph-based), được thiết kế để tận dụng tối đa cơ chế lan truyền thông tin (message passing / neighborhood aggregation) của Graph Convolutional Networks (GCNs) nhưng đồng thời loại bỏ các thành phần không cần thiết cho bài toán Collaborative Filtering — cụ thể là *linear transformation* và các hàm kích hoạt (non-linear activation). Kết quả là một kiến trúc nhẹ, trực quan và hiệu quả cả về mặt hiệu năng lẫn chi phí tính toán.

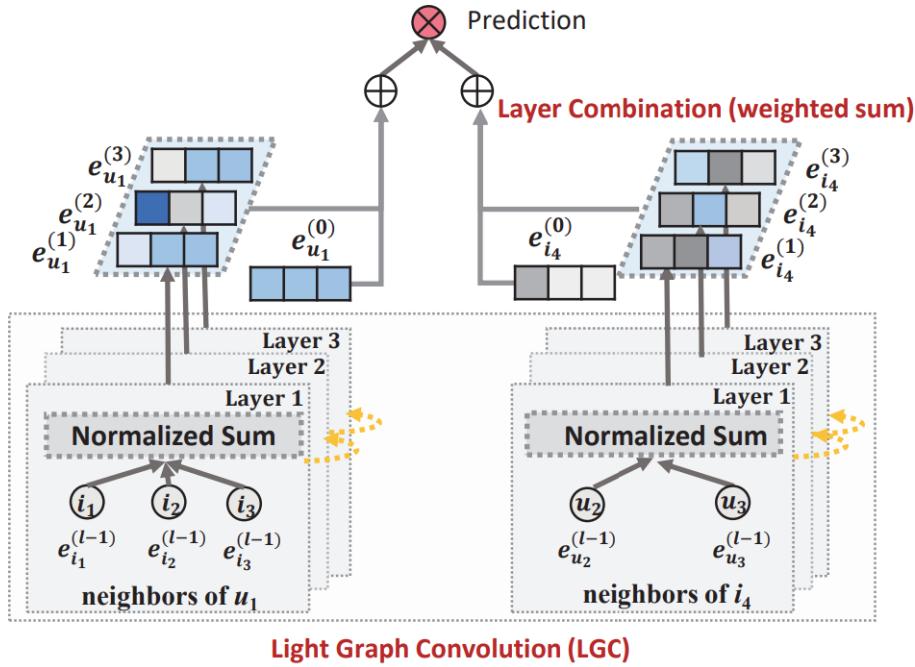
Kiến trúc tổng quan

Những điểm chính:

- Dữ liệu được xem như đồ thị hai phía (bipartite graph) $G = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$, với \mathcal{U} là tập user, \mathcal{I} là tập item, và cạnh $(u, i) \in \mathcal{E}$ biểu thị một tương tác (implicit/explicit).
- Khởi tạo embedding ban đầu cho từng node (user hoặc item): $e_v^{(0)} \in \mathbb{R}^d$.
- Ở mỗi layer chỉ thực hiện *neighborhood aggregation* (không có affine transform hay activation):

$$e_v^{(k+1)} = \sum_{w \in \mathcal{N}(v)} \frac{1}{\sqrt{|\mathcal{N}(v)|} \sqrt{|\mathcal{N}(w)|}} e_w^{(k)},$$

tức sử dụng normalized adjacency symmetric (degree normalization).



Hình 3.3: Sơ đồ tổng quan LightGCN: embedding ban đầu được lan truyền qua K lớp aggregation; embedding cuối cùng là tổ hợp tuyến tính (thường là trung bình) các embedding qua các lớp.

- Embedding cuối cùng dùng để tính score là tổ hợp các embedding qua K tầng:

$$e_v = \sum_{k=0}^K \alpha_k e_v^{(k)},$$

thường chọn $\alpha_k = \frac{1}{K+1}$ (trung bình đều).

- Score giữa user u và item i :

$$\hat{y}_{ui} = e_u^\top e_i.$$

Biểu diễn toán học chi tiết

1. Ma trận kè và chuẩn hóa Gọi $R \in \mathbb{R}^{M \times N}$ là ma trận tương tác (binary hoặc weighted). Ta xây dựng ma trận kè đối xứng của đồ thị bipartite:

$$A = \begin{bmatrix} 0 & R \\ R^\top & 0 \end{bmatrix} \in \mathbb{R}^{(M+N) \times (M+N)}.$$

Dộ lớn hàng (degree) của node v là $d_v = \sum_w A_{vw}$. Định nghĩa ma trận chuẩn hoá đối xứng:

$$\tilde{A} = D^{-1/2} A D^{-1/2}, \quad D = \text{diag}(d_1, \dots, d_{M+N}).$$

Khi đó, propagation ở cấp ma trận:

$$E^{(k+1)} = \tilde{A} E^{(k)},$$

với $E^{(0)} \in \mathbb{R}^{(M+N) \times d}$ là ma trận embedding ban đầu (user+item).

2. Propagation per-node (scalar form) Tương đương với biểu thức cho node v :

$$e_v^{(k+1)} = \sum_{w \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_w}} e_w^{(k)}.$$

3. Layer combination Sau K bước propagation, embedding cuối cùng là:

$$e_v = \sum_{k=0}^K \alpha_k e_v^{(k)}.$$

Paper gốc sử dụng $\alpha_k = 1/(K+1)$. Trong các biến thể có thể học các α_k (trainable) hoặc dùng trọng số khác nhau.

4. Hàm score

$$\hat{y}_{ui} = e_u^\top e_i.$$

(ưu tiên inner product vì tính toán nhanh và tương thích với BPR).

Hàm mất mát (Loss)

LightGCN tối ưu theo mục tiêu xếp hạng pairwise — **BPR loss** [20]:

$$\mathcal{L}_{\text{BPR}} = - \sum_{(u,i,j) \in \mathcal{S}} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \sum_{v \in \mathcal{U} \cup \mathcal{I}} \|e_v^{(0)}\|_2^2,$$

trong đó \mathcal{S} là tập các triple (u, i, j) với i là item dương (observed) và j là item negative (không quan sát). Lưu ý regularization chỉ áp dụng lên embedding ban đầu $e^{(0)}$ (theo nguyên tắc rằng các tham số duy nhất để học là embedding ban đầu).

Thuật toán huấn luyện (pseudo-code)

Thuật toán 3.1 Huấn luyện LightGCN (một epoch, minh họa)

Require: interaction matrix R , embedding dim d , layers K , batch size B , learning rate η

- 1: Khởi tạo $E^{(0)}$ (user+item embeddings)
 - 2: Tính ma trận $\tilde{A} = D^{-1/2}AD^{-1/2}$ dưới dạng sparse
 - 3: **for** each mini-batch of positive pairs (u, i) sampled **do**
 - 4: Lấy negative sample j cho mỗi (u, i)
 - 5: {Propagation: compute layer-wise embeddings via sparse matmul}
 - 6: $E^{(0)} \leftarrow$ embeddings initial
 - 7: **for** $k = 0$ to $K - 1$ **do**
 - 8: $E^{(k+1)} \leftarrow \tilde{A}E^{(k)}$
 - 9: **end for**
 - 10: Tính $E^{(\text{final})} = \sum_{k=0}^K \alpha_k E^{(k)}$
 - 11: Tính score $\hat{y}_{ui}, \hat{y}_{uj}$ và loss batch theo BPR
 - 12: Cập nhật $E^{(0)}$ bằng gradient descent (Adam/SGD); (lưu ý: gradients lan truyền qua các bước propagation)
 - 13: **end for**
-

Ghi chú quan trọng triển khai

- **Tính toán sparse:** propagation sử dụng nhân ma trận thừa $\tilde{A}E^{(k)}$ — cần lưu \tilde{A} ở dạng sparse CSR/COO để tiết kiệm bộ nhớ và tăng tốc.
- **Cache per-layer embeddings:** trong mỗi epoch có thể tính toàn bộ $E^{(k)}$ cho mọi node (chi phí $O(K|\mathcal{E}|d)$) rồi lấy tổ hợp; làm như vậy giúp ta tránh gọi propagation nhiều lần cho từng mini-batch.
- **Chỉ cập nhật $E^{(0)}$:** các tham số duy nhất cần cập nhật là embedding ban đầu; propagation là phép toán tuyến tính cố định phụ thuộc vào \tilde{A} .
- **Negative sampling hiệu quả:** sampling tỷ lệ negatives hợp lý (1–5 negatives mỗi positive) và thực hiện theo mini-batch vectorized.

Phân tích độ phức tạp và bộ nhớ

- **Thời gian (per epoch):** nếu tính toàn bộ propagation cho mọi node một lần trong epoch thì chi phí chủ yếu là $O(K|\mathcal{E}|d)$ (vì mỗi tầng cần một sparse matmul trên cạnh).
- **Bộ nhớ:** lưu embedding cho mọi node $O((M + N)d)$; lưu cấu trúc đồ thị sparse $O(|\mathcal{E}|)$.

- **So sánh với GCN truyền thống:** LightGCN tránh các phép biến đổi tuyến tính per-layer, do đó giảm đáng kể chi phí nhân ma trận dense và kích thước tham số.

Các thiết lập huấn luyện và mẹo thực tế

- **Khởi tạo embedding:** ngẫu nhiên chuẩn (e.g. Xavier/He) hoặc sampling từ phân phối chuẩn nhỏ.
- **Kích thước embedding d :** phỏ biến 64–256; giá trị mặc định 64 hoặc 128 cho nhiều benchmark.
- **Số layers K :** thường 2–4; $K = 3$ là lựa chọn hay gấp. Tránh K quá lớn vì *over-smoothing*.
- **Batching & learning rate:** batch size 1024–4096, learning rate 1e-3 cho Adam là khởi điểm hợp lý.
- **Regularization:** weight decay (áp dụng lên $E^{(0)}$), early stopping theo validation recall/NDCG.
- **Dropout / edge dropout:** một số triển khai thêm edge dropout để tăng tính tổng quát hoá; cần thử nghiệm cẩn thận.
- **Tối ưu hoá tính toán:** nếu đồ thị rất lớn, thực hiện propagation toàn bộ embeddings mỗi vài epoch thay vì mỗi batch; hoặc dùng sampling-based GCN variants nếu cần throughput cực cao.

Ưu điểm

- **Đơn giản nhưng hiệu quả:** loại bỏ affine + non-linearity giảm tham số và giữ được hiệu năng; thực nghiệm trong paper cho thấy LightGCN vượt các GCN-based và MF-based models trên nhiều benchmark.
- **Dễ triển khai scale-up:** propagation chỉ dùng sparse matmul, thích hợp cho hệ thống lớn khi dùng kỹ thuật tính toán sparse.
- **Biểu diễn cộng dồn multi-hop:** bằng cách cộng/e weighted các lớp, LightGCN tận dụng thông tin multi-hop (cộng đồng) hiệu quả.

Hạn chế và nhược điểm

- **Cold-start:** giống nhiều phương pháp CF thuần túy, LightGCN cần các cạnh (interactions) để lan truyền; user/item mới không có cạnh gắp khó khăn trừ khi tích hợp side features.
- **Over-smoothing khi K lớn:** embedding node có thể trở nên giống nhau nếu lan truyền quá sâu.
- **Phụ thuộc vào chất lượng đồ thị:** nếu đồ thị quá thưa hoặc noisy, lợi ích của message passing bị suy giảm.
- **Không trực tiếp sử dụng side information:** bản gốc không kết hợp features content; phải mở rộng nếu muốn dùng text/image side features.

Mở rộng và biến thể phổ biến

- **Kết hợp features:** nối hoặc phỏng đại embedding ban đầu với embedding thu được từ side features (text, hình ảnh), rồi sử dụng LightGCN propagation.
- **Layer-wise learnable weights:** thay vì α_k cố định, học các hệ số trọng số layer và/hoặc áp dụng attention cho từng layer.
- **Edge / node dropout:** cải thiện khái quát hoá bằng cách ngẫu nhiên loại bỏ cạnh/lân cận trong huấn luyện.
- **Sampling-based propagation:** cho đồ thị cực lớn, áp dụng neighborhood sampling (như GraphSAGE style) để giảm chi phí mỗi bước.

Kết luận ngắn

LightGCN là một thiết kế thực dụng cho recommendation dựa trên đồ thị: nó chỉ giữ phần cốt lõi của GCN (neighborhood aggregation) và loại bỏ các thành phần không cần thiết cho collaborative filtering. Kết quả là mô hình vừa nhẹ, vừa mạnh, dễ triển khai ở quy mô lớn và trở thành baseline phổ biến cho các nghiên cứu tiếp theo về GCN trong recommendation.

3.3 Dữ liệu Explicit và Implicit Feedback

Trong các hệ thống gợi ý, dữ liệu về hành vi người dùng được chia thành hai loại chính: **explicit feedback** và **implicit feedback**. Việc hiểu rõ sự khác biệt giữa hai loại dữ liệu này rất quan trọng để lựa chọn mô hình gợi ý phù hợp.

Loại dữ liệu	Explicit Feedback	Implicit Feedback
Ví dụ	Rating, Purchase, Like	Click, View, Time Spent
Đặc điểm	Thể hiện rõ sở thích, giá trị cao, ít	Dễ thu thập, nhiều, tiềm ẩn
Khó khăn	Dữ liệu thưa, khó học mô hình	Có thể nhiều, không trực tiếp

Việc lựa chọn sử dụng explicit hay implicit feedback phụ thuộc vào hệ thống gợi ý và mục tiêu của bài toán, đồng thời ảnh hưởng đến các kỹ thuật học máy được áp dụng.

3.3.1 Explicit Feedback

Explicit feedback là dữ liệu mà người dùng cung cấp một cách trực tiếp, thể hiện rõ ràng mức độ yêu thích của họ đối với một item. Ví dụ:

- Rating: đánh giá 1–5 sao cho phim, sách hoặc sản phẩm.
- Purchase: hành vi mua hàng trực tiếp.
- Like / Dislike: thể hiện sở thích rõ ràng trên các nền tảng xã hội.

Đặc điểm:

- Dữ liệu có giá trị cao, trực tiếp phản ánh sở thích của người dùng.
- Dữ liệu thường **thưa thớt**, ít, và khó thu thập.
- Khó khăn khi học các mô hình do số lượng đánh giá hạn chế, đặc biệt trong hệ thống lớn.

3.3.2 Implicit Feedback

Implicit feedback là dữ liệu được gián tiếp thu thập từ hành vi người dùng, không yêu cầu người dùng đánh giá trực tiếp. Ví dụ:

- Click, view, scroll trên website hoặc ứng dụng.
- Thời gian xem, số lần tương tác với item.

Đặc điểm:

- Dữ liệu **nhiều và dễ thu thập**.

- Phản ánh tiềm ẩn mức độ ưa thích của người dùng, nhưng không trực tiếp như explicit feedback.
- Có thể có nhiều, vì hành vi không phải lúc nào cũng thể hiện sở thích thật sự.

3.3.3 Weighted Matrix Factorization (WMF) [11]

Weighted Matrix Factorization (WMF) là một phương pháp mở rộng của Matrix Factorization, được thiết kế đặc biệt để học từ **dữ liệu tương tác tiềm ẩn (implicit feedback)**, ví dụ như số lần xem video, thời lượng xem, lịch sử mua sắm, v.v.

Trong dữ liệu explicit feedback, các tương tác như rating, like, dislike cho ta biết rõ sở thích của người dùng. Ngược lại, trong **dữ liệu tiềm ẩn**:

- Mọi tương tác đều được xem là tích cực (positive feedback). Không có tương tác không đồng nghĩa với dislike.
- Các mức độ tương tác khác nhau có thể phản ánh mức độ ưu thích của user, ví dụ:

$$r_{ui} = 0.7 \text{ tức user } u \text{ đã xem } 70\% \text{ video } i.$$

WMF đưa ra hai ý tưởng chính:

1. **Confidence levels:** Gán trọng số c_{ui} cho mỗi tương tác dựa trên mức độ chắc chắn user thích item, ví dụ dựa trên số lần xem, thời lượng xem, v.v.
2. **Weighted loss function:** Tối ưu hàm mất mát của Matrix Factorization bằng cách cân nhắc trọng số c_{ui} .

Giả sử p_{ui} là phản hồi nhị phân:

$$p_{ui} = \begin{cases} 1, & \text{nếu user } u \text{ có tương tác với item } i \\ 0, & \text{không tương tác} \end{cases}$$

Và c_{ui} là confidence:

$$c_{ui} = 1 + \alpha r_{ui}$$

Hàm mất mát WMF:

$$\mathcal{L} = \sum_{u,i} c_{ui} (p_{ui} - U_u^\top V_i)^2 + \lambda (\|U_u\|^2 + \|V_i\|^2) \quad (3.8)$$

Trong đó:

- $U \in \mathbb{R}^{m \times k}$, $V \in \mathbb{R}^{n \times k}$ là ma trận biểu diễn ẩn của user và item.
- k là số chiều không gian ẩn.
- λ là tham số điều chỉnh.

Thuật toán huấn luyện bằng Alternating Least Squares (ALS) WMF thường được huấn luyện bằng ALS, với các bước:

Thuật toán 3.2 Weighted Matrix Factorization bằng ALS

Require: Ma trận phản hồi $P = [p_{ui}]$, ma trận confidence $C = [c_{ui}]$, số chiều ẩn k , điều chỉnh λ , số epoch T

- 1: Khởi tạo $U \in \mathbb{R}^{m \times k}$ và $V \in \mathbb{R}^{n \times k}$ ngẫu nhiên
- 2: **for** $t = 1$ đến T **do**
- 3: **for** mỗi user $u = 1$ đến m **do**
- 4: Cập nhật U_u bằng giải phương trình tuyến tính:

$$U_u \leftarrow (V^\top C^u V + \lambda I)^{-1} V^\top C^u P_u$$

Trong đó $C^u = \text{diag}(c_{u1}, \dots, c_{un})$, $P_u = [p_{u1}, \dots, p_{un}]^\top$

- 5: **end for**
- 6: **for** mỗi item $i = 1$ đến n **do**
- 7: Cập nhật V_i bằng giải phương trình tuyến tính:

$$V_i \leftarrow (U^\top C^i U + \lambda I)^{-1} U^\top C^i P_i$$

Trong đó $C^i = \text{diag}(c_{1i}, \dots, c_{mi})$, $P_i = [p_{1i}, \dots, p_{mi}]^\top$

- 8: **end for**
 - 9: **end for**
 - 10: **return** Ma trận biểu diễn ẩn U và V
-

Ưu điểm và hạn chế Ưu điểm:

- Thích hợp cho dữ liệu implicit feedback.
- Khai thác được các mức độ tương tác khác nhau thông qua confidence c_{ui} .

- ALS giúp huấn luyện hiệu quả trên ma trận thưa lớn.

Hạn chế:

- Không phân biệt negative feedback thực sự.
- Cần chọn hệ số α hợp lý.
- Dữ liệu cực kỳ thưa có thể làm một số tương tác ít bị đánh giá thấp.

3.3.4 Bayesian Personalized Ranking (BPR) [20]

Bayesian Personalized Ranking (BPR) là một phương pháp gợi ý được thiết kế để xếp hạng các item cho mỗi người dùng dựa trên dữ liệu **implicit feedback**. Thay vì dự đoán giá trị rating trực tiếp như trong Matrix Factorization truyền thống, BPR tối ưu thứ tự (ranking) của các item cho từng user.

Ý tưởng chính

- Mục tiêu là đưa ra một danh sách các item được xếp hạng theo sở thích của người dùng.
- Thứ tự này được suy ra từ các hành vi tiềm ẩn của người dùng, ví dụ click, view, mua hàng.
- Tiếp cận theo **cặp item**: cho mỗi user, mô hình học để item mà user đã tương tác (positive) xếp trên các item chưa quan sát (unobserved).

Dữ liệu tiềm ẩn

- Dữ liệu quan sát được là **positive feedback**.
- Các cặp user-item không quan sát:
 - Có thể là negative (user thực sự không thích)
 - Hoặc missing value (chưa biết)

Hàm mục tiêu BPR

Giả sử u là user, i là item mà user đã tương tác (positive), j là item chưa tương tác (unobserved). BPR học mô hình sao cho:

$$\hat{x}_{ui} > \hat{x}_{uj}$$

Trong đó \hat{x}_{ui} là dự đoán sở thích của user u với item i .

Hàm mất mát BPR được định nghĩa là:

$$\mathcal{L}_{BPR} = - \sum_{(u,i,j) \in \mathcal{D}} \ln \sigma(\hat{x}_{ui} - \hat{x}_{uj}) + \lambda \|\Theta\|^2 \quad (3.9)$$

Trong đó:

- \mathcal{D} là tập các triplet (u, i, j) , với i là item positive và j là item chưa quan sát.
- σ là hàm sigmoid: $\sigma(x) = 1/(1 + e^{-x})$.
- Θ là tất cả tham số của mô hình (ví dụ embeddings của user và item).
- λ là hệ số điều chỉnh.

Chi tiết đạo hàm của BPR loss

Ta xét một mẫu (triplet) duy nhất (u, i, j) với

$$\Delta_{uij} = \hat{x}_{ui} - \hat{x}_{uj}.$$

Hàm mất mát cho một triplet (không xét regularization) là

$$\ell_{uij} = -\ln \sigma(\Delta_{uij}), \quad \sigma(x) = \frac{1}{1 + e^{-x}}.$$

1. Đạo hàm theo Δ . Áp dụng quy tắc chuỗi:

$$\frac{\partial \ell_{uij}}{\partial \Delta_{uij}} = -\frac{1}{\sigma(\Delta_{uij})} \sigma'(\Delta_{uij}) = -\frac{1}{\sigma(\Delta_{uij})} \sigma(\Delta_{uij})(1 - \sigma(\Delta_{uij})) = -(1 - \sigma(\Delta_{uij})).$$

Đặt kí hiệu

$$\delta_{uij} = 1 - \sigma(\Delta_{uij}) = \sigma(-\Delta_{uij}) \geq 0,$$

vậy

$$\frac{\partial \ell_{uij}}{\partial \Delta_{uij}} = -\delta_{uij}.$$

2. Trường hợp Matrix Factorization (ví dụ phổ biến). Giả sử mô hình là MF tuyến tính:

$$\hat{x}_{ui} = p_u^\top q_i, \quad \hat{x}_{uj} = p_u^\top q_j,$$

với tham số là các embedding p_u, q_i, q_j . Khi đó

$$\Delta_{uij} = p_u^\top (q_i - q_j).$$

Tính các đạo hàm con theo tham số (không xét regularization):

$$\begin{aligned}\frac{\partial \ell_{uij}}{\partial p_u} &= \frac{\partial \ell_{uij}}{\partial \Delta_{uij}} \cdot \frac{\partial \Delta_{uij}}{\partial p_u} = -\delta_{uij} (q_i - q_j), \\ \frac{\partial \ell_{uij}}{\partial q_i} &= \frac{\partial \ell_{uij}}{\partial \Delta_{uij}} \cdot \frac{\partial \Delta_{uij}}{\partial q_i} = -\delta_{uij} p_u, \\ \frac{\partial \ell_{uij}}{\partial q_j} &= \frac{\partial \ell_{uij}}{\partial \Delta_{uij}} \cdot \frac{\partial \Delta_{uij}}{\partial q_j} = -\delta_{uij} (-p_u) = \delta_{uij} p_u.\end{aligned}$$

3. Thêm regularization $\lambda \|\Theta\|^2$. Nếu loss tổng cho batch có dạng

$$\mathcal{L} = \sum_{(u,i,j) \in \mathcal{D}} \ell_{uij} + \lambda \sum_{\theta \in \Theta} \|\theta\|^2,$$

với biểu thức regularization như bạn dùng ($+\lambda \|\Theta\|^2$), thì đạo hàm có thêm thành phần $2\lambda\theta$. Do đó, gradient toàn phần là:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_u} &= -\delta_{uij} (q_i - q_j) + 2\lambda p_u, \\ \frac{\partial \mathcal{L}}{\partial q_i} &= -\delta_{uij} p_u + 2\lambda q_i, \\ \frac{\partial \mathcal{L}}{\partial q_j} &= \delta_{uij} p_u + 2\lambda q_j.\end{aligned}$$

4. Quy tắc cập nhật SGD (learning rate η). Với cập nhật SGD/mini-batch (giả sử cập nhật theo chiều giảm gradient):

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta},$$

ta thu được quy tắc cập nhật cho triplet (u, i, j) :

$$\begin{aligned} p_u &\leftarrow p_u + \eta \left(\delta_{uji} (q_i - q_j) - 2\lambda p_u \right), \\ q_i &\leftarrow q_i + \eta \left(\delta_{uji} p_u - 2\lambda q_i \right), \\ q_j &\leftarrow q_j + \eta \left(-\delta_{uji} p_u - 2\lambda q_j \right). \end{aligned}$$

(Nhóm dấu trên tương đương với công thức: $p_u \leftarrow p_u - \eta \frac{\partial \mathcal{L}}{\partial p_u}$, v.v.)

5. Những biến thể/ghi chú thực tế

- Nếu regularization được viết là $\frac{\lambda}{2} \|\Theta\|^2$, thì thành phần gradient regularization là $\lambda\theta$ thay vì $2\lambda\theta$. Hãy kiểm tra convention trong code của bạn trước khi áp dụng.
- Nếu model có bias b_i hoặc b_u và/hoặc feature-based terms (ví dụ $\hat{x}_{ui} = p_u^\top q_i + b_i + b_u$), thì

$$\frac{\partial \ell_{uji}}{\partial b_i} = -\delta_{uji}, \quad \frac{\partial \ell_{uji}}{\partial b_j} = \delta_{uji},$$

và cập nhật tương ứng (cộng trừ learning-rate) kèm regularization.

- Việc dùng $\delta_{uji} = \sigma(-\Delta_{uji})$ giúp biểu thức luôn không âm và dễ triển khai hiệu quả.
- Trong thực tế thường dùng vectorized update trên mini-batch: tính toàn bộ các Δ cho batch, compute $\delta = \sigma(-\Delta)$ element-wise, rồi apply cập nhật ma trận (matrix operations) để tận dụng tốc độ GPU/BLAS.

Thuật toán huấn luyện

BPR thường được huấn luyện bằng **stochastic gradient descent (SGD)** trên các triplet (u, i, j) :

Thuật toán 3.3 BPR bằng SGD

Require: Tập dữ liệu implicit feedback P , embedding user/item khởi tạo Θ , learning rate η , điều chỉnh λ , số epoch T

- 1: **for** $t = 1$ đến T **do**
- 2: Lấy ngẫu nhiên user u
- 3: Lấy ngẫu nhiên item i mà u đã tương tác (positive)
- 4: Lấy ngẫu nhiên item j chưa tương tác bởi u (unobserved)
- 5: Tính gradient của hàm mất mát:

$$\frac{\partial \mathcal{L}_{BPR}}{\partial \Theta} = -(1 - \sigma(\hat{x}_{ui} - \hat{x}_{uj})) \frac{\partial(\hat{x}_{ui} - \hat{x}_{uj})}{\partial \Theta} + 2\lambda\Theta$$

- 6: Cập nhật tham số: $\Theta \leftarrow \Theta - \eta \frac{\partial \mathcal{L}_{BPR}}{\partial \Theta}$
 - 7: **end for**
 - 8: **return** Embedding user và item
-

Ưu điểm

- Học trực tiếp thứ tự xếp hạng, phù hợp cho dữ liệu implicit feedback.
- Khai thác tốt thông tin từ các cặp item positive/unobserved.
- Linh hoạt áp dụng với MF, neural CF hoặc GCN.

Hạn chế

- Không phân biệt được unobserved là negative hay missing.
- Yêu cầu sampling triplet để huấn luyện, tốn thời gian với dữ liệu lớn.

3.4 Sử dụng đồng thời Implicit và Explicit Feedback

3.4.1 Tổng quan

Trong các hệ thống gợi ý hiện đại, việc chỉ sử dụng một loại dữ liệu phản hồi (explicit hoặc implicit) đôi khi không đủ để mô hình hóa sở thích người dùng một cách chính xác. Do đó, nhiều nghiên cứu và ứng dụng thực tế đề xuất kết hợp đồng thời cả **explicit feedback** và **implicit feedback** để cải thiện chất lượng gợi ý.

Mục tiêu

- Tận dụng cả hai nguồn dữ liệu: explicit feedback cung cấp thông tin rõ ràng về sở thích, implicit feedback cung cấp thông tin phong phú, nhiều và dễ thu thập.
- Cải thiện dự đoán rating hoặc xếp hạng item, đặc biệt khi dữ liệu explicit thưa thớt.
- Giảm thiểu nhiễu từ dữ liệu implicit bằng cách kết hợp với explicit feedback.

Các cách tiếp cận

1. **Weighted combination:** Gán trọng số cho explicit và implicit feedback trong hàm mất mát, ví dụ:

$$\mathcal{L} = \alpha \mathcal{L}_{explicit} + (1 - \alpha) \mathcal{L}_{implicit}$$

với $\alpha \in [0, 1]$ điều chỉnh mức độ ảnh hưởng của từng loại dữ liệu.

2. **Multi-task learning:** Học các embeddings chung cho user và item, sau đó dùng chúng để dự đoán cả explicit ratings và implicit preferences.
3. **Hybrid models:** Kết hợp các mô hình riêng biệt:

- Sử dụng MF, NeuMF hoặc Deep MF cho explicit feedback.
- Sử dụng BPR hoặc WMF cho implicit feedback.
- Kết hợp đầu ra hoặc embeddings từ cả hai mô hình.

Lợi ích

- Tăng độ chính xác của gợi ý, đặc biệt trong các hệ thống có dữ liệu explicit thưa.
- Khai thác triệt để các nguồn dữ liệu có sẵn.
- Hỗ trợ mô hình hóa sở thích phức tạp và cá nhân hóa hơn.

Hạn chế

- Cần thiết kế hàm mất mát hoặc trọng số hợp lý để cân bằng hai nguồn dữ liệu.

- Tăng độ phức tạp mô hình và thời gian huấn luyện.
- Dữ liệu implicit vẫn có thể chứa nhiễu, cần xử lý trước khi kết hợp.

Ứng dụng thực tế

Các hệ thống thương mại điện tử, streaming, và mạng xã hội thường áp dụng kết hợp explicit và implicit feedback để:

- Gợi ý sản phẩm, video hoặc bài hát phù hợp.
- Cập nhật thứ tự xếp hạng item cho từng người dùng.
- Học embeddings user/item phong phú hơn, phục vụ các mô hình gợi ý dự đoán.

3.4.2 Co-Rating [17]

Co-Rating là một phương pháp hybrid, cho phép sử dụng đồng thời cả **explicit feedback (X)** và **implicit feedback (Y)** để cải thiện hiệu quả gợi ý.

Ý tưởng chính

- Explicit feedback X : các giá trị rating, like/dislike, thể hiện rõ sở thích người dùng.
- Implicit feedback Y : hành vi tiềm ẩn như click, view, thời lượng xem, số lần mua hàng.
- Co-Rating kết hợp cả hai loại dữ liệu này để học embeddings user/item phong phú và mô hình hóa sở thích chính xác hơn.

Mô hình

- Học hai ma trận ẩn U (user) và V (item) dùng cho cả explicit và implicit feedback.
- Dự đoán explicit rating:

$$\hat{X}_{ui} = U_u^\top V_i$$

- Dự đoán implicit preference (tiềm ẩn):

$$\hat{Y}_{ui} = f(U_u^\top V_i)$$

Trong đó f là một hàm phi tuyến (ví dụ sigmoid) để chuyển sang xác suất tương tác.

Hàm mất mát

Hàm mất mát tổng hợp gồm hai thành phần:

$$\mathcal{L} = \alpha \sum_{(u,i) \in X} (X_{ui} - \hat{X}_{ui})^2 + (1 - \alpha) \sum_{(u,i) \in Y} c_{ui} \ell(\hat{Y}_{ui}, Y_{ui}) + \lambda(\|U\|^2 + \|V\|^2)$$

Trong đó:

- $\alpha \in [0, 1]$ điều chỉnh mức độ ảnh hưởng của explicit và implicit feedback.
- ℓ là hàm mất mát cho dữ liệu implicit (ví dụ BPR loss hoặc weighted squared error).
- c_{ui} là trọng số confidence của tương tác implicit.
- λ là hệ số điều chuẩn.

Ưu điểm

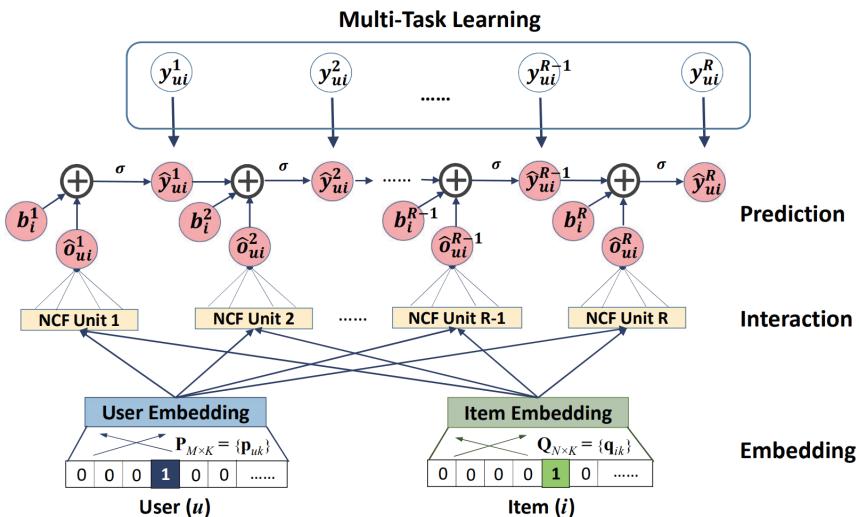
- Kết hợp được thông tin trực tiếp từ explicit feedback và thông tin phong phú từ implicit feedback.
- Học embeddings user/item thống nhất, phục vụ cho cả dự đoán rating và ranking item.
- Giúp cải thiện độ chính xác, đặc biệt khi explicit feedback thưa thớt.

Hạn chế

- Cần cân bằng hợp lý giữa explicit và implicit feedback thông qua α .
- Tăng độ phức tạp mô hình và thời gian huấn luyện.
- Implicit feedback vẫn có thể chứa nhiều, cần xử lý trước khi huấn luyện.

3.4.3 Phương pháp NMTR [6]

Neural Multi-task Recommendation (NMTR) [6] là một phương pháp gọi ý tận dụng *multi-behavior data* (nhiều loại hành vi của user — ví dụ: view, click, add-to-cart, purchase). Thay vì chỉ tối ưu cho một loại hành vi mục tiêu (ví dụ purchase), NMTR học đồng thời nhiều nhiệm vụ (multi-task) tương ứng với các loại hành vi, và khai thác mối quan hệ phụ thuộc/cascading giữa chúng (ví dụ view → click → add_to_cart → purchase). Mục tiêu là cải thiện hiệu năng cho task mục tiêu bằng cách chia sẻ biểu diễn và thông tin giữa các task liên quan (Hình 3.4).



Hình 3.4: Sơ đồ tổng quan NMTR: embeddings chung, các task (behavior) có tower đặc thù, và cơ chế cascading/condition giữa các task.

Các loại hành vi và ma trận tương tác đa hình thức

Giả sử có R loại hành vi khác nhau (ví dụ: xem, nhấp chuột, thêm vào giỏ, mua hàng) được sắp theo thứ tự từ yếu đến mạnh:

$$Y_1 \rightarrow Y_2 \rightarrow \dots \rightarrow Y_R,$$

trong đó Y_R là hành vi mục tiêu (ví dụ mua hàng) cần tối ưu. Mỗi loại hành vi r được biểu diễn bởi một ma trận nhị phân kích thước $M \times N$ (với M người dùng, N item):

$$y_{ui}^r = \begin{cases} 1, & \text{nếu user } u \text{ thực hiện hành vi loại } r \text{ trên item } i, \\ 0, & \text{ngược lại.} \end{cases}$$

Sự giả thiết quan trọng trong NMTR là các hành vi có quan hệ thứ tự (cascaded): một hành vi cấp cao chỉ xảy ra nếu các hành vi cấp thấp hơn đã xảy ra.

Kiến trúc tổng thể của NMTR

NMTR là mô hình mạng nơ-ron đa nhiệm (multi-task) chia sẻ tầng nhúng (embedding) và có các nhánh mạng riêng cho từng loại hành vi.

- Embedding dùng chung:

$$p_u = P^\top v_u^U, \quad q_i = Q^\top v_i^I,$$

trong đó v_u^U và v_i^I là vector one-hot mã hóa ID người dùng và item, $P \in \mathbb{R}^{M \times E}$, $Q \in \mathbb{R}^{N \times E}$ là ma trận nhúng, E là số chiều embedding.

- Output của từng nhánh cho hành vi r :

$$\hat{y}_{ui}^r = \sigma(f_\Theta^r(p_u, q_i)),$$

với f_Θ^r là mạng neural riêng cho hành vi r , σ là hàm sigmoid.

Mô hình hóa phụ thuộc tuần tự của các hành vi (Cascaded Predictions)

$$\begin{aligned}\hat{y}_{ui}^1 &= \sigma(f_\Theta^1(p_u, q_i) + b_i^1), \\ \hat{y}_{ui}^2 &= \sigma(\hat{y}_{ui}^1 + f_\Theta^2(p_u, q_i) + b_i^2), \\ &\vdots \\ \hat{y}_{ui}^R &= \sigma(\hat{y}_{ui}^{R-1} + f_\Theta^R(p_u, q_i) + b_i^R),\end{aligned}$$

trong đó b_i^r là bias của item i cho hành vi thứ r . Thiết kế này giúp hành vi cấp cao được dự đoán dựa trên kết quả hành vi cấp thấp, cải thiện dự đoán cho hành vi mục tiêu khi dữ liệu ít.

Hàm mất mát đa nhiệm và huấn luyện

$$L = - \sum_{r=1}^R \lambda_r \left(\sum_{(u,i) \in Y_r^+} \log \hat{y}_{ui}^r + \sum_{(u,i) \in Y_r^-} \log(1 - \hat{y}_{ui}^r) \right),$$

trong đó:

- Y_r^+ là tập cặp (u, i) có tương tác thực sự cho hành vi r ,

- Y_r^- là tập các mẫu âm (không tương tác),
- λ_r là trọng số điều chỉnh tầm quan trọng của hành vi r .

Hàm mất mát được tối ưu bằng SGD hoặc các biến thể (Adam, AdaGrad) cập nhật đồng thời các nhánh mạng và embedding dùng chung.

Ưu điểm, hạn chế và ứng dụng

Ưu điểm:

- Khai thác tín hiệu từ nhiều hành vi, cải thiện dự đoán hành vi mục tiêu.
- Embedding user/item phong phú và tổng quát hơn.
- Cascaded predictions truyền tín hiệu từ hành vi phổ biến đến hành vi ít phổ biến.

Hạn chế:

- Độ phức tạp tính toán cao và nhiều siêu tham số cần điều chỉnh.
- Giả định thứ tự hoàn chỉnh giữa các hành vi, có thể không phù hợp với dữ liệu thực tế.
- Cần tập dữ liệu đủ lớn để tránh overfitting.

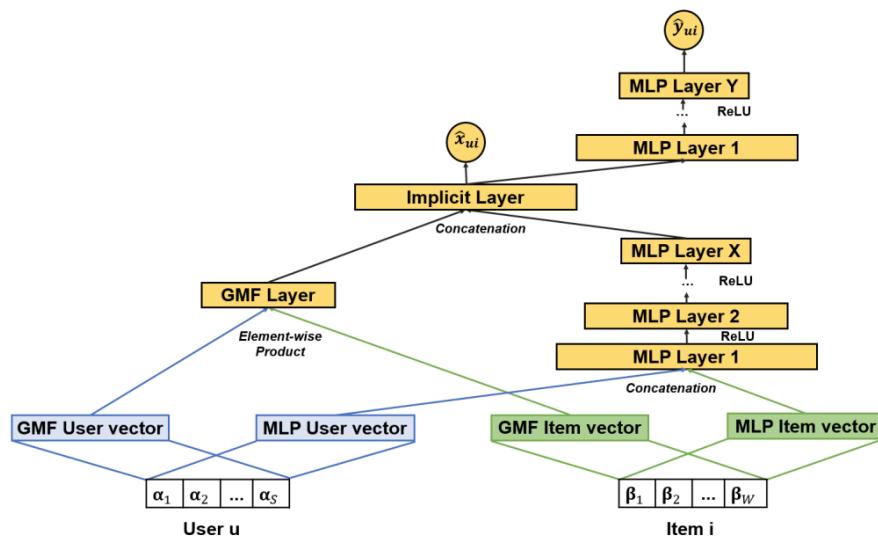
Ứng dụng:

- Hệ thống thương mại điện tử: xem – nhấp – thêm giỏ – mua hàng.
- Hệ thống video hoặc tin tức: xem – like – chia sẻ/đăng ký.
- Hỗ trợ gợi ý cho người dùng mới hoặc ít dữ liệu bằng cách truyền tín hiệu từ hành vi trung gian.

3.4.4 Mô hình ITE (Implicit to Explicit Feedback [26])

Nhóm tác giả Phan [26]) đề xuất một kiến trúc deep learning tên là **Implicit to Explicit (ITE)** nhằm mô hình hoá quan hệ tuần tự giữa các hành vi của người dùng — từ các hành vi *implicit* (ví dụ: view, click) đến các hành vi *explicit* (ví dụ: add-to-cart, purchase). Tác giả cũng trình bày biến thể **ITE-Si** (ITE with Side information) để kết hợp thông tin phụ trợ (category, tags, mô tả item) vào biểu diễn user/item. Thử nghiệm trên hai tập lớn (ví dụ: RetailRocket, Recobell) cho thấy ITE/ITE-Si vượt các baseline.

Kiến trúc tổng quan



Hình 3.5: Minh họa khái niệm module *implicit* và module *explicit*; output của module *implicit* (last hidden layer) được đưa vào module *explicit* (first layer) để mô tả quá trình *implicit* → *explicit*.

Những điểm chính của ITE:

- ITE là tổ hợp của hai *feed-forward* modules: một module học tương tác *implicit* (gọi ngắn là \mathcal{M}_{imp}) và một module học tương tác *explicit* (gọi là \mathcal{M}_{exp}).
- Ý tưởng then chốt: thông tin ngầm (*implicit*) ảnh hưởng đến quyết định công khai (*explicit*), nên ta *chuyển* thông tin (feature vector / hidden representation) từ module *implicit* sang module *explicit* (last hidden của *implicit* nối vào first layer của *explicit*).
- ITE-Si: tương tự ITE nhưng embedding user/item được nâng cấp bằng side information (category, tags, textual features) trước khi đưa vào các module.

Biểu diễn toán học (ký hiệu chính)

- $X = (x_{ui}) \in \{0, 1\}^{M \times N}$: ma trận *implicit* ($x_{ui} = 1$ nếu user u có hành vi *implicit* với item i).
- $Y = (y_{ui}) \in \{0, 1\}^{M \times N}$: ma trận *explicit* ($y_{ui} = 1$ nếu user u có hành vi *explicit* với item i).

- u, i (hoặc p_u, q_i) là vector đại diện user/item (có thể one-hot map qua embedding layer hoặc enriched bằng side-info).
- Hai module neural:

$$\hat{x}_{ui} = \mathcal{M}_{imp}(u, i; \Theta_{imp}), \quad \hat{y}_{ui} = \mathcal{M}_{exp}(u, i, z_{ui}; \Theta_{exp}),$$

trong đó z_{ui} là hidden representation xuất phát từ module implicit (thường lấy là *last hidden layer* của \mathcal{M}_{imp}), tức biểu diễn tóm tắt thông tin implicit liên quan tới cặp (u, i) .

Cách nối hai module (Implicit → Explicit) Cụ thể, nếu \mathcal{M}_{imp} có các lớp ẩn và $h_{ui}^{(L_{imp})}$ là vector ở lớp ẩn cuối cùng, thì \mathcal{M}_{exp} nhận đầu vào:

$$\phi_{ui} = [\text{emb}_u ; \text{emb}_i ; h_{ui}^{(L_{imp})}],$$

và sau đó đi qua một hoặc nhiều lớp feed-forward để sinh \hat{y}_{ui} . (Trong ITE-Si, $\text{emb}_u, \text{emb}_i$ được ghép thêm các embedding từ side information trước khi concat).

Hàm mất mát (Loss)

Tác giả dùng **binary cross-entropy** (logistic loss) cho cả hai hành vi (implicit và explicit), kết hợp negative sampling từ các ô chưa quan sát. Tổng loss huấn luyện là tổng hợp của hai thành phần (implicit + explicit) cùng regularization:

$$\mathcal{L} = \mathcal{L}_{imp} + \gamma \mathcal{L}_{exp} + \lambda \Omega(\Theta).$$

Trong đó

$$\begin{aligned} \mathcal{L}_{imp} &= - \sum_{(u,i) \in X^+ \cup X^-} [x_{ui} \log \hat{x}_{ui} + (1 - x_{ui}) \log (1 - \hat{x}_{ui})], \\ \mathcal{L}_{exp} &= - \sum_{(u,i) \in Y^+ \cup Y^-} [y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log (1 - \hat{y}_{ui})], \end{aligned}$$

với X^+, Y^+ là tập các tương tác quan sát, X^-, Y^- là negative samples được rút từ các ô chưa quan sát. Hệ số γ (trong paper ký hiệu tương đương là một tham số cân bằng, có khi gọi η hoặc tương tự) dùng để điều chỉnh tầm ảnh hưởng của loss explicit so với implicit khi training. Regularization $\Omega(\Theta)$ thường là L_2 trên embedding/weights.

Huấn luyện

- Huấn luyện bằng mini-batch SGD (Adam / Adagrad / RMSProp có thể sử dụng). Mỗi batch gồm các positive samples (từ X^+ và Y^+) và negative samples được lấy ngẫu nhiên từ vùng chưa quan sát.
- Các tham số quan trọng: learning rate, batch size, kích thước embedding, số lớp và kích thước ẩn cho hai module, tỉ lệ negative sampling, và hệ số cân bằng γ giữa hai losses. Tác giả đề nghị điều chỉnh γ để tối ưu hiệu năng task explicit (mục tiêu thực tế như purchase).
- Với ITE-Si, side information được tiền xử lý (one-hot / embedding / avg word embeddings tùy loại) rồi ghép vào embedding user/item trước khi đưa vào mạng. Việc này giúp giảm cold-start cho các item/user có ít tương tác.

Pseudo-code huấn luyện (ý tưởng)

Thuật toán 3.4 Huấn luyện ITE (một epoch, ý tưởng)

- 1: Khởi tạo tham số $\Theta = \{\Theta_{imp}, \Theta_{exp}\}$
 - 2: **for** each minibatch **do**
 - 3: Lấy batch các cặp positive implicit $(u, i) \in X^+$ và positive explicit $(u, i) \in Y^+$
 - 4: Lấy negative samples tương ứng cho mỗi type
 - 5: Forward: compute $\hat{x}_{ui} = \mathcal{M}_{imp}(u, i)$ và lấy $h_{ui}^{(L_{imp})}$
 - 6: Forward explicit: $\hat{y}_{ui} = \mathcal{M}_{exp}(u, i, h_{ui}^{(L_{imp})})$
 - 7: Tính $\mathcal{L}_{imp}, \mathcal{L}_{exp}$, tổng loss \mathcal{L}
 - 8: Backpropagate và cập nhật Θ
 - 9: **end for**
-

Kết quả và thí nghiệm

Tác giả đánh giá trên hai bộ dữ liệu lớn (RetailRocket, Recobell) và so sánh với các baseline bao gồm MF-based và một số phương pháp multi-behavior / multi-task (ví dụ MTMF, NMTR). Kết quả cho thấy ITE cải thiện các chỉ số đánh giá (Recall / NDCG / MAP tùy báo cáo) so với các baseline; ITE-Si (khi có side-info) đạt hiệu năng tốt hơn ITE cơ bản, chứng tỏ lợi ích của side information trong giảm cold-start và nâng cao chất lượng dự đoán explicit.

Ưu điểm

- **Mô hình hoá thứ tự hành vi:** thiết kế hai-phase (implicit → explicit) bắt đúng giả thiết thực tế rằng implicit precedes explicit, giúp nắm bắt mối quan hệ ngữ nghĩa phức tạp giữa các hành động.
- **Linh hoạt với side information:** ITE-Si cho phép tích hợp nhiều nguồn phụ trợ để cải thiện biểu diễn và giảm cold-start.
- **Dễ tích hợp vào pipeline hiện tại:** architecture feed-forward đơn giản, tương thích với negative sampling và training theo mini-batch.

Hạn chế

- **Phụ thuộc vào quality của negative sampling:** như nhiều mô hình binary-cross-entropy trên dữ liệu implicit/explicit, hiệu quả phụ thuộc vào chiến lược sampling âm.
- **Thiết kế kiến trúc và cân bằng loss:** cần tinh chỉnh tham số cân bằng giữa implicit/explicit (γ / η) — nếu cân không hợp lý có thể khiến module implicit hoặc explicit át hẳn module kia.
- **Không trực tiếp mô hình hoá thứ tự thời gian dài hạn:** ITE là mạng feed-forward trong từng điểm (với thông tin sequence tóm tắt qua hidden vector), nhưng nếu cần mô hình hoá chuỗi thời gian dài (session / sequential patterns) có thể cần biến thể RNN/Transformer (tác giả trong các công trình tiếp theo cũng khám phá BERT-based variants).

Mở rộng (những hướng tác giả / cộng đồng gợi ý)

- **BERT-ITE / BERT-ITE-Si (hậu bản / mở rộng):** sau ACML, tác giả và nhóm có báo cáo mở rộng [25] kết hợp BERT-like encoder để mô hình hoá đồng thời long-term và short-term preferences, cải thiện hơn nữa khi cần nắm bắt ngữ cảnh tuần tự phức tạp.
- **Kết hợp với GCN / graph-based encoder:** có thể nâng cấp bước embedding bằng encoder graph để tận dụng mối liên hệ user-item.

Kết luận ngắn

ITE/ITE-Si là một giải pháp thực dụng và hiệu quả để kết hợp implicit và explicit behaviors theo đúng thứ tự logic của hành vi người dùng. Mô hình cho

thấy cải thiện thực nghiệm trên nhiều dataset lớn, và là nền tảng để mở rộng thêm (BERT-based, side-info fusion, graph encoders).

Chương 4

GỢI Ý DỰA TRÊN NỘI DUNG

4.1	Tổng quan gợi ý dựa trên nội dung (Content-Based Recommendation)	77
4.2	Biểu diễn Nội dung và Sự Tương đồng Nội dung	78
4.3	Thảo luận (Discussion)	84
4.4	Tóm tắt (Summary)	85

4.1 Tổng quan gợi ý dựa trên nội dung (Content-Based Recommendation)

Từ những gì đã thảo luận cho đến nay, chúng ta thấy rằng đối với các kỹ thuật lọc cộng tác (collaborative filtering), ngoại trừ các đánh giá của người dùng, không cần phải biết gì về các sản phẩm được gợi ý. Lợi thế chính của phương pháp này là tránh được nhiệm vụ tốn kém trong việc cung cấp mô tả chi tiết và cập nhật cho hệ thống. Tuy nhiên, mặt trái của phương pháp này là với phương pháp lọc cộng tác thuần túy, một cách rất trực quan để chọn các sản phẩm có thể gợi ý dựa trên đặc điểm và sở thích của người dùng không thể thực hiện được. Trong thế giới thực, ví dụ, việc gợi ý cuốn sách mới của Harry Potter cho Alice là điều dễ dàng nếu chúng ta biết rằng (a) cuốn sách này là một tiểu thuyết kỳ ảo và (b) Alice luôn thích những tiểu thuyết kỳ ảo. Một hệ thống gợi ý điện tử có thể hoàn thành nhiệm vụ này chỉ khi có hai thông tin sẵn có: mô tả về đặc điểm của sản phẩm và hồ sơ người dùng

mô tả (các) sở thích trước đây của người dùng, có thể dưới dạng các đặc điểm ưu thích của sản phẩm. Nhiệm vụ gợi ý sau đó là xác định các sản phẩm phù hợp nhất với sở thích của người dùng. Quá trình này thường được gọi là gợi ý dựa trên nội dung. Mặc dù phương pháp này yêu cầu thông tin bổ sung về sản phẩm và sở thích người dùng, nhưng nó không yêu cầu sự tồn tại của một cộng đồng người dùng lớn hoặc lịch sử đánh giá – tức là, danh sách gợi ý có thể được tạo ra ngay cả khi chỉ có một người dùng duy nhất.

4.2 Biểu diễn Nội dung và Sự Tương đồng Nội dung

Cách đơn giản nhất để mô tả các sản phẩm trong danh mục là duy trì một danh sách các đặc điểm cho mỗi sản phẩm (thường gọi là thuộc tính, đặc điểm hoặc hồ sơ sản phẩm). Đối với hệ thống gợi ý sách, ví dụ, ta có thể sử dụng thẻ loại, tên tác giả, nhà xuất bản hoặc bất cứ điều gì mô tả sản phẩm và lưu trữ thông tin này trong hệ thống cơ sở dữ liệu quan hệ. Khi sở thích của người dùng được mô tả bằng các đặc điểm này, nhiệm vụ gợi ý sẽ là khớp các đặc điểm của sản phẩm với sở thích của người dùng.

```
[  
 {  
   "Tiêu đề": "The Night of the Gun",  
   "Thể loại": "Memoir",  
   "Tác giả": "David Carr",  
   "Loại": "Paperback",  
   "Giá": 29.90,  
   "Từ khóa": ["Press", "Journalism", "Addiction", "Personal Memoirs", "New York"],  
 },  
 {  
   "Tiêu đề": "The Lace Reader",  
   "Thể loại": "Fiction, Mystery",  
   "Tác giả": "Brunonia Barry",  
   "Loại": "Hardcover",  
   "Giá": 49.90,  
   "Từ khóa": ["American", "Fiction", "Detective", "Historical"]  
 },  
 {  
   "Tiêu đề": "Into the Fire",  
 }
```

```

    "Thể loại": "Romance, Suspense",
    "Tác giả": "Suzanne Brockmann",
    "Loại": "Hardcover",
    "Giá": 45.90,
    "Từ khóa": ["American", "Fiction", "Murder", "Neo-Nazism"]
}
]

```

4.2.1 Mô hình không gian-véc-tơ và TF-IDF

Nói một cách chính xác, thông tin về nhà xuất bản và tác giả thực sự không phải là "nội dung" của một cuốn sách, mà là kiến thức bổ sung về nó. Tuy nhiên, các hệ thống dựa trên nội dung đã được phát triển để lọc và gợi ý các sản phẩm dựa trên văn bản như tin nhắn e-mail hoặc tin tức. Phương pháp chuẩn trong gợi ý dựa trên nội dung do đó không phải là duy trì một danh sách các đặc điểm "siêu thông tin", như trong ví dụ trước, mà là sử dụng một danh sách các từ khóa liên quan xuất hiện trong tài liệu. Ý tưởng chính, tất nhiên, là danh sách này có thể được tạo tự động từ chính nội dung tài liệu hoặc từ mô tả văn bản tự do của nó.

Nội dung của một tài liệu có thể được mã hóa trong danh sách từ khóa này theo nhiều cách khác nhau. Trong phương pháp đơn giản và cơ bản nhất, ta có thể lập danh sách tất cả các từ xuất hiện trong tất cả các tài liệu và mô tả mỗi tài liệu bằng một véc-tơ Boolean, trong đó "1" chỉ ra rằng một từ xuất hiện trong tài liệu và "0" chỉ ra rằng từ đó không xuất hiện. Nếu hồ sơ người dùng được mô tả bằng một danh sách tương tự (trong đó "1" chỉ ra sự quan tâm đến một từ khóa), việc khớp tài liệu có thể thực hiện bằng cách đo lường sự trùng khớp giữa sở thích và nội dung tài liệu.

Để đưa ra gợi ý, các hệ thống dựa trên nội dung (content-based systems) thường hoạt động bằng cách đánh giá mức độ một mục (item) chưa được xem có "tương tự" đến đâu so với các mục mà người dùng hiện tại đã thích trong quá khứ. Mức độ tương tự có thể được đo lường theo nhiều cách khác nhau.

Ví dụ, với một cuốn sách chưa được xem B , hệ thống có thể đơn giản kiểm tra xem thể loại (genre) của cuốn sách đó có nằm trong danh sách thể loại ưa thích của Alice hay không. Trong trường hợp này, độ tương tự chỉ nhận giá trị 0 hoặc 1.

Một lựa chọn khác là tính toán độ tương tự hoặc mức độ chồng lấn (overlap) của các từ khóa liên quan. Là một thước đo điển hình về độ tương tự, phù hợp cho các đặc trưng đa trị, ta có thể dựa vào **hệ số Dice (Dice coefficient)** như sau: nếu mỗi cuốn sách B_i được mô tả bằng một tập hợp từ khóa $keywords(B_i)$, thì hệ

số Dice đo lường độ tương tự giữa hai cuốn sách b_i và b_j theo công thức:

$$\text{similarity}(b_i, b_j) = \frac{2 \times |\text{keywords}(b_i) \cap \text{keywords}(b_j)|}{|\text{keywords}(b_i)| + |\text{keywords}(b_j)|}$$

Về nguyên tắc, tùy thuộc vào bài toán cụ thể, có thể áp dụng nhiều thước đo độ tương tự khác nhau. Ví dụ, trong nghiên cứu của [33], một phương pháp được đề xuất trong đó nhiều hàm độ tương tự cho các đặc trưng khác nhau của mục được sử dụng. Các hàm này sau đó được kết hợp và gán trọng số để tính toán một thước đo độ tương tự tổng thể, đặc biệt trong những trường hợp vừa có mô tả cấu trúc vừa có mô tả phi cấu trúc về mục.

4.2.2 Mô hình vector và TF-IDF

Trong gợi ý dựa trên nội dung, tài liệu thường được mô tả bằng các từ khóa trích xuất từ nội dung. Biểu diễn Boolean (1 nếu từ xuất hiện, 0 nếu không) quá đơn giản và dễ thiên lệch về văn bản dài.

Do đó, phương pháp **TF-IDF** được sử dụng. Mỗi tài liệu được biểu diễn thành vector trong không gian từ khóa, với trọng số cho mỗi từ i trong tài liệu j được tính như sau:

$$TF(i, j) = \frac{\text{freq}(i, j)}{\max_{z \in \text{OtherKeywords}(i, j)} \text{freq}(z, j)}$$

$$IDF(i) = \log \frac{N}{n(i)}$$

$$TF-IDF(i, j) = TF(i, j) \times IDF(i)$$

Trong đó $\text{freq}(i, j)$ là số lần xuất hiện từ i trong j , N là tổng số tài liệu và $n(i)$ là số tài liệu chứa từ i . Nhờ vậy, tài liệu được biểu diễn không chỉ bởi sự xuất hiện của từ, mà còn bởi mức độ đặc trưng và quan trọng của chúng.

4.2.3 Tìm kiếm Dựa trên Sự Tương Đồng

Khi vấn đề lựa chọn sản phẩm trong lọc cộng tác có thể được mô tả là "gợi ý những sản phẩm mà người dùng tương tự đã thích", thì gợi ý dựa trên nội dung thường được mô tả là "gợi ý những sản phẩm tương tự với những sản phẩm mà người dùng đã thích trong quá khứ". Do đó, nhiệm vụ của hệ thống gợi ý là dự đoán, dựa trên hồ sơ của người dùng, liệu người dùng có thích một sản phẩm chưa

từng thấy hay không. Những kỹ thuật phổ biến nhất liên quan đến mô hình không gian-véc-tơ tài liệu sẽ được mô tả trong phần này.

Hàng xóm Gần Nhất (Nearest Neighbors)

Một phương pháp đơn giản để ước lượng mức độ quan tâm của người dùng đối với một tài liệu nhất định là kiểm tra xem liệu người dùng đã thích những tài liệu tương tự trong quá khứ hay không. Để làm điều này, cần hai loại thông tin: lịch sử đánh giá của người dùng về các sản phẩm trước đó và một phép đo để xác định sự tương đồng giữa các tài liệu.

Phản hồi sự liên quan: Phương pháp Rocchio

Giới thiệu

Phản hồi sự liên quan (Relevance Feedback) là một kỹ thuật trong hệ thống truy xuất thông tin (Information Retrieval - IR) nhằm cải thiện chất lượng truy vấn ban đầu của người dùng. Ý tưởng chính là cho phép người dùng cung cấp phản hồi về các tài liệu (document) có liên quan hoặc không liên quan, sau đó hệ thống điều chỉnh lại vector truy vấn để trả về kết quả chính xác hơn trong các lần truy vấn tiếp theo.

Fương pháp Rocchio [22] là một trong những kỹ thuật cổ điển và nền tảng trong phản hồi sự liên quan, được áp dụng phổ biến trong mô hình không gian vector (Vector Space Model - VSM).

Nguyên lý: Trong VSM, cả truy vấn và tài liệu đều được biểu diễn dưới dạng vector trong không gian đặc trưng (term space). Truy vấn ban đầu thường ngắn gọn và không đủ thông tin. Rocchio đề xuất cách **tái tạo vector truy vấn** dựa trên sự kết hợp giữa:

- Vector truy vấn ban đầu.
- Các vector tài liệu mà người dùng đánh dấu là liên quan (relevant documents).
- Các vector tài liệu mà người dùng đánh dấu là không liên quan (non-relevant documents).

Công thức Rocchio: Vector truy vấn mới \vec{q}_{new} được tính như sau:

$$\vec{q}_{\text{new}} = \alpha \vec{q}_0 + \frac{\beta}{|D_r|} \sum_{\vec{d}_i \in D_r} \vec{d}_i - \frac{\gamma}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \quad (4.1)$$

Trong đó:

- \vec{q}_0 : vector truy vấn ban đầu.
- D_r : tập các tài liệu liên quan (relevant).
- D_{nr} : tập các tài liệu không liên quan (non-relevant).
- α, β, γ : tham số điều chỉnh trọng số.
- \vec{q}_{new} : vector truy vấn được điều chỉnh.

Ý nghĩa:

- Thành phần đầu tiên $\alpha\vec{q}_0$ giữ thông tin gốc của truy vấn.
- Thành phần thứ hai $\frac{\beta}{|D_r|} \sum \vec{d}_i$ dịch chuyển truy vấn gần hơn tới các tài liệu liên quan.
- Thành phần thứ ba $-\frac{\gamma}{|D_{nr}|} \sum \vec{d}_j$ đẩy truy vấn xa các tài liệu không liên quan.

Ví dụ minh họa: Giả sử người dùng tìm kiếm truy vấn “*machine learning*”.

Sau lượt tìm kiếm đầu tiên:

- Người dùng đánh dấu 2 tài liệu liên quan: d_1, d_2 .
- Người dùng đánh dấu 1 tài liệu không liên quan: d_3 .

Áp dụng công thức Rocchio, vector truy vấn được cập nhật gần hơn với nội dung của d_1, d_2 và xa hơn so với d_3 , nhờ vậy kết quả tìm kiếm các vòng sau sẽ tập trung hơn vào chủ đề *machine learning*.

Ưu điểm:

- Đơn giản, dễ triển khai trên mô hình không gian vector.
- Giúp cải thiện đáng kể hiệu quả truy xuất thông tin khi có phản hồi từ người dùng.
- Có thể kết hợp với nhiều mô hình học máy hiện đại.

Hạn chế:

- Phụ thuộc vào chất lượng phản hồi từ người dùng: nếu phản hồi sai, hệ thống có thể bị lệch hướng.

- Không phù hợp khi số lượng tài liệu liên quan được đánh dấu quá ít.
- Không xử lý được tốt các ngữ nghĩa phức tạp, chỉ dựa vào mô hình vector tuyến tính.

Ứng dụng:

- Hệ thống tìm kiếm học thuật (ví dụ: tìm bài báo khoa học).
- Tìm kiếm đa phương tiện (ảnh, video) dựa trên từ khóa.
- Các hệ thống gợi ý kết hợp với phản hồi tự động từ người dùng.

Các phương pháp xác suất (Probabilistic methods)

Các phương pháp phân loại xác suất, đặc biệt là phương pháp Bayes đơn giản, là các phương pháp phân loại nổi bật đã được phát triển trong các hệ thống phân loại văn bản từ đầu. Những phương pháp này dựa trên giả thuyết độc lập điều kiện (về sự xuất hiện của các từ) và đã được ứng dụng thành công trong các hệ thống gợi ý dựa trên nội dung.

Nhớ lại công thức cơ bản để tính toán xác suất hậu nghiệm cho phân loại tài liệu từ phần 2.4.3:

$$P(Y|X) = \prod_{i=1}^d P(X_i|Y) \times P(Y)/P(X)$$

Áp dụng trực tiếp mô hình này cho bài toán phân loại được mô tả bởi Pazzani và Billsus (1997). Các lớp có thể là "thích" và "không thích" (được gọi là nóng và lạnh trong một số bài báo). Các tài liệu được mô tả bằng các véc-tơ đặc trưng Boolean mô tả liệu một từ có xuất hiện trong tài liệu hay không; các véc-tơ đặc trưng này được giới hạn trong 128 từ có tính thông tin nhất, như đã đề cập trước đó.

Các bộ phân loại tuyến tính và học máy

Khi xem bài toán gợi ý dựa trên nội dung như một bài toán phân loại, có thể sử dụng nhiều kỹ thuật học máy khác nhau. Ở mức độ trừu tượng hơn, hầu hết các phương pháp học đều nhắm đến việc tìm các hệ số của một mô hình tuyến tính để phân biệt giữa các tài liệu liên quan và không liên quan.

Trong không gian hai chiều, đường thẳng mà chúng ta tìm kiếm có dạng:

$$w_1x_1 + w_2x_2 = b$$

Trong đó x_1 và x_2 là véc-tơ đại diện của một tài liệu (sử dụng, ví dụ, trọng số TF-IDF), còn w_1, w_2 và b là các tham số cần học. Việc phân loại một tài liệu riêng biệt sẽ dựa vào việc kiểm tra điều kiện $w_1x_1 + w_2x_2 > b$, điều này có thể thực hiện rất hiệu quả.

Các mô hình quyết định rõ ràng (Explicit decision models)

Hai phương pháp học khác đã được sử dụng để xây dựng các hệ thống gợi ý dựa trên nội dung là học cây quyết định và rút ra quy tắc. Chúng khác biệt so với các phương pháp khác ở chỗ chúng tạo ra một mô hình quyết định rõ ràng trong quá trình huấn luyện.

Học cây quyết định dựa trên thuật toán ID3 hoặc các thuật toán C4.5 sau này (xem Quinlan 1993 để biết tổng quan) đã được áp dụng thành công cho nhiều bài toán thực tế, chẳng hạn như các bài toán khai thác dữ liệu. Khi áp dụng cho bài toán gợi ý, các nút bên trong cây được dán nhãn với các đặc điểm của sản phẩm (từ khóa), và các nút này được sử dụng để phân tách các ví dụ kiểm tra dựa trên, ví dụ, sự tồn tại hoặc không tồn tại của một từ khóa trong tài liệu.

4.3 Thảo luận (Discussion)

4.3.1 Đánh giá so sánh (Comparative evaluation)

Pazzani và Billsus [19] trình bày một so sánh giữa các kỹ thuật học khác nhau cho gợi ý dựa trên nội dung. Các thí nghiệm được thực hiện với một số bộ tài liệu nhỏ và được gán nhãn thủ công trong các lĩnh vực khác nhau. Các thí nghiệm được thực hiện với hệ thống Syskill & Webert được thiết lập theo cách mà một tập hợp các tài liệu được sử dụng để học hồ sơ người dùng, và sau đó sử dụng hồ sơ đó để dự đoán liệu người dùng có quan tâm đến các tài liệu chưa thấy.

Phần trăm tài liệu được phân loại đúng được sử dụng làm thước đo độ chính xác. Độ chính xác của các hệ thống gợi ý khác nhau thay đổi khá mạnh trong các thí nghiệm này (từ 60% đến 80%). Giống như hầu hết các thuật toán học, yếu tố quan trọng nhất là kích thước của bộ dữ liệu huấn luyện (lên đến 50 tài liệu trong các thử nghiệm này). Đối với một số bộ ví dụ, sự cải thiện là đáng kể và độ chính xác lên hơn 80% có thể đạt được. Tuy nhiên, trong một số lĩnh vực, bộ phân loại không bao giờ vượt quá mức độ chính xác ngẫu nhiên.

Nhìn chung, việc so sánh chi tiết các thuật toán (sử dụng 20 ví dụ huấn luyện trong mỗi phương pháp) không mang lại kết quả rõ ràng. Những gì có thể nhận thấy là các thuật toán học cây quyết định, mà chúng ta không đề cập chi tiết, không thực hiện tốt trong các cài đặt đã cho và phương pháp "hàng xóm gần nhất" (nearest neighbors) thực hiện kém trong một số lĩnh vực. Các phương pháp Bayes và Rocchio thực hiện ổn định tốt trong tất cả các lĩnh vực, và không thể tìm thấy sự khác biệt đáng kể. Trong các thí nghiệm, một phương pháp mạng nơ-ron với hàm kích hoạt phi tuyến cũng được đánh giá nhưng không mang lại sự cải thiện về độ chính xác phân loại.

4.3.2 Hạn chế (Limitations)

Các hệ thống gợi ý thuần túy dựa trên nội dung có những hạn chế đã biết, điều này đã dẫn đến sự phát triển của các hệ thống lai kết hợp các lợi thế của các kỹ thuật gợi ý khác nhau. Hệ thống Fab là một ví dụ sớm về một hệ thống lai như vậy; Balabanovic và Shoham [2] đã chỉ ra các hạn chế của các hệ thống gợi ý dựa trên nội dung.

- **Phân tích nội dung nông:** Đặc biệt khi các trang web là các sản phẩm cần gợi ý, việc đánh giá chất lượng hoặc sự thú vị của một trang web chỉ bằng cách nhìn vào nội dung văn bản có thể không đủ. Các yếu tố khác, như thẩm mỹ, tính dễ sử dụng, tính kịp thời, hoặc sự chính xác của các liên kết, cũng xác định chất lượng của một trang.
- **Siêu chuyên môn hóa:** Các phương pháp học nhanh chóng có xu hướng gợi ý những sản phẩm tương tự – tức là, các hệ thống này chỉ có thể gợi ý các sản phẩm giống như những sản phẩm mà người dùng đã đánh giá cao.

4.4 Tóm tắt (Summary)

Trong chương này, chúng tôi đã thảo luận các phương pháp khác nhau thường được gọi là kỹ thuật gợi ý dựa trên nội dung. Nguồn gốc của hầu hết các phương pháp có thể tìm thấy trong lĩnh vực truy hồi thông tin (IR), vì các nhiệm vụ IR điển hình như lọc thông tin hoặc phân loại văn bản có thể được xem như một dạng bài toán gợi ý. Các phương pháp được trình bày đều có điểm chung là chúng nhằm mục đích học mô hình sở thích của người dùng dựa trên phản hồi rõ ràng hoặc ngụ ý.

Tuy nhiên, vẫn còn những thử thách. Một thử thách đầu tiên là việc thu thập phản hồi của người dùng và những người dùng mới. Việc cung cấp phản hồi rõ ràng là một công việc nặng nhọc cho người dùng, và việc suy luận phản hồi ngầm từ hành vi của người dùng (chẳng hạn như xem chi tiết sản phẩm trong một khoảng thời gian nhất định) có thể gặp phải vấn đề.

Chương 5

GỢI Ý DỰA TRÊN TRI THỨC

5.1	Giới thiệu	87
5.2	Biểu diễn tri thức và suy luận	89
5.3	Ràng buộc	91
5.4	Trường hợp và độ tương đồng	93
5.5	Tương tác với hệ thống gợi ý dựa trên ràng buộc	94
5.6	Tương tác với hệ thống gợi ý dựa trên case	95

5.1 Giới thiệu

Phần lớn các hệ thống gợi ý thương mại hiện nay được xây dựng dựa trên kỹ thuật lọc cộng tác (Collaborative Filtering - CF). Các hệ thống CF chỉ dựa vào đánh giá của người dùng (và đôi khi thêm thông tin nhân khẩu học) như là nguồn tri thức duy nhất để sinh ra các gợi ý. Do đó, không cần phải nhập hay duy trì thêm bất kỳ tri thức bổ sung nào — chẳng hạn như thông tin về các mặt hàng săn có hoặc các đặc điểm của chúng.

Ngược lại, các kỹ thuật gợi ý dựa trên nội dung (Content-based) sử dụng những nguồn tri thức khác để dự đoán liệu người dùng có thích một sản phẩm hay không. Các nguồn tri thức chính được khai thác bao gồm: thông tin về thể loại, danh mục, cũng như các từ khóa có thể tự động trích xuất từ mô tả văn bản của sản phẩm. Tương tự như CF, một ưu điểm lớn của phương pháp gợi ý dựa trên nội dung là chi phí thu thập và duy trì tri thức tương đối thấp.

Cả hai phương pháp CF và Content-based đều có những ưu điểm và thế mạnh

riêng. Tuy nhiên, trong nhiều tình huống, chúng không phải là lựa chọn tối ưu. Ví dụ, việc mua nhà, xe hơi hoặc máy tính không diễn ra thường xuyên. Trong những kịch bản này, một hệ thống CF thuần túy sẽ hoạt động kém do số lượng đánh giá sẵn có rất ít. Hơn nữa, yếu tố thời gian cũng đóng vai trò quan trọng. Chẳng hạn, những đánh giá cách đây năm năm về một chiếc máy tính có thể không còn phù hợp để dùng trong gợi ý dựa trên nội dung. Điều tương tự cũng đúng với các sản phẩm như xe hơi hay nhà ở, khi sở thích người dùng thay đổi theo thời gian do thay đổi về lối sống hoặc tình trạng gia đình.

Bên cạnh đó, trong các lĩnh vực sản phẩm phức tạp như xe hơi, khách hàng thường muốn đưa ra các yêu cầu cụ thể — chẳng hạn như “giá tối đa của xe là x và màu sắc phải là màu đen.” Việc biểu đạt yêu cầu chi tiết như vậy không phổ biến trong các khung gợi ý thuần túy dựa trên cộng tác hoặc nội dung.

Các hệ gợi ý dựa trên tri thức (Knowledge-based recommender systems) giúp giải quyết những thách thức này. Ưu điểm nổi bật của chúng là không gặp phải vấn đề *khởi động lạnh* (ramp-up), vì không cần dữ liệu đánh giá để tính toán gợi ý. Các gợi ý được xác định độc lập với đánh giá của từng người dùng: hoặc dựa trên độ tương đồng giữa yêu cầu của khách hàng và các sản phẩm, hoặc dựa trên các luật gợi ý được xác định rõ ràng.

Cách hiểu truyền thống về hệ gợi ý tập trung vào khía cạnh lọc thông tin, trong đó hệ thống loại ra những sản phẩm ít có khả năng phù hợp với người dùng và chỉ giữ lại những sản phẩm tiềm năng. Ngược lại, các hệ gợi ý dựa trên tri thức được thiết kế mang tính *tương tác cao*, nên còn được gọi là các hệ thống hội thoại (conversational systems). Đặc tính tương tác này đã mở rộng phạm vi định nghĩa: hệ gợi ý không chỉ là bộ lọc, mà còn là hệ thống “dẫn dắt người dùng theo cách cá nhân hóa đến những sản phẩm hữu ích hoặc thú vị trong một không gian lựa chọn rất lớn, hoặc trực tiếp sinh ra các sản phẩm đó.”

Theo định nghĩa, những hệ gợi ý khai thác các nguồn tri thức mà CF hoặc Content-based không sử dụng mặc định được gọi là *hệ gợi ý dựa trên tri thức*. Có hai loại cơ bản:

- **Hệ gợi ý dựa trên ràng buộc (Constraint-based):** dựa trên một tập hợp luật gợi ý được định nghĩa rõ ràng. Tập sản phẩm được gợi ý được xác định bằng cách tìm kiếm những sản phẩm thỏa mãn các luật đó. Nếu không tìm thấy lời giải, người dùng cần điều chỉnh lại yêu cầu. Các hệ này cũng có khả năng cung cấp giải thích cho các gợi ý.
- **Hệ gợi ý dựa trên tình huống (Case-based):** tập trung vào việc truy

xuất những sản phẩm tương tự dựa trên các phép đo độ tương đồng. Hệ thống dùng các chỉ số độ giống nhau để tìm ra những sản phẩm gần nhất (trong một ngữ cảnh định sẵn) so với yêu cầu người dùng.

Cả hai phương pháp trên đều có quy trình gợi ý tương tự: người dùng đưa ra yêu cầu, và hệ thống tìm kiếm sản phẩm phù hợp. Điểm khác biệt nằm ở cách sử dụng tri thức — hệ dựa trên ràng buộc buộc luật rõ ràng, trong khi hệ dựa trên tình huống khai thác các thước đo độ tương đồng.

5.2 Biểu diễn tri thức và suy luận

Nhìn chung, các hệ thống gợi ý dựa trên tri thức dựa vào thông tin chi tiết về các đặc điểm của sản phẩm. Trong miền ứng dụng, vấn đề gợi ý có thể được mô tả đơn giản là: chọn ra những sản phẩm từ danh mục sao cho phù hợp với nhu cầu, sở thích hoặc các ràng buộc cứng của người dùng. Yêu cầu của người dùng có thể được biểu diễn bằng các giá trị cụ thể hoặc khoảng giá trị cho các thuộc tính, ví dụ: “giá phải thấp hơn 20 triệu đồng” hoặc bằng chức năng mong muốn, chẳng hạn: “máy tính xách tay phải phù hợp cho việc chơi game”.

5.2.1 Ví dụ minh họa: Laptop

```
[
  {
    "id": "l1",
    "Giá (triệu VND)": 15,
    "CPU": "i5-1135G7",
    "RAM": "8GB",
    "Ổ cứng": "SSD 256",
    "Card đồ họa": "tích hợp",
    "Màn hình": "14''",
    "Pin (giờ)": 6
  },
  {
    "id": "l2",
    "Giá (triệu VND)": 19,
    "CPU": "i7-1165G7",
    "RAM": "16GB",
    "Ổ cứng": "SSD 512",
  }
]
```

```
        "Card đồ họa": "rời (MX450)",
        "Màn hình": "15.6''",
        "Pin (giờ)": 7
    },
    {
        "id": "13",
        "Giá (triệu VND)": 22,
        "CPU": "R7-5800H",
        "RAM": "16GB",
        "Ổ cứng": "SSD 512",
        "Card đồ họa": "rời (RTX3050)",
        "Màn hình": "15.6''",
        "Pin (giờ)": 5
    },
    {
        "id": "14",
        "Giá (triệu VND)": 13,
        "CPU": "i3-1115G4",
        "RAM": "8GB",
        "Ổ cứng": "SSD 256",
        "Card đồ họa": "tích hợp",
        "Màn hình": "14''",
        "Pin (giờ)": 8
    },
    {
        "id": "15",
        "Giá (triệu VND)": 28,
        "CPU": "i9-12900H",
        "RAM": "32GB",
        "Ổ cứng": "SSD 1TB",
        "Card đồ họa": "rời (RTX3060)",
        "Màn hình": "16''",
        "Pin (giờ)": 6
    }
]
```

5.2.2 Giải thích ví dụ

Hệ thống gợi ý dựa trên ràng buộc (Constraint-based): Nếu người dùng đưa ra các ràng buộc:

- Giá dưới 20 triệu
- RAM ít nhất 16GB
- Có card đồ họa rời

thì hệ thống sẽ lọc danh mục, và kết quả trả về là **laptop l2** (19 triệu, i7, 16GB RAM, MX450) vì đây là sản phẩm duy nhất thỏa cả ba ràng buộc.

Hệ thống gợi ý dựa trên tình huống (Case-based): Nếu người dùng đưa ra một trường hợp mẫu như “tôi muốn một chiếc laptop cho gaming tầm trung”, hệ thống sẽ tính độ tương đồng giữa yêu cầu này với các sản phẩm dựa trên các chỉ số CPU, card đồ họa, RAM, và giá. Trong trường hợp này, **laptop l3** (22 triệu, Ryzen 7, RTX3050) có độ tương đồng cao nhất vì cấu hình hướng đến game thủ, mặc dù giá hơi cao hơn ràng buộc ngầm định.

Nhận xét: Cùng một danh mục sản phẩm, tùy theo cách biểu diễn tri thức (ràng buộc hay tình huống), hệ thống sẽ có cơ chế suy luận khác nhau để đưa ra kết quả gợi ý phù hợp.

5.3 Ràng buộc

Một bài toán ràng buộc cổ điển (Constraint Satisfaction Problem - CSP) có thể được mô tả bởi bộ ba (V, D, C) , trong đó:

- V là tập hợp các biến,
- D là tập hợp các miền giá trị hữu hạn của các biến này,
- C là tập hợp các ràng buộc mô tả các tổ hợp giá trị mà các biến có thể nhận đồng thời.

Một nghiệm của CSP tương ứng với việc gán một giá trị cho mỗi biến trong V sao cho tất cả các ràng buộc đều được thỏa mãn.

Bảng 5.1: Nhiệm vụ gợi ý: biến người dùng, biến sản phẩm, ràng buộc và kết quả

VC	{max-price(0...1000), usage(work, gaming, multimedia), screen-size(small, medium, large)}
VPROD	{price(0...1000), CPU(i3,i5,i7,i9), RAM(8,16,32), GPU(integrated, MX450, RTX3050, RTX3060), screen-size(13,14,15,16)}
CF	{usage = gaming → GPU ≠ integrated}
CR	{usage = gaming → max-price > 15}
CPROD	{(id=l1 ∧ price=15 ∧ CPU=i5 ∧ RAM=8 ∧ GPU=integrated ∧ screen=14) ∨ ... ∨ (id=l5 ∧ price=28 ∧ CPU=i9 ∧ RAM=32 ∧ GPU=RTX3060 ∧ screen=16)}
REQ	{max-price=20, usage=gaming, screen-size=15}
RES	{id=l2, price=19, CPU=i7, RAM=16, GPU=MX450, screen=15}

- **Biến người dùng (VC):** mô tả các yêu cầu tiềm năng của khách hàng, ví dụ max-price là giá tối đa, usage là mục đích sử dụng laptop (work/gaming/-multimedia), screen-size là kích thước màn hình mong muốn.
- **Biến sản phẩm (VPROD):** mô tả các thuộc tính của sản phẩm, ví dụ CPU, RAM, GPU, kích thước màn hình, giá.
- **Ràng buộc tương thích (CR):** giới hạn các giá trị hợp lệ cho biến người dùng, ví dụ nếu mục đích là gaming thì giá tối thiểu phải > 15 triệu.
- **Điều kiện lọc (CF):** xác định điều kiện chọn sản phẩm dựa trên các thuộc tính người dùng, ví dụ người dùng cần gaming thì GPU không được tích hợp.
- **Ràng buộc sản phẩm (CPROD):** xác định các sản phẩm hiện có, mỗi tổ hợp trong ràng buộc này mô tả đầy đủ một sản phẩm.
- **Yêu cầu người dùng (REQ):** được mã hóa như ràng buộc đơn trên các biến trong VC và VPROD, ví dụ max-price=20, usage=gaming.

Nghiệm của CSP ($V = VC \cup VPROD, D, C = CR \cup CF \cup CPROD \cup REQ$) tương ứng với một gợi ý nhất quán. Ví dụ, hệ thống sẽ trả về **laptop l2** với giá 19 triệu, CPU i7, RAM 16GB, GPU MX450, màn hình 15", đáp ứng tất cả các ràng buộc và yêu cầu của người dùng.

Truy vấn liên kết (Conjunctive queries)

Một cách khác để truy xuất sản phẩm từ danh mục là coi bài toán như một nhiệm vụ lọc dữ liệu. Trong trường hợp này, không cần tìm nghiệm CSP mà chỉ cần

xây dựng truy vấn cơ sở dữ liệu liên kết (conjunctive query). Ví dụ:

$$\sigma[\text{GPU} \neq \text{integrated}, \text{price} \leq 20](L)$$

trong đó L là bảng sản phẩm, σ là toán tử chọn, và điều kiện trong ngoặc vuông là điều kiện lọc. Kết quả của truy vấn này là tập hợp các sản phẩm thỏa yêu cầu, ví dụ $\{l_2, l_3\}$ nhưng l_3 vượt quá giá tối đa 20 triệu nên chỉ còn l_2 .

5.4 Trưởng hợp và độ tương đồng

Trong các phương pháp gợi ý dựa trên trưởng hợp (case-based recommendation), các sản phẩm được truy xuất dựa trên các chỉ số tương đồng, mô tả mức độ mà thuộc tính sản phẩm phù hợp với các yêu cầu của người dùng. Chỉ số tương đồng khoảng cách (distance similarity) của một sản phẩm p với yêu cầu $r \in REQ$ thường được định nghĩa như sau:

$$\text{similarity}(p, REQ) = \frac{\sum_{r \in REQ} w_r \cdot \text{sim}(p, r)}{\sum_{r \in REQ} w_r} \quad (5.1)$$

Trong đó:

- $\text{sim}(p, r)$ biểu thị khoảng cách giữa giá trị thuộc tính $\phi_r(p)$ của sản phẩm p với yêu cầu của khách hàng $r \in REQ$. Ví dụ: $\phi_{mpix}(p_1) = 8.0$.
- w_r là trọng số quan trọng của yêu cầu r .

Trong thực tế, có những thuộc tính mà khách hàng muốn tối đa hóa (ví dụ: độ phân giải của máy ảnh), gọi là các thuộc tính *more-is-better* (MIB), và cũng có những thuộc tính mà khách hàng muốn tối thiểu hóa (ví dụ: giá của sản phẩm hoặc mức độ rủi ro của dịch vụ tài chính), gọi là các thuộc tính *less-is-better* (LIB).

5.4.1 Tính toán độ tương đồng cục bộ

Trưởng hợp thuộc tính MIB:

$$\text{sim}(p, r) = \frac{\phi_r(p) - \min(r)}{\max(r) - \min(r)} \quad (5.2)$$

Trưởng hợp thuộc tính LIB:

$$\text{sim}(p, r) = \frac{\max(r) - \phi_r(p)}{\max(r) - \min(r)} \quad (5.3)$$

Trường hợp dựa trên khoảng cách tuyệt đối với yêu cầu định nghĩa sẵn:

$$\text{sim}(p, r) = 1 - \frac{|\phi_r(p) - r|}{\max(r) - \min(r)} \quad (5.4)$$

Các chỉ số tương đồng trên thường là cơ sở cho các hệ thống gợi ý dựa trên trường hợp (case-based recommendation) khác nhau. Các phương pháp gợi ý dựa trên tiện ích (utility-based recommendation) có thể coi là một dạng cụ thể của gợi ý dựa trên tri thức, nhưng thường được áp dụng kết hợp với gợi ý dựa trên ràng buộc hoặc đôi khi kết hợp với gợi ý dựa trên trường hợp.

5.5 Tương tác với hệ thống gợi ý dựa trên ràng buộc

Hệ thống gợi ý dựa trên ràng buộc (constraint-based recommenders) cho phép người dùng định nghĩa các yêu cầu và sở thích để nhận được đề xuất phù hợp. Tương tác giữa người dùng và hệ thống thường diễn ra theo hai cách:

- **Xác định yêu cầu ban đầu một lần:** Người dùng nhập tất cả các yêu cầu từ đầu. Ví dụ, khi tìm laptop, người dùng có thể nhập giá tối đa, dung lượng RAM, kích thước màn hình và mục đích sử dụng. Hệ thống trả về danh sách các sản phẩm đáp ứng các ràng buộc này.
- **Tương tác theo kiểu hướng dẫn (wizard-style):** Người dùng nhập yêu cầu từng bước, được hệ thống hướng dẫn thông qua hộp thoại tương tác. Ví dụ, hệ thống hỏi trước về mục đích sử dụng, sau đó hỏi ngân sách và các yêu cầu chi tiết khác.

Sau khi tính toán, người dùng có thể:

- Xem danh sách sản phẩm phù hợp kèm giải thích vì sao mỗi sản phẩm được đề xuất.
- Điều chỉnh yêu cầu nếu chưa hài lòng.
- Xem các giải pháp thay thế hoặc danh sách sản phẩm khác đáp ứng một phần yêu cầu.
- Thu hẹp số lượng sản phẩm bằng các bộ lọc bổ sung hoặc điều chỉnh yêu cầu.

5.5.1 Thiết lập giá trị mặc định (Defaults)

Dể hỗ trợ người dùng không chắc chắn về các yêu cầu kỹ thuật, hệ thống có thể cung cấp các giá trị mặc định nhằm giúp họ chọn lựa hợp lý.

Các loại giá trị mặc định:

- **Static defaults (tĩnh):** Giá trị cố định, không thay đổi theo ngữ cảnh (ví dụ: RAM mặc định 8GB, ổ cứng 512GB).
- **Dependent defaults (phụ thuộc):** Giá trị mặc định thay đổi dựa trên lựa chọn trước đó (ví dụ: nếu chọn laptop chơi game, RAM mặc định là 16GB).
- **Derived defaults (suy ra):** Giá trị được tính toán dựa trên dữ liệu người dùng hoặc hành vi của người dùng tương tự.

Hệ thống cũng xác định **câu hỏi tiếp theo** dựa trên các thuộc tính quan trọng, giảm bớt số thông tin mà người dùng cần nhập.

5.5.2 Xử lý yêu cầu không thỏa mãn

Khi không có sản phẩm nào đáp ứng đầy đủ yêu cầu, hệ thống có thể áp dụng:

- **Giảm ràng buộc (Constraint Relaxation):** Nới lỏng một số ràng buộc để tìm được sản phẩm phù hợp. Ví dụ, nếu yêu cầu RAM $\geq 32\text{GB}$ và giá ≤ 1000 USD không có sản phẩm nào đáp ứng, hệ thống có thể đề xuất RAM $\geq 16\text{GB}$.
- **Đề xuất sửa lỗi (Repair Proposals):** Hệ thống tính toán giải pháp bằng cách điều chỉnh các yêu cầu để đạt được sản phẩm khả thi. Ví dụ: đề xuất tăng ngân sách hoặc giảm một yêu cầu không bắt buộc.

Các phương pháp này giúp người dùng nhận được đề xuất khả thi mà vẫn giữ các yêu cầu quan trọng nhất.

5.6 Tương tác với hệ thống gợi ý dựa trên case

Tương tự như các hệ thống gợi ý dựa trên ràng buộc, các phiên bản đầu tiên của hệ thống gợi ý dựa trên case (case-based recommenders) sử dụng phương pháp hoàn toàn dựa trên truy vấn (query-based), trong đó người dùng phải xác định (và đổi khi xác định lại) các yêu cầu của mình cho đến khi tìm được sản phẩm mục tiêu (item phù hợp với mong muốn và nhu cầu của người dùng). Dối với những người

không chuyên trong lĩnh vực sản phẩm, quá trình yêu cầu này có thể trở nên nhảm chán vì các thuộc tính liên quan đến nhau của sản phẩm đòi hỏi kiến thức chuyên sâu.

Hạn chế này đã thúc đẩy phát triển các phương pháp duyệt (browsing-based) trong việc tìm kiếm sản phẩm, nơi người dùng – có thể chưa biết rõ họ đang tìm gì – điều hướng trong không gian sản phẩm nhằm tìm các lựa chọn hữu ích. Trong bối cảnh này, **phản hồi chỉnh sửa (critiquing)** là một phương pháp hiệu quả để hỗ trợ điều hướng và cũng là một trong những khái niệm chính của gợi ý dựa trên case.

5.6.1 Phản hồi chỉnh sửa (Critiquing)

Ý tưởng của phản hồi chỉnh sửa là người dùng chỉ ra những thay đổi mong muốn dưới dạng các mục tiêu chưa được thỏa mãn bởi sản phẩm hiện tại (entry item hoặc recommended item). Ví dụ:

- Nếu giá của máy ảnh hiển thị hiện tại quá cao, người dùng có thể chọn critique *cheaper*.
- Nếu người dùng muốn máy ảnh có độ phân giải cao hơn, họ có thể chọn critique *more mpix*.
- Ví dụ khác: “Vị trí khách sạn nên gần biển hơn” hoặc “Căn hộ nên hiện đại hơn”.

Các critiques có thể được áp dụng cho cả các thuộc tính kỹ thuật và các đặc điểm trừu tượng.

Các hệ thống case-based hiện đại kết hợp phương pháp dựa trên truy vấn với phương pháp duyệt. Phản hồi chỉnh sửa giúp người dùng điều hướng hiệu quả trong không gian sản phẩm, trong khi truy xuất case dựa trên độ tương đồng hỗ trợ tìm các sản phẩm tương tự với sản phẩm hiện đang được xem xét. Phản hồi chỉnh sửa giúp người dùng dễ dàng thể hiện sở thích mà không cần chỉ định giá trị cụ thể cho từng thuộc tính của sản phẩm, đồng thời tiết kiệm thời gian trong quá trình lựa chọn sản phẩm mà vẫn đảm bảo chất lượng gợi ý tương đương với phương pháp truy vấn truyền thống.

5.6.2 Các bước chính của hệ thống critiquing

1. Đề xuất sản phẩm (Item recommendation):

- Đầu vào là truy vấn ban đầu của người dùng q và tập hợp các sản phẩm ứng cử CI (bao gồm tất cả sản phẩm hiện có).
- Thuật toán đầu tiên chọn một sản phẩm r để hiển thị cho người dùng, được gọi là *entry item*. Các sản phẩm hiển thị sau đó gọi là *recommended items*.
- Trong vòng critiquing đầu tiên, việc truy xuất dựa trên truy vấn q của người dùng. Entry items thường được chọn bằng cách tính toán độ tương đồng giữa yêu cầu và các sản phẩm ứng cử.
- Trong các vòng critiquing tiếp theo, recommended items được chọn dựa trên độ tương đồng với sản phẩm hiện tại và các tiêu chí của critique do người dùng xác định.

2. Xem xét sản phẩm (Item reviewing):

- Người dùng xem xét sản phẩm được gợi ý và chấp nhận hoặc chọn critique khác, kích hoạt vòng critiquing mới.
- Khi một critique được kích hoạt, chỉ những sản phẩm ứng cử thỏa mãn tiêu chí của critique mới được xem xét tiếp, giảm tập CI .
- Ví dụ: nếu người dùng chọn critique *cheaper* và giá máy ảnh hiện tại là 300, trong vòng critiquing tiếp theo, hệ thống loại bỏ tất cả máy ảnh có giá ≥ 300 .

5.6.3 Phản hồi chỉnh sửa hợp chất (Compound critiquing)

Cho đến nay, chúng ta chủ yếu xem xét các **unit critiques**, cho phép người dùng chỉ định các thay đổi liên quan đến một thuộc tính duy nhất của sản phẩm. Mặc dù đơn giản, unit critiques có khả năng hạn chế trong việc hẹp không gian tìm kiếm. Ví dụ, một unit critique trên giá chỉ loại bỏ khoảng một nửa số sản phẩm.

Compound critiques được định nghĩa là các critiques hoạt động trên nhiều thuộc tính cùng lúc, giúp tăng hiệu quả của các phiên gợi ý, giảm số vòng critiquing cần thiết. Ví dụ, compound critique *cheaper & more mpix* đặt mục tiêu cho cả hai thuộc tính giá và độ phân giải, giúp loại bỏ nhiều sản phẩm không phù hợp trong một vòng critiquing.

Một ưu điểm quan trọng của compound critiques là cho phép người dùng tiến nhanh hơn trong không gian sản phẩm. Tuy nhiên, nếu được xây dựng tinh, compound critiques vẫn có nhược điểm: tất cả các lựa chọn critique luôn được hiển thị cho mỗi sản phẩm, dẫn đến các đề xuất không hợp lý trong những trường hợp đặc

thù (ví dụ, máy tính cao cấp với CPU nhanh nhất và dung lượng lưu trữ tối đa, một critique *faster CPU & more storage* sẽ vẫn được đề xuất).

5.6.4 Phản hồi chỉnh sửa động (Dynamic critiquing)

Dynamic critiquing khắc phục hạn chế trên bằng cách tạo critiques dựa trên các **mẫu (patterns)** mô tả sự khác biệt giữa sản phẩm được gợi ý (entry item) và các sản phẩm ứng cử (candidate items). Các critiques này được sinh ra *theo thời gian thực* trong mỗi vòng critiquing.

Các bước cơ bản trong dynamic critiquing gồm:

1. Đề xuất sản phẩm (Item recommendation):

- Thuật toán DynamicCritiquing nhận đầu vào: truy vấn ban đầu q , tập sản phẩm ứng cử CI , số lượng tối đa critiques hợp chất k , và giá trị hỗ trợ tối thiểu σ_{min} cho các quy tắc kết hợp.
- Chọn sản phẩm r để hiển thị cho người dùng (entry item ở vòng đầu tiên) bằng procedure *ItemRecommend*.
- Dựa trên r , thuật toán tính toán các compound critiques $cci \in CC$ thông qua các procedure:
 - *CritiquePatterns*: xác định các mẫu critique dựa trên so sánh thuộc tính giữa entry item và candidate items.
 - *APriori*: khai thác các critiques hợp chất từ các critique patterns bằng thuật toán Apriori.
 - *SelectCritiques*: xếp hạng các compound critiques và chọn các critiques quan trọng nhất để hiển thị.

2. Xác định mẫu critique (Identification of critique patterns):

- Mẫu critique đại diện cho sự khác biệt tổng quát giữa entry item và các candidate items.
- Ví dụ: so với entry item ei_8 , item ci_1 rẻ hơn, có độ phân giải thấp hơn, zoom thấp hơn, kích thước LCD nhỏ hơn, và không có chức năng quay phim. Mẫu critique sẽ là ($<, <, <, <, =$).

3. Khai thác các compound critiques từ critique patterns:

- Mục tiêu: xác định các critiques hợp chất xuất hiện thường xuyên trong các mẫu critique, dựa trên giả thuyết người dùng muốn thay đổi các thuộc tính theo đúng tổ hợp được đề xuất.

- Thuật toán Apriori được sử dụng để sinh các quy tắc kết hợp $p \Rightarrow q$, ví dụ: $> zoom \Rightarrow < price$, nghĩa là nếu tăng zoom thì giảm giá.
- Mỗi quy tắc kết hợp được đánh giá bằng **support** (tỷ lệ phần trăm các critique patterns chứa cả antecedent và consequent) và **confidence** (xác suất quy tắc đúng dựa trên dữ liệu).

Ưu điểm của dynamic critiquing:

- Sinh critiques hợp chất *theo ngữ cảnh*, phù hợp với entry item hiện tại.
- Giúp giảm số vòng critiquing, tiết kiệm thời gian cho người dùng.
- Cho phép hệ thống gợi ý linh hoạt, tránh đề xuất không hợp lý từ critiques tĩnh.

Chương 6

GỢI Ý DỰA TRÊN PHIÊN

6.1	Giới thiệu về gợi ý dựa trên phiên (Session-based Recommendation)	100
6.2	Gợi ý dựa trên độ tương đồng item-to-item	101
6.3	Phát hiện các luật kết hợp (Association Rule Mining) .	104
6.4	Tiếp cận dựa trên RNN: mô tả chi tiết và các mở rộng	107
6.5	BERT4Rec: Bidirectional Encoder Representations for Sequential Recommendation	114

6.1 Giới thiệu về gợi ý dựa trên phiên (Session-based Recommendation)

Trong nhiều hệ thống gợi ý, việc định danh rõ ràng người dùng là điều kiện quan trọng để khai thác lịch sử tương tác dài hạn và xây dựng hồ sơ cá nhân hóa. Tuy nhiên, trên thực tế, có nhiều tình huống mà hệ thống không thể nhận diện chính xác danh tính của người dùng, ví dụ như trong các ứng dụng thương mại điện tử không yêu cầu đăng nhập, hoặc các dịch vụ trực tuyến mà người dùng có xu hướng truy cập nhanh, không để lại nhiều thông tin. Trong các trường hợp này, toàn bộ dữ liệu có thể khai thác được chỉ giới hạn trong một **phiên làm việc (session)** – tức chuỗi các hành động hoặc tương tác của người dùng diễn ra liên tiếp trong một khoảng thời gian nhất định.

Gợi ý dựa trên phiên (session-based recommendation) ra đời để giải quyết bài toán này. Thay vì dựa vào hồ sơ dài hạn của người dùng, hệ thống tập

trung vào sở thích ngắn hạn, được thể hiện thông qua chuỗi hành vi trong một phiên duy nhất. Điều này đặc biệt hữu ích trong các bối cảnh mà sự quan tâm của người dùng có tính tức thời và thay đổi nhanh, chẳng hạn như:

- Người dùng duyệt sản phẩm trong một cửa hàng trực tuyến nhưng không đăng nhập.
- Người dùng nghe nhạc hoặc xem video trên nền tảng trực tuyến mà không có lịch sử lưu trữ.
- Các kịch bản mà sự quan tâm chỉ tồn tại trong một thời điểm cụ thể, như tìm kiếm vé máy bay hoặc đặt phòng khách sạn.

Để xây dựng hệ gợi ý dựa trên phiên, nhiều phương pháp đã được phát triển:

1. **Dựa trên độ tương đồng item-to-item:** tính toán mức độ giống nhau giữa các sản phẩm và đề xuất những item tương tự với các mục đã được người dùng tương tác trong phiên.
2. **Dựa trên luật kết hợp (association rules):** khai thác các mẫu đồng xuất hiện thường xuyên giữa các sản phẩm để dự đoán mục tiếp theo.
3. **Dựa trên mô hình học sâu cho dữ liệu chuỗi:** các mạng RNN, LSTM, Transformer4Rec, BERT4Rec được áp dụng nhằm mô hình hóa quan hệ thứ tự trong chuỗi hành vi, từ đó nắm bắt được ngữ cảnh và xu hướng ngắn hạn của người dùng.

Như vậy, gợi ý dựa trên phiên là một hướng tiếp cận quan trọng trong hệ gợi ý hiện đại, cho phép hệ thống cung cấp trải nghiệm cá nhân hóa ngay cả khi không có dữ liệu định danh người dùng, đồng thời đáp ứng tốt nhu cầu gợi ý theo ngữ cảnh và sở thích tức thời.

6.2 Gợi ý dựa trên độ tương đồng item-to-item

6.2.1 Giới thiệu

Trong lĩnh vực hệ gợi ý, phương pháp dựa trên độ tương đồng *item-to-item similarity* là một trong những kỹ thuật nền tảng, được ứng dụng rộng rãi nhờ tính đơn giản, khả năng mở rộng và hiệu quả trong thực tiễn. Khác với các phương pháp dựa trên hồ sơ người dùng, kỹ thuật này tập trung vào việc khai thác mối quan hệ giữa các sản phẩm (items) để đưa ra dự đoán, thay vì xây dựng mô hình cá nhân

hóa dài hạn. Ý tưởng cốt lõi là: nếu hai sản phẩm thường xuyên được tiêu thụ, xem xét hoặc tương tác cùng nhau trong cùng một ngữ cảnh, thì chúng có khả năng cao là liên quan và có thể thay thế hoặc bổ sung cho nhau.

6.2.2 Biểu diễn item

Để tính toán được độ tương đồng, trước tiên mỗi item cần được biểu diễn trong một không gian đặc trưng. Một số cách tiếp cận phổ biến:

- Biểu diễn dựa trên nội dung (*content-based features*), chẳng hạn như thể loại, màu sắc, mô tả sản phẩm hoặc embedding từ mô hình học sâu.
- Biểu diễn dựa trên hành vi, trong đó mỗi item được mã hóa dựa trên dữ liệu tương tác của người dùng (ví dụ: lịch sử mua hàng hoặc lượt xem).

6.2.3 Độ đo tương đồng

Có nhiều thước đo tương đồng được áp dụng để đánh giá mức độ liên hệ giữa hai item:

- **Độ tương đồng cosin (cosine similarity):**

$$\text{sim}(u, v) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|}$$

trong đó \vec{u} và \vec{v} là vector biểu diễn hai item.

- **Độ đo dựa trên xác suất có điều kiện:**

$$P_{uv} = \frac{\text{freq}(uv)}{\text{freq}(v)}$$

với:

- freq(uv) là số người dùng đã tương tác với v và sau đó tương tác với u ,
- freq(v) là số người dùng đã từng tương tác với v .

Để khắc phục hiện tượng các item phổ biến chiếm ưu thế, một công thức hiệu chỉnh được sử dụng:

$$\text{Sim}(u, v) = \frac{\text{freq}(uv)}{\text{freq}(u)^\alpha \cdot \text{freq}(v)}$$

trong đó α là hệ số điều chỉnh, giúp giảm thiểu ảnh hưởng của các item phổ biến.

6.2.4 Quy trình gợi ý

Phương pháp item-to-item similarity hoạt động theo các bước sau:

1. Với mỗi item trong tập tương tác của người dùng, tìm ra top- K item có độ tương đồng cao nhất.
2. Hợp các tập này lại để tạo thành tập ứng viên, đồng thời loại bỏ các phần tử trùng lặp.
3. Tính tổng điểm tương đồng của mỗi ứng viên so với toàn bộ item mà người dùng đã tương tác trong phiên.
4. Xếp hạng các ứng viên theo tổng điểm và chọn ra top- N để gợi ý.

6.2.5 Đánh giá

Phương pháp item-to-item mang lại nhiều ưu điểm đáng kể:

- Đơn giản và dễ triển khai.
- Hiệu quả trong môi trường dữ liệu lớn, nhờ khả năng tính toán phân tán.
- Không yêu cầu hồ sơ người dùng, phù hợp với bối cảnh session-based recommendation.

Tuy nhiên, phương pháp này cũng tồn tại hạn chế:

- Phụ thuộc mạnh vào chất lượng độ đo tương đồng.
- Khó khai thác được ngữ cảnh phức tạp và các quan hệ phi tuyến.
- Gặp vấn đề *cold-start* đối với các sản phẩm mới hoặc ít dữ liệu.

6.2.6 Ví dụ minh họa

Giả sử trong một phiên duyệt sản phẩm, người dùng đã tương tác với $\{\text{Áo sơ mi}, \text{Quần âu}\}$. Hệ thống tính toán độ tương đồng cho thấy:

- Áo sơ mi \leftrightarrow Cà vạt (sim = 0.75)
- Quần âu \leftrightarrow Giày da (sim = 0.68)

Sau khi hợp nhất tập ứng viên và xếp hạng theo điểm tổng, danh sách gợi ý được đưa ra là: $\{\text{Cà vạt}, \text{Giày da}\}$.

6.2.7 Kết luận

Phương pháp item-to-item similarity là một trụ cột quan trọng trong hệ gợi ý hiện đại, đặc biệt trong bối cảnh các hệ thống gợi ý dựa trên phiên. Với ưu điểm dễ áp dụng và khả năng mở rộng, phương pháp này đã được triển khai thành công trong nhiều hệ thống thương mại điện tử lớn. Tuy vậy, để cải thiện chất lượng gợi ý trong các tình huống phức tạp, các kỹ thuật học sâu và mô hình hóa ngữ cảnh hiện đang được nghiên cứu và phát triển như một sự mở rộng tự nhiên của phương pháp item-to-item truyền thống.

6.3 Phát hiện các luật kết hợp (Association Rule Mining)

6.3.1 Giới thiệu

Bài toán phát hiện các luật kết hợp (*association rule mining*) là một trong những kỹ thuật quan trọng trong khai phá dữ liệu và hệ gợi ý. Cho trước một tập các giao dịch (*transactions*), mục tiêu là tìm ra các luật dự đoán khả năng xuất hiện của một tập các mục (items) trong một giao dịch dựa trên sự xuất hiện của các mục khác.

Ví dụ, trong một hệ thống bán lẻ:

- $\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$
- $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\}$
- $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\}$

6.3.2 Các định nghĩa cơ bản

- **Tập mục (Itemset):** Một tập hợp gồm một hoặc nhiều mục. Ví dụ: $\{\text{Milk, Bread, Diaper}\}$.
- **k-itemset:** Tập mục gồm k mục.
- **Tổng số hỗ trợ (Support count) σ :** số lần xuất hiện của một tập mục trong cơ sở dữ liệu. Ví dụ: $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$.
- **Độ hỗ trợ (Support) s :** tỷ lệ các giao dịch chứa một tập mục. Ví dụ: $s(\{\text{Milk, Bread, Diaper}\}) = \frac{2}{5}$.
- **Tập mục thường xuyên (Frequent/Large itemset):** tập mục có độ hỗ trợ $\geq \text{minsup}$.

- **Luật kết hợp (Association rule):** Một biểu thức có dạng $X \rightarrow Y$, trong đó X và Y là các tập mục, $X \cap Y = \emptyset$.
- **Độ tin cậy (Confidence) c :** tỷ lệ số giao dịch chứa cả X và Y so với số giao dịch chứa X .

$$\text{conf}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

6.3.3 Bài toán phát hiện luật kết hợp

Với một tập các giao dịch T , mục đích là tìm tất cả các luật $X \rightarrow Y$ thỏa mãn:

- $\text{support}(X \cup Y) \geq \text{minsup}$,
- $\text{confidence}(X \rightarrow Y) \geq \text{minconf}$.

Phương pháp vét cạn (*brute-force*) liệt kê tất cả các luật có thể, tính toán độ hỗ trợ và độ tin cậy, sau đó loại bỏ các luật không đạt ngưỡng. Tuy nhiên, do số lượng luật có thể sinh ra là cực kỳ lớn, chi phí tính toán là không khả thi trong thực tế.

6.3.4 Nguyên lý và giải thuật Apriori

Để giảm độ phức tạp, giải thuật **Apriori** được sử dụng với nguyên tắc cắt tỉa dựa trên độ hỗ trợ:

- Nếu một tập mục là **thường xuyên**, thì tất cả các tập con của nó cũng thường xuyên.
- Nếu một tập mục là **không thường xuyên**, thì tất cả các tập cha của nó cũng không thường xuyên.

Điều này dựa trên đặc tính *anti-monotone* của độ hỗ trợ:

$$\text{support}(X) \geq \text{support}(Y), \quad \forall Y \subseteq X$$

Các bước của Apriori

1. Sinh tất cả các tập mục thường xuyên mức 1 (*frequent 1-itemsets*).
2. Gán $k = 1$.
3. Lặp lại cho đến khi không còn tập mục thường xuyên mới:

- Từ các tập mục thường xuyên mức k , sinh ra tập ứng viên mức $(k + 1)$.
- Loại bỏ các tập mục ứng viên có chứa tập con không thường xuyên.
- Tính độ hỗ trợ của các ứng viên, loại bỏ tập không đạt $minsup$.
- Nhận được tập mục thường xuyên mức $(k + 1)$.

6.3.5 Sinh ra luật kết hợp từ tập mục thường xuyên

Khi đã có tập mục thường xuyên L , các luật được sinh ra từ L bằng cách xét mọi tập con khác rỗng $f \subset L$ và tạo luật:

$$f \rightarrow (L \setminus f)$$

Nếu $|L| = k$, thì số lượng luật tiềm năng là $2^k - 2$.

Ví dụ với $L = \{Milk, Diaper, Beer\}$, ta có thể sinh ra:

- $\{Milk, Diaper\} \rightarrow \{Beer\}$ (s=0.4, c=0.67)
- $\{Milk, Beer\} \rightarrow \{Diaper\}$ (s=0.4, c=1.0)
- $\{Diaper, Beer\} \rightarrow \{Milk\}$ (s=0.4, c=0.67)
- $\{Beer\} \rightarrow \{Milk, Diaper\}$ (s=0.4, c=0.67)

6.3.6 Sinh luật hiệu quả

Dộ tin cậy không có đặc tính đơn điệu tuyệt đối. Tuy nhiên, các luật sinh ra từ cùng một tập mục thường xuyên có thể được so sánh theo đặc tính bán đơn điệu.

Ví dụ:

$$conf(ABC \rightarrow D) \geq conf(AB \rightarrow CD) \geq conf(A \rightarrow BCD)$$

Điều này cho phép cắt tỉa thêm trong quá trình sinh luật. Kỹ thuật kết hợp luật con với cùng tiền tố để sinh luật mới, đồng thời loại bỏ luật nếu một trong các luật con của nó không đạt ngưỡng $minconf$.

6.3.7 Kết luận

Phát hiện luật kết hợp là phương pháp hiệu quả trong việc tìm ra mối liên hệ tiềm ẩn giữa các item, có ứng dụng rộng rãi trong hệ gợi ý sản phẩm, phân tích giỏ hàng, và tiếp thị chéo. Giải thuật Apriori là phương pháp kinh điển, cho phép giảm mạnh chi phí tính toán so với vét cạn, tuy nhiên với dữ liệu quy mô lớn, các biến thể như FP-Growth được sử dụng để tăng tốc quá trình khai phá.

6.4 Tiếp cận dựa trên RNN: mô tả chi tiết và các mở rộng

6.4.1 Tổng quan

Recurrent Neural Networks (RNNs) là lớp mô hình học sâu được thiết kế để xử lý dữ liệu chuỗi nhờ khả năng lưu trữ và truyền thông tin trạng thái theo thời gian. Trong bối cảnh *session-based recommendation*, lịch sử tương tác trong một phiên được xem như một chuỗi i_1, i_2, \dots, i_T (mỗi i_t là biểu diễn của một item tại bước t). Mục tiêu là mô hình hóa phân phối điều kiện của item kế tiếp i_{t+1} dựa trên lịch sử $\{i_1, \dots, i_t\}$:

$$P(i_{t+1} | i_1, \dots, i_t).$$

RNN và các biến thể của nó (LSTM, GRU) cho phép nắm bắt thông tin tuần tự, tương tác ngắn hạn và một phần ngữ cảnh của phiên [10].

Kiến trúc RNN cơ bản

Một đơn vị RNN chuẩn tại thời điểm t có thể được biểu diễn bằng:

$$h_t = F(Wh_{t-1} +Ui_t + b_h), \quad o_t = G(Vh_t + b_o),$$

trong đó:

- $i_t \in \mathbb{R}^d$: embedding của item tại bước t ,
- $h_t \in \mathbb{R}^h$: trạng thái ẩn tại bước t ,
- $o_t \in \mathbb{R}^{|I|}$: logits/score cho toàn bộ tập item I (nếu dùng softmax),
- W, U, V và b_h, b_o là các tham số cần học,
- F thường là tanh hoặc ReLU, G thường là hàm tuyến tính trước softmax.

Hàm mất mát và chiến lược huấn luyện

Hai lớp phép toán mất mát phổ biến trong bài toán dự đoán item tiếp theo:

Softmax + Cross-entropy (pointwise / full softmax)

$$P(i | h_t) = \frac{\exp(o_{t,i})}{\sum_{k \in I} \exp(o_{t,k})}, \quad \mathcal{L}_t = -\log P(i^* | h_t),$$

Pairwise ranking (BPR)

$$\mathcal{L}_t^{\text{BPR}} = -\frac{1}{N_s} \sum_{j=1}^{N_s} \log \sigma(o_{t,i^*} - o_{t,j}),$$

TOP1 loss (GRU4Rec)

$$\mathcal{L}_t^{\text{TOP1}} = \frac{1}{N_s} \sum_{j=1}^{N_s} \left(\sigma(o_{t,j} - o_{t,i^*}) + \sigma(o_{t,j}^2) \right),$$

Kỹ thuật huấn luyện

- Negative sampling / sampled softmax
- Teacher forcing và scheduled sampling
- Session-parallel mini-batching
- Masking & padding

Vấn đề chuỗi dài và giải pháp: LSTM & GRU**Long Short-Term Memory (LSTM)**

$$\begin{aligned} i_t &= \sigma(W_i i_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f i_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o i_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \tanh(W_c i_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

Gated Recurrent Unit (GRU)

$$\begin{aligned} z_t &= \sigma(W_z i_t + U_z h_{t-1}) \\ r_t &= \sigma(W_r i_t + U_r h_{t-1}) \\ \tilde{h}_t &= \tanh(W_h i_t + U_h (r_t \odot h_{t-1})) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned}$$

Mảng rộng kiến trúc cho hệ gợi ý

- Stacked / deep RNNs
- Bidirectional RNN
- Hierarchical RNNs
- Session-parallel training (GRU4Rec)
- Two-stage architectures

Tối ưu hóa, điều chỉnh và regularization

- Embedding size, hidden size
- Dropout, weight decay, gradient clipping
- Optimizer: Adam, learning rate schedule
- Early stopping theo Recall@K, MRR@K

Suy luận (Inference) và tối ưu cho sản phẩm

- Cách suy luận: tính h_t , sinh scores o_t , chọn top- K
- Scaling: sampled softmax, ANN
- Latency: cache embedding, dot-product optimization
- Cập nhật online: mini-batch hoặc incremental

Ví dụ sơ lược về quy trình huấn luyện (pseudocode)

1. Chuẩn bị dữ liệu: sessions \rightarrow sequences of item IDs
2. Tạo embeddings cho items
3. Tổ chức session-parallel minibatches (GRU4Rec style)
4. Với mỗi epoch:
 - For each batch:
 - compute h_t via RNN/LSTM/GRU (teacher-forcing)
 - compute logits o_t for target items (sampled)
 - compute loss (BPR / cross-entropy / TOP1)
 - backpropagate, clip gradients, optimizer.step()

5. Validation theo Recall@K, MRR@K -> early stopping
6. Lưu model besten và inference

Kết luận

Tiếp cận dựa trên RNN (và các biến thể LSTM/GRU) cung cấp nền tảng vững chắc cho việc mô hình hóa hành vi tuần tự trong session-based recommendation. Khi được huấn luyện và triển khai đúng cách, RNN vẫn cạnh tranh tốt với các mô hình hiện đại, đặc biệt trong các hệ thống đòi hỏi latency thấp và bộ nhớ nhỏ.

6.4.2 Session-based Recommendations with Recurrent Neural Networks [10]

Giới thiệu

Session-based Recommendations with RNNs (GRU4Rec) [10] là một phương pháp tiên phong trong việc áp dụng **mạng nơ-ron hồi tiếp (Recurrent Neural Networks - RNN)** cho bài toán gợi ý theo phiên (session-based recommendation). Khác với hệ thống gợi ý truyền thống dựa trên lịch sử dài hạn của người dùng, session-based recommendation tập trung vào **chuỗi hành vi ngắn hạn** trong một phiên làm việc (session), chẳng hạn như chuỗi click hay lượt xem sản phẩm.

Ý tưởng chính

- Sử dụng mạng RNN (cụ thể là GRU) để mô hình hóa chuỗi hành vi của người dùng trong một session.
- Trạng thái ẩn của RNN lưu giữ thông tin ngữ cảnh, từ đó dự đoán hành vi tiếp theo.
- Không yêu cầu thông tin định danh người dùng (user ID), thích hợp cho các hệ thống khuyến nghị ẩn danh (anonymous recommendation).

Kiến trúc mô hình

- **Input:** Chuỗi các item trong một session $s = (i_1, i_2, \dots, i_T)$.
- **Embedding layer:** Mỗi item i_t được ánh xạ thành vector embedding $\mathbf{e}_{i_t} \in \mathbb{R}^d$.
- **GRU Layer:** Dùng GRU để cập nhật trạng thái ẩn:

$$\mathbf{h}_t = \text{GRU}(\mathbf{e}_{i_t}, \mathbf{h}_{t-1})$$

- **Output Layer:** Dùng trạng thái ẩn \mathbf{h}_t để tính điểm (score) cho tất cả các item:

$$\hat{y}_t = \text{softmax}(W\mathbf{h}_t + b)$$

với $\hat{y}_t \in \mathbb{R}^N$ là phân phối xác suất trên toàn bộ item.

Hàm mất mát

Bài toán được coi như **dự đoán item tiếp theo** trong phiên. Các hàm mất mát được đề xuất:

- **Cross-entropy loss:**

$$\mathcal{L}_{CE} = - \sum_{t=1}^T \log \hat{y}_t[i_{t+1}]$$

trong đó $\hat{y}_t[i_{t+1}]$ là xác suất dự đoán đúng item tiếp theo.

- **Ranking loss** (TOP1, BPR):

 - **TOP1:**

$$\mathcal{L}_{TOP1} = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} \sigma(\hat{x}_{uj} - \hat{x}_{ui}) + \sigma(\hat{x}_{uj}^2)$$

 - **BPR Loss** (tương tự [20]):

$$\mathcal{L}_{BPR} = - \sum_{(i,j)} \ln \sigma(\hat{x}_{ui} - \hat{x}_{uj})$$

Thuật toán huấn luyện

- Các session được chia thành batch động (mini-batch processing).
- SGD hoặc RMSProp được dùng để tối ưu.
- Negative sampling được áp dụng để giảm chi phí tính toán khi tính loss.

Ưu điểm

- Mô hình hóa được ngữ cảnh tuần tự trong phiên.
- Không cần user ID, phù hợp cho recommendation ẩn danh.
- Hiệu quả vượt trội hơn các baseline truyền thống (k-NN, item similarity).

Hạn chế

- Chỉ khai thác thông tin trong một session, bỏ qua lịch sử dài hạn của user.
- Huấn luyện trên dữ liệu lớn tốn tài nguyên.
- Các phiên ngắn có thể không đủ thông tin để học biểu diễn tốt.

6.4.3 SASRec: Self-Attentive Sequential Recommendation [14]

Giới thiệu

SASRec (Self-Attentive Sequential Recommendation) [14] là một trong những mô hình **Transformer-based** đầu tiên được áp dụng thành công cho bài toán gợi ý tuần tự. Khác với các mô hình dựa trên RNN/LSTM/GRU vốn dựa trên lan truyền tuần tự, SASRec tận dụng cơ chế *self-attention* để mô hình hóa quan hệ phụ thuộc giữa các item trong chuỗi lịch sử mà không cần phụ thuộc theo thứ tự tuần tự. Điều này mang lại khả năng: (i) nắm bắt được cả phụ thuộc ngắn hạn lẫn dài hạn, (ii) song song hóa trong huấn luyện, (iii) diễn giải tốt hơn thông qua trọng số attention.

Mục tiêu của SASRec là học phân phối điều kiện:

$$P(i_{t+1} | i_1, i_2, \dots, i_t),$$

trong đó i_t là item mà người dùng đã tương tác ở bước t trong một session.

Chuẩn bị dữ liệu huấn luyện

Dữ liệu gốc thường ở dạng các bộ ba (*user, item, timestamp*). Để huấn luyện, cần biến đổi thành các chuỗi tuần tự theo từng người dùng hoặc phiên làm việc:

$$s_u = [i_1, i_2, \dots, i_T],$$

trong đó s_u là lịch sử tương tác của người dùng u được sắp xếp theo thời gian.

Từ một chuỗi s_u , sinh ra các cặp huấn luyện (*input sequence, target*):

$$([i_1], i_2), ([i_1, i_2], i_3), \dots, ([i_1, \dots, i_{T-1}], i_T).$$

Ví dụ, với chuỗi [A, B, C, D]:

- Input: [A] → Target: B
- Input: [A, B] → Target: C

- Input: [A, B, C] → Target: D

Do độ dài chuỗi khác nhau, SASRec sử dụng:

- **Truncation:** chỉ giữ L item gần nhất khi chuỗi dài hơn L .
- **Padding:** thêm token [PAD] vào đầu để đủ độ dài L khi chuỗi ngắn.
- **Masking:** che padding token và che các bước tương lai (causal masking).

Trong huấn luyện, với mỗi bước t , ngoài positive sample i_{t+1} , một số negative samples được lấy từ tập item để tối ưu hàm mất mát (binary cross-entropy hoặc BPR loss).

Kiến trúc SASRec

Dầu vào của SASRec là chuỗi embedding:

$$z_t^0 = e_t + p_t,$$

trong đó e_t là embedding của item i_t , p_t là positional embedding.

Mô hình gồm nhiều lớp self-attention và feed-forward network (FFN).

Self-attention. Với đầu vào $Z \in \mathbb{R}^{T \times d}$:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V,$$

trong đó $Q = ZW_Q$, $K = ZW_K$, $V = ZW_V$. SASRec sử dụng **causal masking** để đảm bảo chỉ sử dụng thông tin quá khứ.

Multi-head attention.

$$\text{MHA}(Z) = \text{Concat}(\text{head}_1, \dots, \text{head}_H)W_O.$$

Feed-forward network (FFN).

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

Kết hợp residual connection và layer normalization:

$$Z^{(l+1)} = \text{LayerNorm}(Z^{(l)} + \text{MHA}(Z^{(l)})),$$

$$Z^{(l+1)} = \text{LayerNorm}(Z^{(l+1)} + \text{FFN}(Z^{(l+1)})).$$

Đầu ra. Sau L lớp, tại thời điểm t , vector z_t^L dùng để dự đoán item tiếp theo:

$$\hat{y}_{t+1} = \text{softmax}(z_t^L W^T),$$

trong đó W là ma trận embedding item (thường chia sẻ với embedding đầu vào).

Hàm mất mát

SASRec thường dùng **binary cross-entropy** với negative sampling:

$$\mathcal{L}_t = -\log \sigma(\hat{y}_{t,i^*}) - \sum_{j=1}^{N_s} \log \sigma(-\hat{y}_{t,j}),$$

với i^* là item đúng tại bước $t + 1$, j là negative samples.

Đặc điểm và mở rộng

- **Ưu điểm:** huấn luyện song song, nắm bắt quan hệ dài hạn, diễn giải được.
- **Hạn chế:** độ phức tạp $O(T^2)$ theo độ dài chuỗi, dễ overfit khi dữ liệu nhỏ.
- **Mở rộng:**
 - **BERT4Rec:** học ngữ cảnh hai chiều với masked prediction.
 - **TiSASRec:** thêm thông tin khoảng cách thời gian.
 - **Transformer4Rec:** framework mở rộng cho thực nghiệm và triển khai.

Kết luận

SASRec là bước ngoặt trong gợi ý tuần tự, thay thế RNN bằng self-attention. Với attention nhân quả và khả năng huấn luyện song song, SASRec trở thành baseline mạnh mẽ, được ứng dụng rộng rãi trong thương mại điện tử, nhạc, video và nhiều hệ thống gợi ý quy mô lớn.

6.5 BERT4Rec: Bidirectional Encoder Representations for Sequential Recommendation

6.5.1 Giới thiệu

BERT4Rec (Sun et al., 2019[23]) là một mô hình tiên phong đưa kiến trúc **Transformer encoder hai chiều** (BERT – Bidirectional Encoder Representations

from Transformers) vào bài toán gọi ý tuần tự. Khác với SASRec vốn chỉ sử dụng cơ chế attention một chiều (causal attention), BERT4Rec tận dụng khả năng mô hình hóa ngữ cảnh **hai chiều**, tức là mỗi item có thể phụ thuộc đồng thời vào cả lịch sử trước và sau trong chuỗi. Điều này giúp BERT4Rec học biểu diễn giàu ngữ nghĩa hơn, tương tự cách BERT đã cách mạng hóa NLP.

6.5.2 Chuẩn bị dữ liệu huấn luyện

Tương tự SASRec, dữ liệu đầu vào của BERT4Rec được biểu diễn dưới dạng chuỗi tương tác

$$s_u = [i_1, i_2, \dots, i_T].$$

Khác biệt quan trọng là BERT4Rec áp dụng **Masked Item Prediction (MIP)**, tương tự như **Masked Language Modeling (MLM)** trong BERT. Cụ thể:

- Một tỷ lệ các vị trí trong chuỗi (ví dụ 15%) được thay bằng token đặc biệt **[MASK]**.
- Mô hình được huấn luyện để dự đoán item gốc tại các vị trí đã bị che.
- Nhờ đó, mô hình học biểu diễn hai chiều, thay vì chỉ phụ thuộc vào lịch sử quá khứ.

Ví dụ, với chuỗi gốc: [A, B, C, D], sau khi mask có thể thành: [A, [MASK], C, D]. Mô hình được yêu cầu dự đoán item ở vị trí thứ 2 là B.

6.5.3 Kiến trúc mô hình

Kiến trúc của BERT4Rec gần giống BERT trong NLP, bao gồm:

- **Embedding layer**: ánh xạ item ID và vị trí sang vector.
- **Bidirectional Transformer encoder**: nhiều lớp self-attention không dùng causal mask, cho phép mỗi vị trí trong chuỗi chú ý đến cả trái và phải.
- **Prediction head**: softmax dự đoán xác suất item tại các vị trí bị mask.

Với input $Z \in \mathbb{R}^{T \times d}$, cơ chế self-attention giống SASRec:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V,$$

nhưng không áp dụng *causal masking*, do đó mỗi vị trí có thể truy cập toàn bộ ngữ cảnh.

6.5.4 Hàm mất mát

Hàm mất mát được định nghĩa dựa trên các vị trí bị che:

$$\mathcal{L} = - \sum_{t \in M} \log P(i_t | s_u^{\setminus M}),$$

trong đó:

- M là tập các vị trí bị mask,
- $s_u^{\setminus M}$ là chuỗi đầu vào sau khi mask.

6.5.5 Ưu điểm và hạn chế

Ưu điểm.

- Học biểu diễn hai chiều giàu thông tin, mạnh hơn mô hình một chiều như SASRec.
- Masked training giúp mô hình hóa tốt hơn các quan hệ phức tạp trong chuỗi.
- Có khả năng dự đoán tốt hơn trong các kịch bản cold-start ngắn hạn.

Hạn chế.

- Huấn luyện phức tạp và tốn tài nguyên hơn do masking.
- Không trực tiếp tối ưu cho bài toán *next-item prediction*, mà thông qua pre-training với MIP.

6.5.6 Kết luận

BERT4Rec mở rộng ý tưởng từ SASRec bằng cách áp dụng cơ chế attention hai chiều và masked training. Trong khi SASRec tối ưu trực tiếp cho bài toán dự đoán item tiếp theo, BERT4Rec khai thác toàn bộ ngữ cảnh để học biểu diễn mạnh mẽ hơn, từ đó cải thiện hiệu năng trong nhiều tình huống. Hai mô hình này bổ sung cho nhau: SASRec đơn giản, nhanh và hiệu quả cho dự đoán trực tiếp, trong khi BERT4Rec phù hợp với huấn luyện trên dữ liệu lớn và các kịch bản recommendation phức tạp.

6.5.7 TiSASRec: Time Interval Aware Self-Attention for Sequential Recommendation [16]

Giới thiệu

TiSASRec mở rộng SASRec bằng cách tích hợp thông tin **thời gian** vào cơ chế self-attention. Trong nhiều ứng dụng thực tế (ví dụ e-commerce, xem video, nghe nhạc), khoảng thời gian giữa các hành vi của người dùng phản ánh mạnh mẽ sở thích:

- Nếu hai hành vi diễn ra gần nhau, chúng thường liên quan về mặt ngữ nghĩa.
- Nếu khoảng cách thời gian xa, sự phụ thuộc giữa chúng yếu dần.

SASRec gốc chỉ xét đến **thứ tự tuần tự**, bỏ qua yếu tố thời gian. TiSASRec bổ sung **time interval encoding** để khắc phục hạn chế này.

Ý tưởng chính

- Thay vì chỉ dùng positional encoding như SASRec, TiSASRec kết hợp thêm **temporal encoding**.
- Cơ chế attention được điều chỉnh để xét đến cả vị trí và khoảng cách thời gian giữa các item.
- Nhờ đó, mô hình nắm bắt tốt hơn sự thay đổi sở thích của người dùng theo thời gian.

Kiến trúc mô hình

TiSASRec giữ nguyên khung của SASRec, nhưng thêm module xử lý thời gian:

- **Input sequence:** $s_u = [i_1, i_2, \dots, i_T]$, kèm theo timestamp $[t_1, t_2, \dots, t_T]$.
- **Item embedding:** ánh xạ mỗi item i_t thành $\mathbf{e}_{i_t} \in \mathbb{R}^d$.
- **Positional embedding:** giữ nguyên như SASRec.
- **Time interval embedding:** với hai hành vi i_m, i_n có khoảng cách thời gian $\Delta t_{m,n} = |t_m - t_n|$, gán embedding $\mathbf{e}_{\Delta t_{m,n}}$.
- **Time-aware attention:** công thức attention được điều chỉnh:

$$\alpha_{mn} = \frac{(\mathbf{h}_m W_Q)(\mathbf{h}_n W_K + \mathbf{e}_{pos}(m-n) + \mathbf{e}_{\Delta t_{m,n}})^T}{\sqrt{d}},$$

với α_{mn} là trọng số attention từ vị trí m đến n .

- **Prediction layer:** tương tự SASRec, dùng trạng thái cuối để dự đoán item tiếp theo.

Hàm mất mát

Mục tiêu vẫn là **next-item prediction**. Hàm mất mát thường dùng là **Binary Cross-Entropy (BCE)** hoặc **BPR loss**:

$$\mathcal{L}_{BPR} = - \sum_{(i,j)} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}),$$

trong đó i là item đúng, j là item âm được lấy mẫu.

Ưu điểm

- Mô hình hóa cả **thứ tự tuần tự** và **yếu tố thời gian**.
- Nâng cao độ chính xác trong các hệ thống mà thời gian giữa hành vi quan trọng (mua sắm, âm nhạc, video).
- Giữ lại sự linh hoạt và khả năng song song hóa của Transformer.

Hạn chế

- Chi phí tính toán cao hơn SASRec do phải tính embedding cho từng cặp khoảng thời gian.
- Với dữ liệu có độ phân giải thời gian quá mịn (ví dụ tính theo giây), số lượng interval có thể quá lớn.
- Không luôn cần thiết trong các domain mà thời gian không mang nhiều ý nghĩa.

Kết luận

TiSASRec là bước tiến tự nhiên từ SASRec, khi bổ sung thêm yếu tố thời gian vào self-attention. Nếu SASRec chỉ mô hình hóa **thứ tự**, thì TiSASRec mô hình hóa cả **thứ tự + thời gian**, từ đó phù hợp hơn cho recommendation trong các hệ thống giàu thông tin temporal.

6.5.8 So sánh SASRec, BERT4Rec và TiSASRec

So sánh ba mô hình SASRec, BERT4Rec và TiSASRec có thể được trình bày theo từng khía cạnh như sau:

- **Hướng attention:**

- **SASRec:** sử dụng attention một chiều (causal), chỉ dựa vào lịch sử quá khứ.
- **BERT4Rec:** attention hai chiều (bidirectional), cho phép tận dụng toàn bộ ngữ cảnh cả trước và sau.
- **TiSASRec:** attention một chiều (causal) nhưng có bổ sung thông tin thời gian giữa các hành vi.

- **Thông tin bổ sung:**

- **SASRec:** positional embedding để mã hóa thứ tự.
- **BERT4Rec:** positional embedding kết hợp với Masked Item Prediction.
- **TiSASRec:** positional embedding và **time interval embedding** để mã hóa khoảng cách thời gian giữa các hành vi.

- **Mục tiêu huấn luyện:**

- **SASRec:** dự đoán trực tiếp item tiếp theo (next-item prediction).
- **BERT4Rec:** Masked Item Prediction (MIP), thường dùng trong giai đoạn pretraining rồi fine-tuning.
- **TiSASRec:** next-item prediction trực tiếp, có tính đến yếu tố thời gian.

- **Biểu diễn học được:**

- **SASRec:** dựa thuần túy trên lịch sử tuần tự.
- **BERT4Rec:** học được biểu diễn hai chiều, giàu thông tin hơn.
- **TiSASRec:** học biểu diễn tuần tự kết hợp cả thứ tự và yếu tố thời gian.

- **Ưu điểm:**

- **SASRec:** đơn giản, hiệu quả, dễ huấn luyện.
- **BERT4Rec:** khai thác toàn bộ ngữ cảnh, học được biểu diễn mạnh mẽ.
- **TiSASRec:** nắm bắt được cả thứ tự lẫn thời gian, phù hợp cho dữ liệu có tính *temporal*.

- **Hạn chế:**

- **SASRec:** không xét đến thông tin từ tương lai hoặc thời gian.
- **BERT4Rec:** huấn luyện phức tạp, không tối ưu trực tiếp cho next-item prediction.
- **TiSASRec:** chi phí tính toán cao, số lượng interval embedding lớn nếu dữ liệu dày đặc.

- **Ứng dụng phù hợp:**

- **SASRec:** hệ thống e-commerce, nhạc, video với yêu cầu dự đoán trực tiếp item tiếp theo.
- **BERT4Rec:** phù hợp cho pretraining + fine-tuning trong hệ thống lớn với dữ liệu chuỗi dài.
- **TiSASRec:** ứng dụng có yêu tố thời gian rõ rệt như mua sắm, nghe nhạc, xem video theo session.

Chương 7

GỢI Ý DỰA TRÊN LAI GHÉP

7.1	Giới thiệu về hệ gợi ý lai	121
7.2	Thiết kế lai trong hệ gợi ý	122
7.3	Thiết kế lai đơn khối	123
7.4	Thiết kế lai song song	125
7.5	Các thiết kế lai nâng cao	128

7.1 Giới thiệu về hệ gợi ý lai

Trong các hệ thống gợi ý, việc kết hợp nhiều thuật toán khác nhau để tạo thành **hệ gợi ý lai** ngày càng được quan tâm do khả năng cải thiện độ chính xác và khắc phục hạn chế của từng phương pháp đơn lẻ. Một ví dụ nổi bật là cuộc thi Netflix Prize, nơi hàng trăm nhà nghiên cứu đã kết hợp nhiều kỹ thuật lọc cộng tác khác nhau nhằm nâng cao hiệu quả gợi ý phim.

Các hệ gợi ý lai có thể dựa trên ba hướng tiếp cận nền tảng bao gồm: *lọc cộng tác* (*collaborative filtering*), *dựa trên nội dung* (*content-based*) và *dựa trên tri thức* (*knowledge-based*). Ngoài ra, các hệ gợi ý dựa trên độ hữu ích (*utility-based*) có thể được xem như một trường hợp đặc biệt của hướng dựa trên tri thức, và các hệ gợi ý nhân khẩu học (*demographic-based*) có thể được coi là biến thể của lọc cộng tác khi tận dụng thông tin hồ sơ nhân khẩu học của người dùng để xác định nhóm tương đồng.

Một khía cạnh cốt lõi trong thiết kế hệ gợi ý lai là **cách thức kết hợp** các thuật toán. Các thành phần gợi ý có thể hoạt động song song rồi gộp kết quả, hoặc được

kết nối theo dạng chuỗi (*pipeline*), trong đó đầu ra của một hệ thống trở thành đầu vào của hệ tiếp theo.

Về mặt lý thuyết, bài toán gợi ý có thể được mô hình hóa như một hàm tiện ích $rec(u, i)$, biểu diễn độ hữu ích của một sản phẩm i đối với người dùng u . Nhiệm vụ của hệ gợi ý là chọn ra tập n sản phẩm có giá trị tiện ích cao nhất cho từng người dùng. Tùy theo cách tiếp cận, giá trị tiện ích này có thể được ước lượng từ dữ liệu cộng đồng (lọc cộng tác), từ đặc trưng sản phẩm (dựa trên nội dung), hoặc từ tri thức được mã hóa và các lược đồ tiện ích (dựa trên tri thức).

Quan trọng hơn, mỗi mô hình gợi ý yêu cầu các loại dữ liệu đầu vào khác nhau. Cụ thể:

- **Lọc cộng tác:** cần dữ liệu cộng đồng và hồ sơ người dùng.
- **Dựa trên nội dung:** cần đặc trưng sản phẩm và hồ sơ người dùng.
- **Dựa trên tri thức:** cần tri thức miền, mô hình tiện ích và hồ sơ người dùng.

Tất cả các hướng tiếp cận đều phải truy cập **mô hình người dùng** và **tham số ngữ cảnh** để cá nhân hóa kết quả. Tuy nhiên, trong nhiều tình huống thực tế, chỉ một phần thông tin người dùng có sẵn, do đó không phải biến thể nào cũng có thể áp dụng.

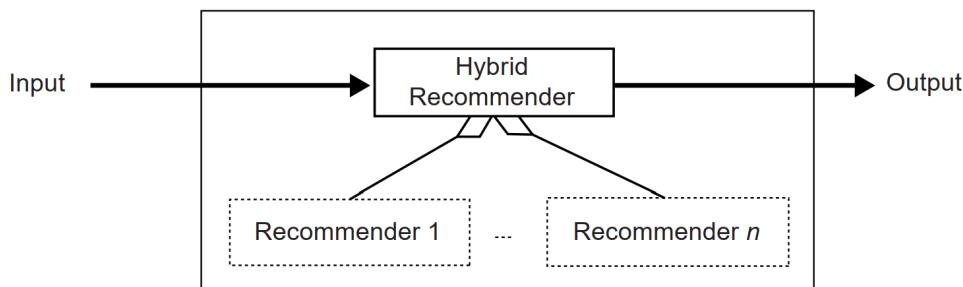
7.2 Thiết kế lai trong hệ gợi ý

Một chiêu đặc trưng quan trọng của các thuật toán lai là **thiết kế**. Có thể phân loại thành ba dạng cơ bản: *monolithic*, *parallelized* và *pipelined*.

- **Lai khôi:** Đây là thiết kế trong đó một thuật toán gợi ý đơn lẻ được xây dựng để kết hợp nhiều chiến lược khác nhau. Trong trường hợp này, các hệ gợi ý khác nhau đóng góp một cách gián tiếp thông qua dữ liệu đầu vào. Ví dụ, một hệ gợi ý dựa trên nội dung có thể khai thác dữ liệu cộng đồng để xác định độ tương đồng giữa các sản phẩm. Như vậy, dữ liệu được tăng cường bằng một kỹ thuật, và sau đó được khai thác thực tế bởi kỹ thuật khác.
- **Lai song song:** Trong thiết kế này, ít nhất hai hệ gợi ý riêng biệt được triển khai song song. Mỗi hệ hoạt động độc lập trên dữ liệu đầu vào của mình và tạo ra danh sách gợi ý riêng. Sau đó, trong một bước lai hóa, các kết quả này được kết hợp lại để hình thành tập gợi ý cuối cùng. Các chiến lược như gán trọng số, trộn hoặc chuyển đổi được thực hiện trong cấu trúc song song này.

- **Lai theo cơ chế chuỗi:** Với thiết kế dạng chuỗi (pipeline), đầu ra của một hệ gợi ý trở thành đầu vào cho hệ gợi ý kế tiếp. Ngoài ra, các thành phần phía sau trong chuỗi có thể tận dụng thêm một phần dữ liệu gốc ban đầu. Các kỹ thuật như *cascade* hoặc *meta-level* là ví dụ tiêu biểu cho loại thiết kế này.

7.3 Thiết kế lai đơn khối



Hình 7.1: Minh họa hệ gợi ý lai đơn khối.

Khác với hai thiết kế khác trong hệ gợi ý – vốn bao gồm hai hoặc nhiều thành phần riêng biệt rồi kết hợp kết quả – các hệ lai đơn khối bao gồm một thành phần gợi ý duy nhất (Hình 7.1, trong đó tích hợp nhiều cách tiếp cận thông qua tiền xử lý và kết hợp nhiều nguồn tri thức). Việc lai hóa được thực hiện bằng cách sửa đổi hành vi của thuật toán để khai thác nhiều loại dữ liệu đầu vào khác nhau. Thông thường, các bước tiền xử lý theo đặc thù dữ liệu sẽ được dùng để biến đổi dữ liệu thành dạng biểu diễn phù hợp với một mô hình thuật toán cụ thể. Trong loại thiết kế này, có thể phân biệt hai chiến lược chính: **kết hợp đặc trưng** (feature combination) và **bổ sung đặc trưng** (feature augmentation).

7.3.1 Hệ gợi ý lai kết hợp đặc trưng

Hệ gợi ý loại này sử dụng nhiều loại dữ liệu đầu vào khác nhau trong cùng một thành phần đơn lẻ. Ví dụ, một cách tiếp cận kết hợp đặc trưng có thể đồng thời khai thác đặc trưng cộng tác (như lượt thích/mua của người dùng) và đặc trưng nội dung (ví dụ: thể loại của sản phẩm).

Để minh họa, giả sử một bảng dữ liệu lưu lại lượt mua sách của một số người dùng, trong đó thông tin sản phẩm chỉ giới hạn ở thể loại sách. Nếu chỉ áp dụng gợi ý cộng tác, ta sẽ thấy người dùng A có vẻ giống với cả User1 và User2. Tuy nhiên, sau khi biến đổi dữ liệu thành các đặc trưng lai – chẳng hạn như “Người dùng thích

nhiều sách thuộc thể loại X” hoặc “Người dùng thích một vài sách thể loại X” – bức tranh trở nên khác. Khi đó, User1 được nhận diện là có hành vi giống Alice hơn, vì họ cùng tập trung nhiều vào sách trinh thám, trong khi User2 lại tỏ ra không ổn định khi mua dàn trải ở cả ba thể loại.

Ngoài ra, trong một số nghiên cứu khác, các loại phản hồi người dùng cũng được phân loại và kết hợp dựa trên mức độ chính xác dự đoán cũng như độ phổ biến. Ví dụ, phản hồi có thể gồm: hành vi điều hướng (R_{nav}), lượt nhấp xem chi tiết (R_{view}), yêu cầu ngữ cảnh (R_{ctx}), và mua hàng thực tế (R_{buy}). Các loại phản hồi này có độ tin cậy khác nhau: mua hàng và yêu cầu ngữ cảnh thường mang nhiều giá trị dự đoán, trong khi hành vi điều hướng lại dễ nhiều hơn. Do đó, các nguồn phản hồi này được ưu tiên theo thứ tự:

$$(R_{buy}, R_{ctx}) \prec R_{view} \prec R_{nav}.$$

Thuật toán kết hợp đặc trưng sẽ dựa vào mức ưu tiên này để chọn ra những người dùng tương tự với mức độ tin cậy cao hơn.

Nhờ sự đơn giản, phương pháp kết hợp đặc trưng ngày nay rất phổ biến, nhất là trong việc tích hợp gợi ý cộng tác và gợi ý theo nội dung. Tuy nhiên, việc kết hợp thêm tri thức dựa trên ràng buộc (knowledge-based), chẳng hạn như “giá phải thấp hơn sản phẩm A”, với dữ liệu cộng tác hay nội dung vẫn chưa được nghiên cứu nhiều. Đây có thể là một hướng tiềm năng trong tương lai.

7.3.2 Hệ gợi ý lai bổ sung đặc trưng

Khác với kết hợp đặc trưng, hệ lai bổ sung đặc trưng không chỉ đơn thuần ghép dữ liệu đầu vào, mà còn áp dụng các bước biến đổi phức tạp hơn. Trong đó, đầu ra của một bộ gợi ý sẽ được dùng để **bổ sung đặc trưng** cho bộ gợi ý chính. Điều này khác với thiết kế “chuỗi xử lý” (pipelined), vì các thuật toán con ở đây gắn chặt vào thành phần chính để đảm bảo hiệu năng và chức năng.

Một ví dụ điển hình là *lọc cộng tác tăng cường nội dung* (content-boosted collaborative filtering). Trong phương pháp này, nếu người dùng chưa đánh giá một sản phẩm, hệ thống sẽ dùng mô hình dựa trên nội dung để dự đoán giá trị và coi đó như một “giả đánh giá” (pseudo-rating). Sau đó, ma trận đánh giá kết hợp cả đánh giá thật và dự đoán nội dung sẽ được dùng cho thuật toán cộng tác.

Công thức dự đoán của hệ thống này được xây dựng dựa trên bình cộng có trọng số, trong đó các trọng số được điều chỉnh theo:

- Số lượng sản phẩm mà hai người dùng cùng đánh giá (yếu tố “độ tin cậy về ý

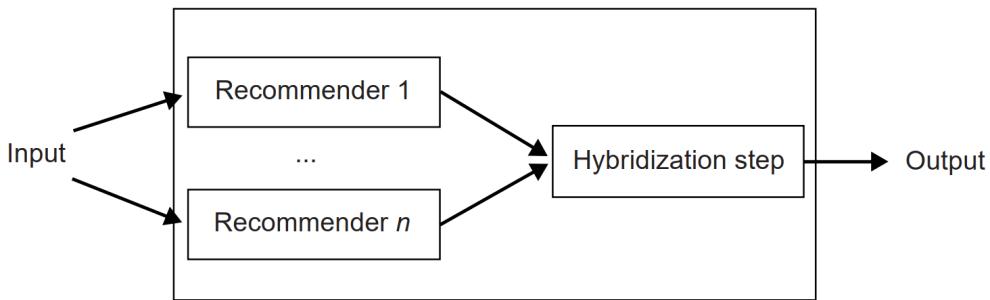
nghĩa”).

- Số lượng tổng đánh giá mà một người dùng thực hiện (tránh phụ thuộc vào người có quá ít dữ liệu).
- Mức độ tin cậy của mô hình nội dung khi tạo ra các “giả đánh giá”.

Ví dụ, nếu Alice chưa đánh giá sản phẩm *Item5*, hệ thống sẽ tạo giá trị dự đoán dựa trên cả dữ liệu cộng tác và nội dung. Kết quả cho thấy mặc dù User1 (người giống Alice nhất) đánh giá *Item5* khá cao, nhưng do các trọng số điều chỉnh, hệ thống dự đoán rằng Alice sẽ không hứng thú với sản phẩm này.

Các ứng dụng khác của bổ sung đặc trưng bao gồm hệ gợi ý sách dựa trên nội dung, hay hệ gợi ý bài báo khoa học trong đó trích dẫn giữa các bài được diễn giải như một dạng dữ liệu cộng tác.

7.4 Thiết kế lai song song



Hình 7.2: Minh họa hệ gợi ý lai song song.

7.4.1 Thiết kế lai song song

Các thiết kế lai song song (Hình 7.1) sử dụng nhiều bộ gợi ý cùng lúc và áp dụng một cơ chế lai cụ thể để tổng hợp đầu ra của chúng. Các chiến lược kết hợp phổ biến bao gồm lai trộn, lai có trọng số và lai chuyển đổi. Ngoài ra, các chiến lược kết hợp khác cho nhiều danh sách gợi ý, chẳng hạn như sơ đồ bô phiếu đa số, cũng có thể được áp dụng.

7.4.2 Lai trộn (Mixed hybrids)

Chiến lược lai trộn kết hợp kết quả của các hệ thống gợi ý khác nhau ở cấp giao diện người dùng, trong đó các kết quả từ các kỹ thuật khác nhau được trình bày cùng nhau. Do đó, kết quả gợi ý cho người dùng u và sản phẩm i của một chiến lược lai trộn là tập hợp các bộ ($score, k$) cho từng bộ gợi ý thành phần rec_k :

$$rec_{mixed}(u, i) = \{(rec_k(u, i), k) \mid k = 1 \dots n\} \quad (7.1)$$

Các sản phẩm được xếp hạng cao nhất từ mỗi bộ gợi ý sau đó được hiển thị cho người dùng. Tuy nhiên, khi hợp nhất các kết quả khác nhau thành một thực thể duy nhất (ví dụ: lịch phát sóng truyền hình), cần có một cơ chế giải quyết xung đột. Một biến thể khác của lai trộn là gộp các gợi ý từ nhiều hệ thống khác nhau theo từng danh mục sản phẩm. Ví dụ: trong miền du lịch, một gói gợi ý có thể bao gồm chỗ ở, các hoạt động thể thao và giải trí, được sinh ra bởi các hệ gợi ý riêng biệt. Một bộ giải bài toán ràng buộc (CSP solver) được sử dụng để giải quyết xung đột, đảm bảo rằng chỉ các tập hợp mục nhất quán theo ràng buộc miền (ví dụ: “hoạt động và chỗ ở phải cách nhau dưới 50 km”) mới được gợi ý.

7.4.3 Lai có trọng số (Weighted hybrids)

Chiến lược lai có trọng số kết hợp các gợi ý từ hai hoặc nhiều hệ thống bằng cách tính tổng có trọng số của điểm số. Với n bộ gợi ý rec_k và trọng số tương ứng β_k :

$$rec_{weighted}(u, i) = \sum_{k=1}^n \beta_k \times rec_k(u, i) \quad (7.2)$$

trong đó điểm số cần được chuẩn hóa về cùng một miền giá trị, và $\sum_{k=1}^n \beta_k = 1$.

Bảng 7.1: Ví dụ về lai có trọng số

Item	rec1		rec2		recw	
	score	rank	score	rank	score	rank
Item1	0.5	1	0.8	2	0.65	1
Item2	0	-	0.9	1	0.45	2
Item3	0.3	2	0.4	3	0.35	3
Item4	0.1	3	0	-	0.05	-
Item5	0	-	0	-	0	-

Ví dụ trên cho thấy mặc dù một sản phẩm chỉ được một hệ gợi ý khuyến nghị

(ví dụ: Item2), nó vẫn có thể được xếp hạng cao sau bước lai trọng số. Khi trọng số thay đổi, chất lượng dự đoán cũng thay đổi. Để đánh giá, có thể sử dụng độ sai tuyệt đối trung bình (MAE):

$$MAE = \frac{\sum_{r_i \in R} \sum_{k=1}^n \beta_k \times |rec_k(u, i) - r_i|}{|R|} \quad (7.3)$$

Trong đó R là tập các sản phẩm đã được người dùng đánh giá.

Bảng 7.2: Các tham số trọng số động, sai số tuyệt đối và MAE cho người dùng Alice

β_1	β_2	Item	r_i	rec1	rec2	error / MAE
0.1	0.9	Item1	1	0.5	0.8	0.23
		Item4	1	0.1	0	0.99 / 0.61
0.3	0.7	Item1	1	0.5	0.8	0.29
		Item4	1	0.1	0	0.97 / 0.63
0.5	0.5	Item1	1	0.5	0.8	0.35
		Item4	1	0.1	0	0.95 / 0.65
0.7	0.3	Item1	1	0.5	0.8	0.41
		Item4	1	0.1	0	0.93 / 0.67
0.9	0.1	Item1	1	0.5	0.8	0.47
		Item4	1	0.1	0	0.91 / 0.69

Rõ ràng, việc gán trọng số động sẽ ổn định dần khi có nhiều dữ liệu đánh giá hơn từ người dùng. Ngoài MAE, các độ đo sai số khác như sai số bình phương trung bình (MSE) hoặc độ đo dựa trên thứ hạng cũng có thể được sử dụng. Trong trường hợp cực đoan, việc điều chỉnh trọng số động có thể dẫn đến một hệ lai chuyển đổi.

7.4.4 Lai chuyển đổi (Switching hybrids)

Hệ lai chuyển đổi yêu cầu một “bộ chọn” quyết định sử dụng bộ gợi ý nào trong một tình huống cụ thể, phụ thuộc vào hồ sơ người dùng và/hoặc chất lượng kết quả gợi ý. Công thức có thể biểu diễn như sau:

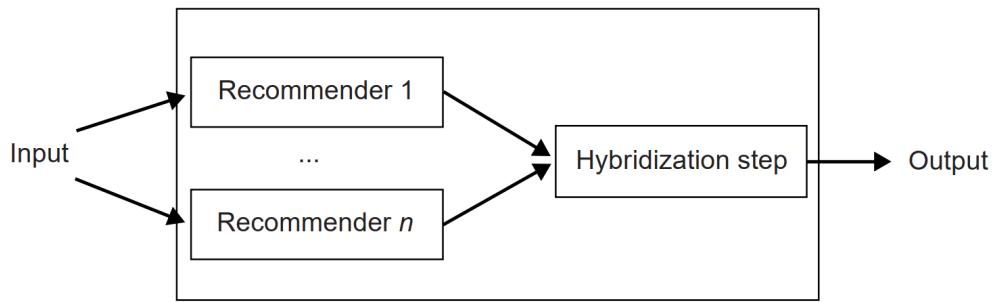
$$\exists! k \in \{1 \dots n\} : rec_{switching}(u, i) = rec_k(u, i) \quad (7.4)$$

Trong đó k được xác định bởi điều kiện chuyển đổi. Ví dụ: để giải quyết vấn đề cold-start, hệ lai chuyển đổi có thể bắt đầu bằng bộ gợi ý dựa trên tri thức, sau đó chuyển sang lọc cộng tác khi dữ liệu đánh giá đủ lớn. Các chiến lược chuyển đổi cũng có thể được áp dụng để tối ưu kết quả, chẳng hạn sử dụng tuần tự các bộ lọc nội dung, lọc cộng tác và phân loại Bayes ngay thơ. Ngoài ra, các tiêu chí chuyển

đổi có thể được điều chỉnh linh hoạt hơn, kể cả dựa trên ngữ cảnh, ý định hoặc kỳ vọng của người dùng.

Tóm lại, chất lượng của cơ chế chuyển đổi là yếu tố then chốt trong biến thể lai này.

7.5 Các thiết kế lai nâng cao



Hình 7.3: Minh họa hệ gợi ý lai chuỗi.

7.5.1 Lai theo chuỗi xử lý (Pipelined Hybridization)

Trong thiết kế lai theo chuỗi xử lý (Hình 7.3), các kỹ thuật gợi ý được sắp xếp theo từng giai đoạn. Kết quả hoặc mô hình được sinh ra từ một giai đoạn sẽ được sử dụng làm đầu vào cho giai đoạn kế tiếp, cho đến khi giai đoạn cuối cùng sinh ra danh sách gợi ý cho người dùng. Các biến thể của thiết kế này khác nhau chủ yếu ở dạng đầu ra được truyền cho giai đoạn tiếp theo: một thành phần có thể tiền xử lý dữ liệu đầu vào để xây dựng mô hình, hoặc có thể sinh ra một danh sách gợi ý để tinh chỉnh thêm.

7.5.2 Lai theo tầng (Cascade Hybridization)

Trong thiết kế lai theo tầng, các kỹ thuật gợi ý được áp dụng tuần tự, trong đó mỗi kỹ thuật sau chỉ *tinh chỉnh* kết quả của kỹ thuật trước. Danh sách gợi ý của kỹ thuật thứ k chỉ bao gồm các mục đã được gợi ý (có điểm khác 0) bởi kỹ thuật thứ $k - 1$.

Giả sử có n kỹ thuật, với rec_1 là kỹ thuật đầu tiên và rec_n là kỹ thuật cuối cùng. Công thức tổng quát của gợi ý theo tầng được viết như sau:

$$rec_{\text{cascade}}(u, i) = rec_n(u, i)$$

trong đó, với mọi $k \geq 2$ phải thỏa mãn:

$$rec_k(u, i) = \begin{cases} rec_k(u, i) & \text{nếu } rec_{k-1}(u, i) \neq 0 \\ 0 & \text{ngược lại} \end{cases}$$

Trong thiết kế này, các kỹ thuật (từ $k = 2$ trở đi) chỉ có thể thay đổi thứ hạng của các mục đã được gợi ý từ trước hoặc loại bỏ chúng bằng cách đặt giá trị tiện ích bằng 0. Chúng không thể thêm mới các mục đã bị loại bỏ ở các bước trước đó. Điểm hạn chế là kích thước tập gợi ý có thể giảm dần theo số tầng, và trong một số tình huống, hệ thống không tạo ra đủ số lượng gợi ý cần thiết. Để khắc phục, người ta có thể kết hợp với chiến lược chuyển đổi (switching), chẳng hạn chuyển sang phương pháp lai có trọng số khi số lượng gợi ý quá ít.

7.5.3 Lai theo siêu mức (Meta-level Hybridization)

Trong thiết kế lai theo siêu mức, một hệ gợi ý đầu tiên xây dựng mô hình, sau đó mô hình này được một hệ gợi ý chính khác khai thác để sinh ra gợi ý. Công thức khái quát có thể được biểu diễn như sau:

$$rec_{\text{meta-level}}(u, i) = rec_n(u, i, rec_{n-1})$$

Trong đó, rec_{n-1} là mô hình do hệ gợi ý trước xây dựng, và rec_n là hệ gợi ý cuối cùng sử dụng mô hình này để dự đoán. Thông thường, $n = 2$ trong các hệ thống thực tế.

Ví dụ, một hệ gợi ý dựa trên nội dung có thể xây dựng hồ sơ người dùng từ văn bản hoặc thuộc tính sản phẩm. Sau đó, một hệ gợi ý cộng tác có thể sử dụng những hồ sơ này để tìm ra người dùng tương tự và đưa ra gợi ý dựa trên hành vi của nhóm người dùng tương đồng. Cách tiếp cận này đặc biệt hữu ích trong trường hợp dữ liệu còn thừa thớt, vì nó kết hợp ưu điểm của hai phương pháp: khả năng khởi tạo hồ sơ từ dữ liệu nội dung và khả năng tận dụng sự tương đồng từ cộng đồng người dùng.

Chương 8

ĐÁNH GIÁ HỆ THỐNG GỌI Ý

8.1	Giới thiệu	130
8.2	Các đặc điểm chung của nghiên cứu đánh giá	131
8.3	Đối tượng nghiên cứu	132
8.4	Hạn chế của các phương pháp đánh giá	133
8.5	Phương pháp nghiên cứu (Research Methods)	133
8.6	Thiết lập đánh giá (Evaluation Settings)	134
8.7	Các độ đo phổ biến trong đánh giá hệ gợi ý dựa trên dữ liệu lịch sử	135

8.1 Giới thiệu

Hệ gợi ý yêu cầu người dùng tương tác không chỉ với hệ thống máy tính mà còn với các người dùng khác. Do đó, nhiều phương pháp nghiên cứu hành vi xã hội có thể áp dụng để trả lời các câu hỏi như:

- Người dùng có thấy các tương tác với hệ thống gợi ý hữu ích không?
- Họ hài lòng với chất lượng gợi ý mà họ nhận được không?
- Điều gì thúc đẩy người dùng đóng góp dữ liệu như đánh giá hay bình luận để cải thiện chất lượng dự đoán của hệ thống?

- Người dùng thích nhận gợi ý vì yếu tố ngẫu nhiên và mồi mỉ hay chỉ vì họ được tiết kiệm thời gian tìm kiếm?

Ngoài các khía cạnh trải nghiệm người dùng, việc đánh giá còn liên quan đến các tiêu chí kỹ thuật của hệ thống như: khả năng phản hồi, khả năng mở rộng, xử lý tải cao, độ tin cậy, chi phí bảo trì, và khả năng mở rộng chức năng.

Các yếu tố cơ bản của phương pháp nghiên cứu gồm:

- Đối tượng nghiên cứu: người dùng hoặc phần cứng
- Phương pháp nghiên cứu: thí nghiệm, bán thí nghiệm, phi thí nghiệm
- Bối cảnh nghiên cứu: thực tế hoặc phòng thí nghiệm

8.2 Các đặc điểm chung của nghiên cứu đánh giá

Nghiên cứu thực nghiệm cần chú ý đến độ nghiêm ngặt và tính hợp lệ:

8.2.1 Tính mô tả và lặp lại

Mô tả chi tiết phương pháp, tuân thủ quy trình hệ thống, và ghi nhận các quyết định đảm bảo nghiên cứu có thể được lặp lại và kết quả được xác minh.

8.2.2 Tính hợp lệ

- **Hợp lệ nội tại (Internal validity):** Kết quả quan sát được là do điều kiện kiểm soát trong thí nghiệm, không phải do khác biệt giữa người tham gia hoặc tác động bên ngoài không kiểm soát.
- **Hợp lệ ngoại lai (External validity):** Kết quả có thể khái quát hóa sang các nhóm người dùng hoặc tình huống khác.

8.2.3 Độ tin cậy (Reliability)

Dữ liệu và phép đo không có sự mâu thuẫn hay sai sót.

8.2.4 Tính nhạy (Sensibility)

Các khác biệt quan sát được trong các phép đo phản ánh đúng sự khác biệt thực tế.

Ngoài ra, khi đánh giá kết quả, cần quan tâm không chỉ đến ý nghĩa thống kê mà còn đến **tầm ảnh hưởng thực tế**, ví dụ: mức tăng 10% độ chính xác dự đoán có giúp tăng độ trung thành của khách hàng hay giảm tỷ lệ rời bỏ nền tảng không.

8.3 Đối tượng nghiên cứu

8.3.1 Người dùng thực

Nhóm quan tâm bao gồm khách hàng trực tuyến, sinh viên, hoặc người dùng web nhận gợi ý cá nhân hóa.

8.3.2 Dữ liệu lịch sử hoặc tổng hợp

- **Dữ liệu tổng hợp (synthetic data):** Có thể kiểm soát được các tham số như phân phối người dùng, kích thước dữ liệu, độ thừa thớt. Phù hợp để thử nghiệm hiệu năng thuật toán nhưng có rủi ro thiên lệch.
- **Dữ liệu tự nhiên (historical data):** Dữ liệu tương tác thực tế, gồm đánh giá rõ ràng (explicit) và phản hồi ngầm (implicit, ví dụ: click, giao dịch mua).

8.3.3 Độ thừa dữ liệu (Sparsity)

$$\text{sparsity} = 1 - \frac{|R|}{|I| \cdot |U|}$$

trong đó:

- R = số đánh giá
- I = số sản phẩm
- U = số người dùng

8.3.4 Ví dụ tập dữ liệu phổ biến

MovieLens, EachMovie, Netflix, BX, Jester, Entree, Ta-Feng, CiteULike, Bibsonomy, del.icio.us.

8.4 Hạn chế của các phương pháp đánh giá

- Dữ liệu lịch sử chỉ biết các đánh giá đã có; các mục chưa đánh giá có thể bị hiểu nhầm là không thích, dẫn đến **false positives**.
- Thử nghiệm trực tiếp với người dùng có thể xác định **true positives** và **false positives**, nhưng không biết **false negatives**.
- Giải pháp hoàn hảo là cung cấp một **thị trường minh bạch** nơi người dùng đánh giá tất cả các mục, nhưng điều này khó thực hiện với số lượng sản phẩm lớn.

8.5 Phương pháp nghiên cứu (Research Methods)

Định nghĩa mục tiêu nghiên cứu và xác định các khía cạnh của người dùng hoặc đối tượng nghiên cứu có liên quan trong bối cảnh hệ gợi ý là điểm khởi đầu của bất kỳ đánh giá nào. Những khía cạnh được quan sát hoặc đo lường này được gọi là **biến** trong nghiên cứu thực nghiệm, và có thể là **biến độc lập** hoặc **biến phụ thuộc**.

- Một số biến luôn là độc lập theo bản chất, ví dụ: giới tính, thu nhập, trình độ học vấn, hoặc tính cách, vì chúng về nguyên tắc là cố định trong quá trình nghiên cứu.
- Biến độc lập khác có thể được kiểm soát bởi thiết kế đánh giá, ví dụ: loại thuật toán gợi ý áp dụng cho người dùng, hoặc các mục được gợi ý.
- Biến phụ thuộc là những biến chịu ảnh hưởng của biến độc lập, ví dụ: mức độ hài lòng của người dùng, cảm nhận về tính hữu ích, hoặc tỷ lệ click-through.

8.5.1 Thiết kế thí nghiệm (Experimental Design)

Trong nghiên cứu thí nghiệm, một hoặc nhiều biến độc lập được thao tác để xác định tác động của chúng lên biến phụ thuộc:

Các đối tượng (units) được phân ngẫu nhiên vào các **treatment** khác nhau (ví dụ: các thuật toán gợi ý khác nhau). Biến phụ thuộc (ví dụ: v_1, v_2) được đo trước và sau **treatment**, bằng cách sử dụng bảng hỏi hoặc quan sát hành vi người dùng. Các yếu tố môi trường từ bên ngoài thí nghiệm cũng cần được kiểm soát, ví dụ: phân nhóm người dùng theo kinh nghiệm hoặc trình độ với sản phẩm.

Trong các thí nghiệm offline với dữ liệu lịch sử, các đơn vị (phiên tương tác người dùng) không cần được phân ngẫu nhiên; tất cả thuật toán có thể được đánh giá trên tất cả người dùng.

8.5.2 Thiết kế bán thí nghiệm (Quasi-Experimental Design)

Khác với thí nghiệm thực sự, các đối tượng không được phân ngẫu nhiên. Điều này có thể tạo ra thiên lệch không kiểm soát, ví dụ: người dùng có xu hướng mua mạnh hơn cũng tự chọn sử dụng hệ thống gợi ý. Do đó, kết quả của các thiết kế bán thí nghiệm cần được giải thích cẩn trọng.

8.5.3 Thiết kế phi thí nghiệm (Nonexperimental Design)

Bao gồm nghiên cứu định lượng và định tính:

- **Định lượng (Quantitative):** đo lường số liệu các khía cạnh của đối tượng, ví dụ: đánh giá sự hữu ích của hệ thống gợi ý trên thang Likert 7 điểm, hoặc thời gian xem trang web.
- **Định tính (Qualitative):** phỏng vấn mở, ghi lại quá trình “think-aloud” khi người dùng tương tác, hoặc nhóm tập trung để tìm hiểu động cơ sử dụng hệ thống gợi ý.

Một thiết kế phi thí nghiệm quan trọng là **nghiên cứu theo chiều dài thời gian (Longitudinal research)**, trong đó đối tượng được quan sát liên tục theo thời gian, ví dụ: tác động của hệ thống gợi ý đến giá trị vòng đời khách hàng. Ngược lại, **nghiên cứu cắt ngang (Cross-sectional)** phân tích các biến đo cùng lúc trong các nhóm khác nhau.

8.5.4 Nghiên cứu tình huống (Case Study)

Các nghiên cứu tình huống tập trung vào trả lời câu hỏi *how* và *why*, kết hợp các phương pháp định lượng và định tính để điều tra hiện tượng trong bối cảnh thực tế, ví dụ: vai trò của hệ thống gợi ý trong thành công của Amazon.com.

8.6 Thiết lập đánh giá (Evaluation Settings)

Hai loại thiết lập cơ bản:

- **Phòng thí nghiệm (Lab Study):** tạo môi trường nghiên cứu riêng, dễ kiểm soát các biến ngoại lai nhưng có thể làm người tham gia không phản ánh hành vi thực.
- **Thực địa (Field Study):** thực hiện trong môi trường thực, người dùng tự động có động lực sử dụng hệ thống, nhưng nhà nghiên cứu ít kiểm soát hơn.

8.7 Các độ đo phổ biến trong đánh giá hệ gợi ý dựa trên dữ liệu lịch sử

8.7.1 Đánh giá hệ thống gợi ý theo truy xuất thông tin

Thiết lập ground truth

Trong đánh giá hệ thống gợi ý, ground truth được xác định bởi chuyên gia hoặc dữ liệu thực tế. Mọi items tốt (relevant items) được ghi nhận, và tất cả các items được hệ thống gợi ý đều được xét đến trong đánh giá.

Precision và Recall

Dề xuất có thể xem như một công việc truy xuất thông tin (Information Retrieval). Các độ đo cơ bản gồm:

Precision: Tỉ lệ phần tử có liên quan được truy xuất trên tổng số phần tử được truy xuất:

$$\text{Precision} = \frac{|\text{Relevant Items} \cap \text{Retrieved Items}|}{|\text{Retrieved Items}|}.$$

Ví dụ: Tỷ lệ các bộ phim được đề xuất mà thực sự được quan tâm bởi người dùng.

Recall: Tỉ lệ phần tử có liên quan được truy xuất trên tổng số phần tử có liên quan:

$$\text{Recall} = \frac{|\text{Relevant Items} \cap \text{Retrieved Items}|}{|\text{Relevant Items}|}.$$

Ví dụ: Tỷ lệ các bộ phim tốt (relevant) mà hệ thống đã gợi ý cho người dùng.

Precision vs. Recall

Khi điều chỉnh hệ thống gợi ý để tăng Precision, Recall thường giảm, và ngược lại. Sự cân bằng giữa hai chỉ số này là vấn đề quan trọng trong đánh giá hiệu quả của hệ thống.

Độ đo F1 và F_β

Để kết hợp Precision và Recall thành một giá trị duy nhất, sử dụng độ đo F1:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Độ đo F_β cho phép đặt trọng số khác nhau cho Recall với hệ số β :

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}.$$

Với $\beta = 1$, F_β trở về F1, đặt trọng số bằng nhau cho Precision và Recall.

8.7.2 Độ đo AUC (Area Under the ROC Curve)

Khái niệm

AUC là diện tích dưới đường cong ROC (Receiver Operating Characteristic Curve). ROC thể hiện mối quan hệ giữa:

- **True Positive Rate (TPR):** Tỉ lệ phần tử liên quan được dự đoán đúng.
- **False Positive Rate (FPR):** Tỉ lệ phần tử không liên quan bị dự đoán sai là liên quan.

AUC đo lường khả năng phân biệt giữa các item liên quan (positive) và không liên quan (negative) trong dự đoán của hệ thống.

Công thức

Cho một tập dự đoán ranking hoặc xác suất, AUC được tính bằng:

$$\text{AUC} = \frac{\sum_{i \in \text{Positive}} \sum_{j \in \text{Negative}} \mathbf{1}(\hat{r}_i > \hat{r}_j)}{|\text{Positive}| \cdot |\text{Negative}|},$$

trong đó:

- \hat{r}_i và \hat{r}_j là giá trị dự đoán (score hoặc ranking) cho các item positive và negative.
- $\mathbf{1}(\cdot)$ là hàm chỉ báo (indicator function), nhận giá trị 1 nếu biểu thức đúng, 0 nếu sai.

Ý nghĩa

- AUC = 1: Hệ thống gợi ý phân loại hoàn hảo, tất cả item liên quan xếp trên item không liên quan.
- AUC = 0.5: Hệ thống gợi ý ngẫu nhiên, không có khả năng phân biệt.
- $0.5 < \text{AUC} < 1$: Hiệu quả gợi ý tốt hơn ngẫu nhiên, càng gần 1 càng tốt.

Ứng dụng trong gợi ý

- Dánh giá các mô hình dự đoán ranking item dựa trên xác suất tương tác.
- So sánh mô hình khi dữ liệu có imbalance (nhiều item negative hơn positive).
- Thường kết hợp với Precision, Recall hoặc NDCG để có đánh giá toàn diện.

8.7.3 Độ đo MRR (Mean Reciprocal Rank)

Khái niệm

Mean Reciprocal Rank (MRR) đánh giá hiệu quả của hệ thống gợi ý dựa trên vị trí xếp hạng (rank) của item liên quan đầu tiên trong danh sách gợi ý. MRR đặc biệt phù hợp khi người dùng quan tâm đến item quan trọng nhất trong danh sách.

Công thức

Cho tập người dùng U , với mỗi người dùng u có danh sách gợi ý đã xếp hạng, MRR được tính như sau:

$$\text{MRR} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\text{rank}_u},$$

trong đó rank_u là vị trí xếp hạng (rank) của item liên quan đầu tiên (relevant item) trong danh sách gợi ý của người dùng u .

Ví dụ

Nếu danh sách gợi ý cho người dùng có item liên quan xuất hiện ở vị trí thứ 3, vị trí thứ 1, và vị trí thứ 2 cho các người dùng khác nhau, thì MRR được tính trung bình các giá trị:

$$\text{MRR} = \frac{1}{3} \left(\frac{1}{3} + \frac{1}{1} + \frac{1}{2} \right) = \frac{11}{18} \approx 0.611.$$

Ý nghĩa

- MRR phản ánh mức độ hiệu quả của hệ thống trong việc đưa các item quan trọng lên đầu danh sách.
- Giá trị MRR càng cao (gần 1) càng tốt, biểu thị rằng item liên quan xuất hiện càng gần đầu danh sách.
- MRR thường kết hợp với các độ đo ranking khác như NDCG để đánh giá tổng thể chất lượng gợi ý.

8.7.4 Độ đo NDCG (Normalized Discounted Cumulative Gain)

Khái niệm

NDCG đánh giá chất lượng xếp hạng của danh sách gợi ý, đặc biệt khi các item có mức độ liên quan khác nhau. Nó giảm trọng số của các item xuất hiện ở vị trí thấp hơn trong danh sách, nhấn mạnh rằng các item quan trọng nên xuất hiện sớm.

Công thức

Đầu tiên, tính Discounted Cumulative Gain (DCG) cho danh sách gợi ý L :

$$\text{DCG}@k = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i + 1)},$$

trong đó rel_i là độ liên quan (relevance) của item tại vị trí i , k là độ dài danh sách xét.

Tiếp theo, tính Ideal DCG (IDCG) – DCG tối ưu khi các item liên quan xếp đầu:

$$\text{IDCG}@k = \sum_{i=1}^k \frac{2^{rel_i^*} - 1}{\log_2(i + 1)},$$

với rel_i^* là thứ tự liên quan giảm dần.

Cuối cùng, NDCG được chuẩn hóa:

$$\text{NDCG}@k = \frac{\text{DCG}@k}{\text{IDCG}@k}.$$

Ý nghĩa

- NDCG tập trung vào vị trí đầu danh sách gợi ý.

- Giá trị NDCG dao động từ 0 đến 1, càng gần 1 càng tốt.
- Phù hợp khi item có mức độ liên quan khác nhau (ví dụ: đánh giá từ 1–5 sao).

8.7.5 Độ đo MAP (Mean Average Precision)

Khái niệm

MAP đánh giá hiệu quả xếp hạng của hệ thống gợi ý bằng cách tính trung bình Precision tại các vị trí có item liên quan, sau đó lấy trung bình trên tất cả người dùng. MAP nhấn mạnh việc gợi ý các item liên quan xuất hiện sớm trong danh sách.

Công thức

Cho mỗi người dùng u , tính Average Precision (AP):

$$\text{AP}_u = \frac{1}{|\text{Rel}_u|} \sum_{k=1}^n P_u(k) \cdot \text{rel}_u(k),$$

trong đó:

- $P_u(k)$ là Precision tại vị trí k .
- $\text{rel}_u(k) = 1$ nếu item tại vị trí k là liên quan, 0 nếu không.
- $|\text{Rel}_u|$ là tổng số item liên quan cho người dùng u .

MAP tính trung bình AP trên tất cả người dùng:

$$\text{MAP} = \frac{1}{|U|} \sum_{u \in U} \text{AP}_u.$$

Ý nghĩa

- MAP kết hợp cả chất lượng xếp hạng và việc xuất hiện sớm các item liên quan.
- Giá trị MAP càng cao, hệ thống càng chính xác trong việc gợi ý các item liên quan đầu danh sách.
- Thường kết hợp với Precision@k, Recall@k, NDCG@k để đánh giá toàn diện.

Tóm tắt

- Precision, Recall, F1: đánh giá dựa trên độ chính xác và khả năng truy xuất.
- AUC: đánh giá khả năng phân biệt item liên quan và không liên quan.
- MRR: đánh giá xếp hạng item liên quan đầu tiên.
- NDCG: đánh giá chất lượng ranking theo vị trí và độ liên quan.
- MAP: đánh giá trung bình precision tại các vị trí có item liên quan.

8.7.6 Ví dụ minh họa các độ đo đánh giá

Xét 3 người dùng (U1, U2, U3) và 5 items (I1–I5) với dữ liệu ground truth và danh sách gợi ý như sau:

User	Ground truth (Relevant items)	Recommended list
U1	{I1, I2}	[I1, I3, I2, I4, I5]
U2	{I3}	[I2, I3, I1, I5, I4]
U3	{I4, I5}	[I5, I4, I2, I1, I3]

Bảng 8.1: Dữ liệu minh họa cho 3 users và 5 items

1. Precision, Recall và F1

Precision@5:

$$\text{Precision} = \frac{|\text{Relevant} \cap \text{Recommended}|}{5}$$

- U1: $2/5 = 0.4$
- U2: $1/5 = 0.2$
- U3: $2/5 = 0.4$

Recall@5:

$$\text{Recall} = \frac{|\text{Relevant} \cap \text{Recommended}|}{|\text{Relevant}|}$$

- U1: $2/2 = 1.0$
- U2: $1/1 = 1.0$
- U3: $2/2 = 1.0$

F1:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- U1: $2 * (0.4 * 1) / (0.4 + 1) \approx 0.571$
- U2: $2 * (0.2 * 1) / (0.2 + 1) \approx 0.333$
- U3: $2 * (0.4 * 1) / (0.4 + 1) \approx 0.571$

F1 trung bình: ≈ 0.492

2. MRR (Mean Reciprocal Rank)

$$\text{MRR} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\text{rank of first relevant item}}$$

- U1: $1/1 = 1$
- U2: $1/2 = 0.5$
- U3: $1/1 = 1$

$$\text{MRR} = (1 + 0.5 + 1)/3 = 0.833$$

3. NDCG@5

$$\text{DCG} = \sum_{i=1}^5 \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

- U1: relevant tại pos1, pos3
 $\text{DCG} = 1/\log_2(1 + 1) + 1/\log_2(3 + 1) = 1 + 0.5 = 1.5$
 $\text{IDCG} = 1/\log_2(1 + 1) + 1/\log_2(2 + 1) \approx 1.631$
 $\text{NDCG} = 0.920$
- U2: relevant tại pos2
 $\text{DCG} = 1/\log_2(2 + 1) \approx 0.631$
 $\text{IDCG} = 1$
 $\text{NDCG} = 0.631$

- U3: relevant tại pos1, pos2
 $DCG = 1 + 0.6309 = 1.631$
 $IDCG = 1 + 0.6309 = 1.631$
 $NDCG = 1.0$

NDCG trung bình xấp xỉ 0.850

4. MAP (Mean Average Precision)

$$AP_u = \frac{1}{|\text{Rel}_u|} \sum_{k=1}^n P_u(k) \cdot \text{rel}_u(k)$$

- U1: relevant tại pos1, pos3
 $AP = (1 + 2/3)/2 = 0.833$
- U2: relevant tại pos2
 $AP = 1/2 = 0.5$
- U3: relevant tại pos1, pos2
 $AP = (1 + 1)/2 = 1$

$$MAP = (0.833 + 0.5 + 1)/3 = 0.778$$

5. Tóm tắt các độ đo

Metric	Value
Precision@5	0.333
Recall@5	1.0
F1	0.492
MRR	0.833
NDCG@5	0.850
MAP	0.778

Bảng 8.2: Kết quả các độ đo cho ví dụ 3 users và 5 items

Chương 9

ỨNG DỤNG DEEP LEARNING: GỢI Ý TIN TỨC

9.1	Hệ thống gợi ý tin tức và học sâu	143
9.2	Mô hình gợi ý đa góc nhìn MAML [27]	149
9.3	Mô hình chú ý cá nhân hóa NPA [28]	151
9.4	Mô hình gợi ý với cơ chế tự chú ý: NRMS [29]	154
9.5	Mô hình dựa gợi ý dựa trên biểu diễn bộ nhớ dài hạn và ngắn hạn LSTUR [1]	156
9.6	So sánh các mô hình MAML, NRMS, NPA và LSTUR	159

9.1 Hệ thống gợi ý tin tức và học sâu

Trong kỷ nguyên thông tin số, người dùng thường xuyên phải đối mặt với tình trạng quá tải thông tin khi tiếp cận các nền tảng trực tuyến. Hệ thống gợi ý tin tức (*news recommender system* – NRS) ra đời nhằm hỗ trợ người dùng nhanh chóng tìm thấy những bài báo phù hợp với sở thích và nhu cầu của mình [30]. Tuy nhiên, đặc thù của tin tức (tính thời sự, vòng đời ngắn, khối lượng bài viết mới rất lớn) khiến việc thiết kế hệ thống gợi ý gặp nhiều thách thức, bao gồm: (i) dữ liệu huấn luyện hạn chế, (ii) sở thích người dùng thay đổi nhanh, và (iii) thiếu thông tin đánh giá tường minh như *rating*. Trong bối cảnh này, học sâu (*deep learning*) được ứng dụng để học biểu diễn ngữ nghĩa của văn bản cũng như mô hình hóa hành vi người dùng, từ đó nâng cao độ chính xác gợi ý. Các mạng nơ-ron như CNN, RNN, GNN, cùng với các

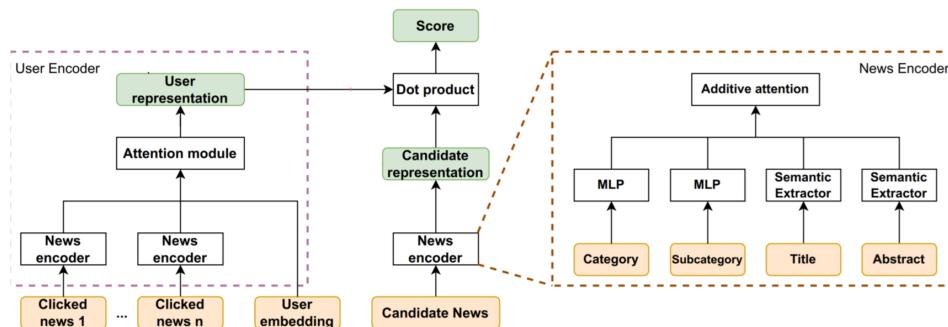
kiến trúc hiện đại dựa trên *attention* và Transformer (BERT, GPT), đã chứng minh hiệu quả vượt trội trong nhiều nghiên cứu [28, 29, 1].

9.1.1 Tổng quan về hệ thống gợi ý tin tức

Khác với hệ thống gợi ý sản phẩm hay phim, NRS phải xử lý luồng nội dung luôn cập nhật và hành vi người dùng biến động nhanh chóng [27]. Hầu hết các hệ thống gợi ý tin tức hiện đại dựa trên dữ liệu hành vi nhấp chuột (*click-stream*) và khai thác trực tiếp nội dung bài báo (tiêu đề, trích dẫn, thể loại, thực thể). Deep learning giúp:

- trích xuất đặc trưng ngữ nghĩa từ văn bản bằng CNN,
- nắm bắt chuỗi hành vi dài/ngắn hạn bằng RNN hoặc GRU/LSTM,
- mô hình hóa quan hệ phức tạp qua GNN,
- và biểu diễn ngữ cảnh ngữ nghĩa sâu bằng *self-attention* và Transformer.

Những kỹ thuật này vượt trội so với phương pháp truyền thống dựa trên lọc cộng tác hoặc gợi ý dựa trên nội dung đơn giản.



Hình 9.1: Minh họa kiến trúc tổng quan của hệ thống gợi ý tin tức dựa trên deep learning.

9.1.2 Kiến trúc tổng quát của hệ thống gợi ý học sâu

Một hệ thống gợi ý tin tức sử dụng học sâu thường bao gồm ba thành phần chính [30] (Hình 9.1):

1. **News encoder:** xây dựng biểu diễn vector của tin tức từ các thành phần như tiêu đề, trích dẫn, nội dung, thể loại và thực thể. CNN hoặc Transformer thường được dùng để trích xuất ngữ nghĩa.
2. **User encoder:** tạo biểu diễn người dùng dựa trên lịch sử đọc tin. Có thể sử dụng RNN/GRU để mô hình hóa chuỗi hành vi, hoặc attention để chọn lọc tin quan trọng.
3. **Click predictor:** dự đoán xác suất người dùng click vào một tin ứng viên, thường dựa trên tích vô hướng giữa vector người dùng và tin tức hoặc thông qua MLP.

Kiến trúc “news encoder – user encoder – click predictor” đã trở thành khung chuẩn cho các nghiên cứu gần đây.

9.1.3 Các mô hình điển hình

Trong giai đoạn 2019–2020, nhiều mô hình học sâu tiêu biểu đã được đề xuất cho bài toán gợi ý tin tức:

NPA (Neural News Recommendation with Personalized Attention). Mô hình NPA [28] sử dụng cơ chế *personalized attention* để học biểu diễn tin tức và người dùng. News encoder dùng CNN để xử lý tiêu đề, trong khi user encoder kết hợp các tin đã click với attention cá nhân hóa, trong đó embedding ID người dùng đóng vai trò truy vấn. Điểm mạnh: một từ hoặc tin tức có thể mang trọng số khác nhau tùy từng người dùng.

NRMS (Neural News Recommendation with Multi-Head Self-Attention). NRMS [29] áp dụng multi-head self-attention để mô hình hóa quan hệ ngữ nghĩa giữa các từ trong tiêu đề và giữa các tin trong lịch sử đọc. Kết hợp thêm *additive attention*, NRMS đạt kết quả tốt trên nhiều tập dữ liệu thực tế.

LSTUR (Long- and Short-Term User Representations). LSTUR [1] khai thác đồng thời sở thích dài hạn (qua embedding người dùng) và ngắn hạn (qua GRU trên lịch sử gần đây). News encoder xử lý tiêu đề và category bằng attention. Hai phần long-term và short-term được kết hợp để tạo vector người dùng cuối cùng, giúp hệ thống nắm bắt tốt hơn sự thay đổi theo thời gian.

NAML (Neural News Recommendation with Attentive Multi-View Learning). NAML [27] tận dụng nhiều “góc nhìn” của một bài báo: tiêu đề, nội dung và thẻ loại. Attention được áp dụng ở cả mức từ và mức view. Trong user encoder, attention tiếp tục được dùng để chọn lọc tin tức quan trọng nhất trong lịch sử.

9.1.4 Tập dữ liệu MIND

MIND (*Microsoft News Dataset*) [30] là bộ dữ liệu quy mô lớn, công bố năm 2020:

- hơn 160.000 bài báo tiếng Anh,
- 15 triệu impression từ 1 triệu người dùng.

Mỗi bài báo bao gồm: `news_id`, `category`, `subcategory`, `title`, `abstract`, `url`, `title_entities`, `abstract_entities`.

Dữ liệu hành vi được lưu trong `behaviors.tsv`, mỗi hàng gồm: Impression ID, User ID (tên danh), thời gian, lịch sử click, danh sách tin ứng viên và nhãn 0/1 (click/no-click). MIND có cả bản đầy đủ và bản MIND-small (50.000 người dùng) cho thí nghiệm nhanh. Đây hiện là tập dữ liệu chuẩn cho nghiên cứu gợi ý tin tức.

9.1.5 Tiền xử lý dữ liệu MIND và thống kê

Để sử dụng MIND, các bước tiền xử lý phổ biến gồm:

1. **Làm sạch văn bản:** loại bỏ tin trùng lặp, gỡ HTML, chuẩn hóa chữ, tokenization, stopwords, stemming/lemmatization.
2. **Xử lý thực thể:** sử dụng embedding từ file `entity_embedding.vec`.
3. **Mã hóa thuộc tính:** category, subcategory, user ID và news ID thành số nguyên liên tục.
4. **Mở rộng impressions:** chuyển mỗi impression thành nhiều mẫu (user, candidate, label).
5. **Embedding ngôn ngữ:** Word2Vec, GloVe, hoặc fine-tuning BERT/GPT.

- Impression ID: ID của một hiển thị.
- User ID: ID (tên danh) của người dùng.
- Time: Thời gian hiển thị với định dạng "MM/DD/YYYY HH:MM:SS AM/PM".
- History: Lịch sử nhấp chuột trên tin tức (danh sách các ID của tin tức đã được nhấp) của người dùng trước hiển thị này (được sắp xếp theo thời gian)
- Impressions: Danh sách tin tức hiển thị trong impression và hành vi nhấp chuột của người dùng (1-click, 0-no click) (đã được xáo trộn)

Impression ID:	1
User ID:	U134050
Time:	11/15/2019 8:55:22 AM
History:	N12246 N128820 N119226 N4065 N67770 N33446 N103285 N99640 N106837 N76775 N129444 N120026 N115495 N86141 N98680 N108013 N121782 N30276 N93576 N123743 N48804 N79909 N68728 N71356 N102447 N33846 N83623 N97597 N59416 N1014 N20285
Impressions:	N91737-0 N30206-0 N54368-0 N117802-0 N18190-0 N122944-0 N69938-1 N18356-0 N123209-0 N46894-0 N29160-0 N49978-0 N89764-0 N30582-0 N58465-0 N35304-0

Hình 9.2: Minh họa file behaviors.tsv.

News ID:	N1387
Category:	entertainment
SubCategory:	gaming
Title:	Best PS4 games 2019: play great PlayStation 4 games
Abstract:	These are the best PS4 games available today, from Sony's awesome exclusives to third-party must plays
URL:	https://assets.msn.com/labs/mind/AACkFyc.html
Title Entities:	[{"Label": "PlayStation 4", "Type": "J", "WikidataId": "Q5014725", "Confidence": 1.0, "OccurrenceOffsets": [5, 32], "SurfaceForms": ["PS4", "PlayStation 4"]}]
Abstract Entities:	[{"Label": "PlayStation 4", "Type": "J", "WikidataId": "Q5014725", "Confidence": 1.0, "OccurrenceOffsets": [19], "SurfaceForms": ["PS4"]}, {"Label": "Sony", "Type": "O", "WikidataId": "Q41187", "Confidence": 0.936, "OccurrenceOffsets": [51], "SurfaceForms": ["Sony"]}]]

Hình 9.3: Minh họa file news.tsv.

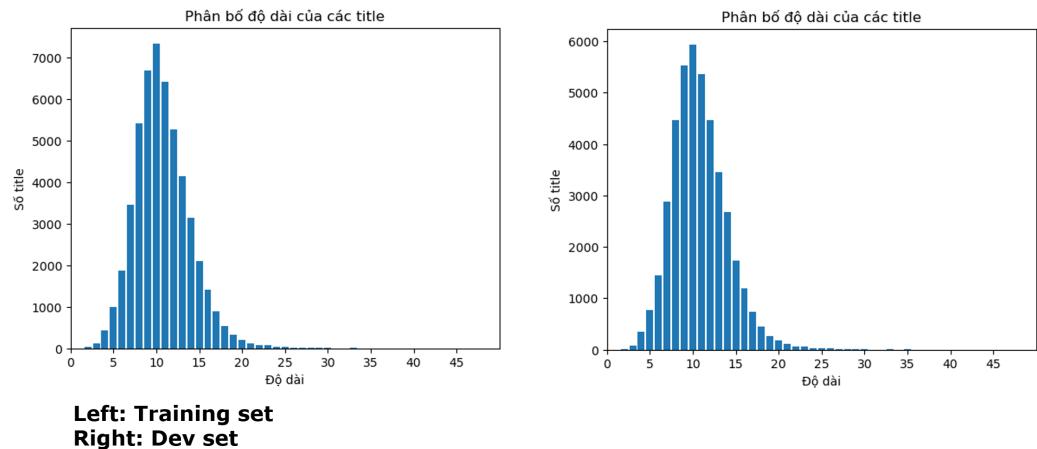
- **Tổng quan bộ dữ liệu MIND small (unique):**

# users	94,057	avg. #words per title	10.77
# news	65,238	avg. #words per abstract	35.53
# impressions	230,117	# sample (tổng số tin tức được hiển thị)	8,584,442
# category	18	# positive samples (tin tức được click)	347,727
# sub-category	270	# negative samples (tin tức không được click)	8,236,715

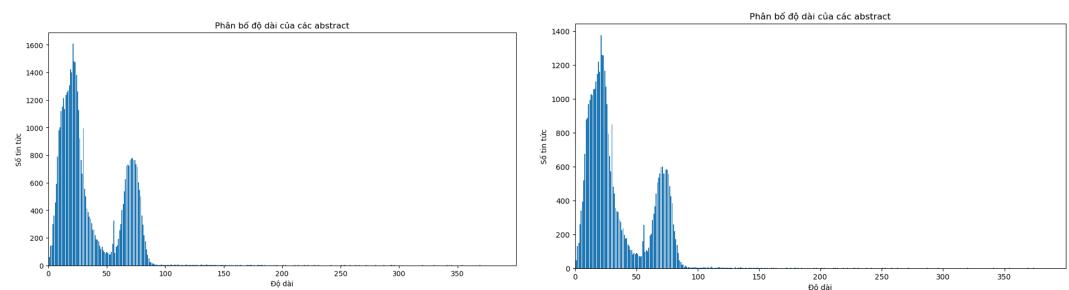
Hình 9.4: Thống kê tập Mind-small.

9.1.6 Xu hướng nghiên cứu và tiềm năng

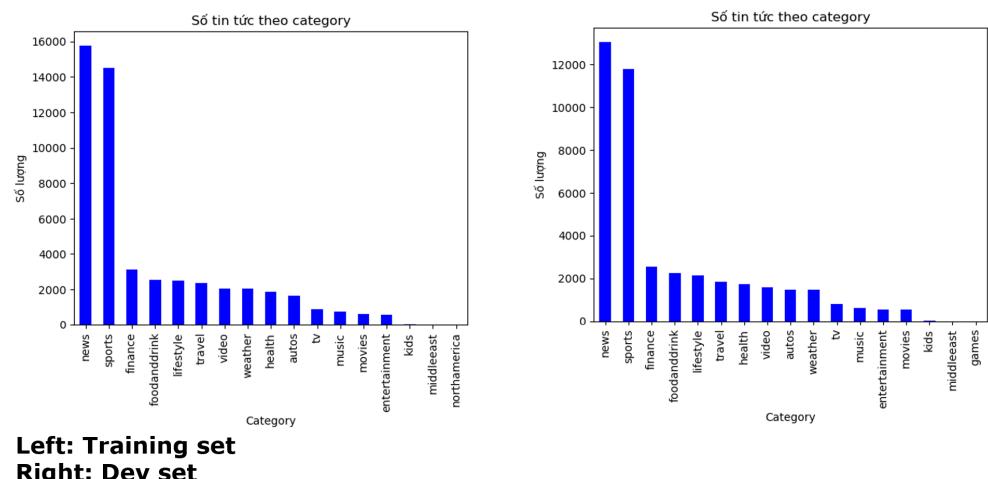
Nghiên cứu gần đây tập trung vào:



Hình 9.5: Phân bố độ dài của tiêu đề.



Hình 9.6: Phân bố độ dài của tóm tắt (abstract).

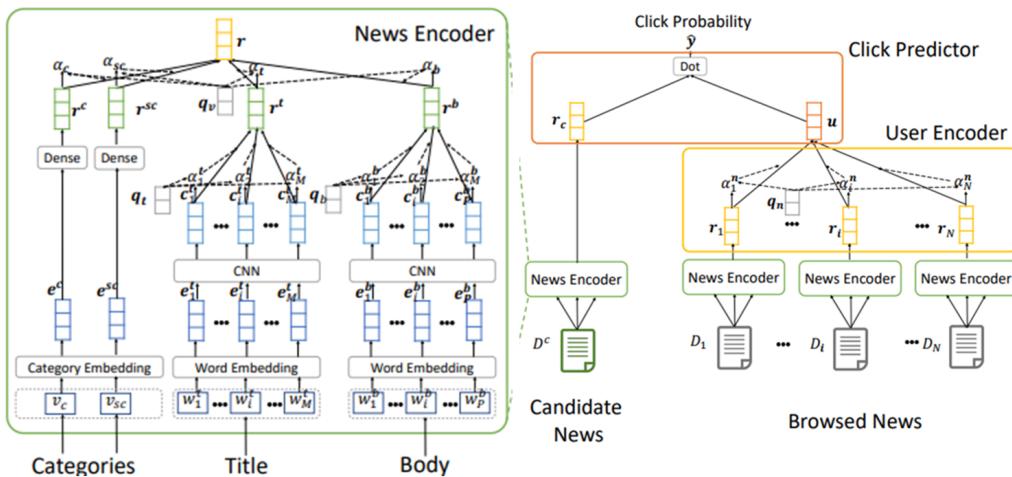


Hình 9.7: Số lượng news theo mỗi category.

- **Mã hóa nội dung sâu hơn:** LLMs (BERT, GPT), dữ liệu đa phương tiện (text + hình ảnh + video).
- **Mô hình người dùng nâng cao:** multi-interest, preference dynamics, meta-learning.
- **Phương pháp huấn luyện mới:** contrastive learning, causal inference.
- **Tối ưu tính toán:** knowledge distillation, huấn luyện hiệu quả cho LLM.
- **Trustworthiness:** fairness, explainability, federated learning để bảo vệ dữ liệu cá nhân.

Nhìn chung, lĩnh vực gợi ý tin tức đang phát triển mạnh mẽ nhờ sự kết hợp giữa NLP hiện đại và các kỹ thuật học sâu tiên tiến, hướng tới hệ thống cá nhân hóa, đáng tin cậy và thân thiện với người dùng.

9.2 Mô hình gợi ý đa góc nhìn MAML [27]



Hình 9.8: Kiến trúc tổng thể MAML [27].

9.2.1 Kiến trúc tổng thể

Mô hình MAML được thiết kế nhằm khai thác nhiều loại thông tin của một bài báo (*title*, *body*, *category*, *subcategory*) để tạo ra biểu diễn tin tức giàu ngữ nghĩa hơn. Kiến trúc mô hình gồm ba thành phần chính (Hình 9.9):

- **News Encoder:** ánh xạ mỗi bài báo thành vector đặc trưng $r \in \mathbb{R}^d$ thông qua các góc nhìn khác nhau.
- **User Encoder:** kết hợp các tin tức mà người dùng đã đọc để thu được biểu diễn người dùng $u \in \mathbb{R}^d$.
- **Click Predictor:** ước lượng xác suất một người dùng sẽ click vào một tin tức ứng cử dựa trên độ tương đồng giữa u và r_c .

9.2.2 Bộ mã hóa tin tức (News Encoder)

Mỗi bài báo bao gồm nhiều trường thông tin:

- **Tiêu đề và nội dung:** Chuỗi từ $\{w_1, w_2, \dots, w_T\}$ được ánh xạ thành vector nhúng từ $\{e_1, e_2, \dots, e_T\}$, sau đó qua CNN để trích xuất đặc trưng cục bộ. Cơ chế *word-level attention* được áp dụng để tập trung vào những từ quan trọng:

$$\alpha_i = \frac{\exp(q^\top h_i)}{\sum_{j=1}^T \exp(q^\top h_j)}, \quad r_{\text{text}} = \sum_{i=1}^T \alpha_i h_i, \quad (9.1)$$

trong đó h_i là biểu diễn từ sau CNN và q là vector tham số học được.

- **Chuyên mục và tiểu chuyên mục:** Được ánh xạ thành vector nhúng $r_{\text{cat}}, r_{\text{subcat}} \in \mathbb{R}^d$ thông qua một lớp embedding và fully-connected layer.

Cuối cùng, các góc nhìn (*title, body, category, subcategory*) được kết hợp bằng *view-level attention*:

$$\beta_v = \frac{\exp(p^\top r_v)}{\sum_{v'} \exp(p^\top r_{v'})}, \quad r = \sum_v \beta_v r_v, \quad (9.2)$$

trong đó r_v là biểu diễn từ mỗi góc nhìn và p là vector tham số học.

9.2.3 Bộ mã hóa người dùng (User Encoder)

Một người dùng được mô tả bởi tập các bài báo đã đọc $\{r_1, r_2, \dots, r_N\}$ (mỗi r_i được tính bởi News Encoder). Mô hình sử dụng *news-level attention* để gán trọng số khác nhau cho các tin tức:

$$\gamma_i = \frac{\exp(s^\top r_i)}{\sum_{j=1}^N \exp(s^\top r_j)}, \quad u = \sum_{i=1}^N \gamma_i r_i, \quad (9.3)$$

trong đó s là vector tham số học được, u là biểu diễn người dùng.

9.2.4 Bô dự đoán click (Click Predictor)

Cho một người dùng với biểu diễn u và một bài báo ứng cử với biểu diễn r_c , xác suất click được tính bằng:

$$\hat{y} = \sigma(u^\top r_c), \quad (9.4)$$

với $\sigma(\cdot)$ là hàm sigmoid.

9.2.5 Hàm mất mát (Loss Function)

Mô hình sử dụng kỹ thuật *negative sampling*. Với mỗi phiên duyệt tin, một bài báo được click là mẫu dương, và K bài báo không được click là mẫu âm. Xác suất lựa chọn được chuẩn hóa qua softmax:

$$P(c^+ | u) = \frac{\exp(u^\top r_{c^+})}{\exp(u^\top r_{c^+}) + \sum_{k=1}^K \exp(u^\top r_{c_k^-})}, \quad (9.5)$$

trong đó c^+ là bài báo được click, c_k^- là các bài báo âm.

Hàm mất mát cực đại hoá log-likelihood tương ứng:

$$\mathcal{L} = - \sum_{(u, c^+, \{c_k^-\})} \log P(c^+ | u). \quad (9.6)$$

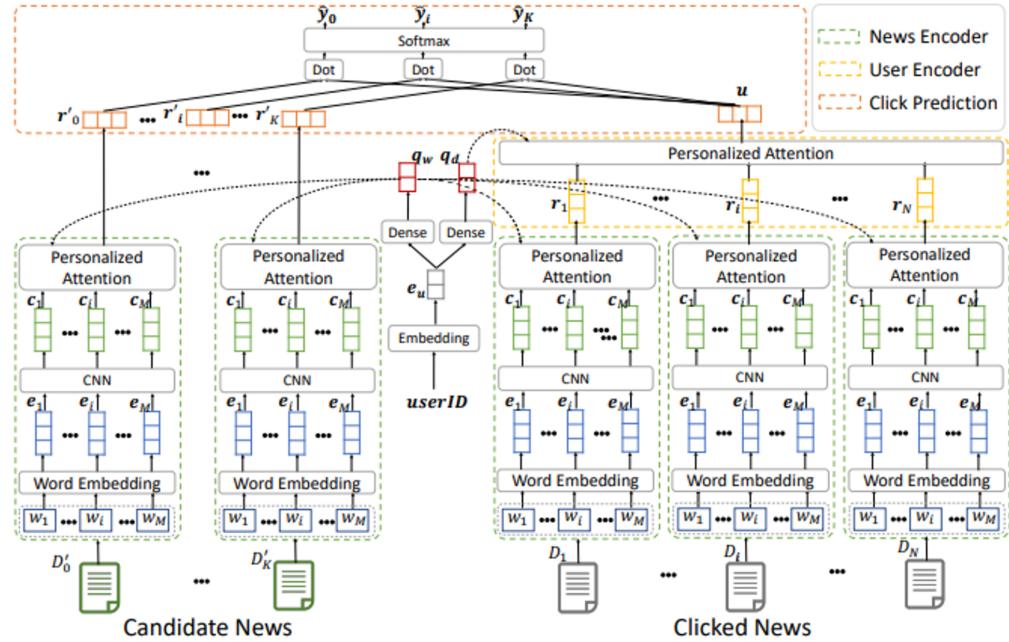
9.2.6 Tóm tắt

Với cơ chế *attention* nhiều cấp (từ, góc nhìn, tin tức), MAML có khả năng chọn lọc thông tin quan trọng ở cả mức độ chi tiết và tổng thể, từ đó tạo ra biểu diễn tin tức và người dùng giàu ngữ nghĩa, giúp nâng cao hiệu quả gọi ý tin tức cá nhân hóa.

9.3 Mô hình chú ý cá nhân hóa NPA [28]

9.3.1 Giới thiệu

Trong bài toán gợi ý tin tức, một thách thức lớn là sự đa dạng sở thích của người dùng. Các mô hình dựa trên nội dung trước đây (ví dụ NAML) thường sử dụng cơ chế attention chung cho mọi người dùng, dẫn đến việc chưa cá nhân hóa hoàn toàn. Mô hình **NPA (Neural News Recommendation with Personalized Attention)** được Wu et al. (2019b) [28] đề xuất nhằm giải quyết vấn đề này thông qua việc áp dụng *personalized attention* – tức là tham số attention được điều chỉnh riêng cho từng người dùng, thay vì dùng chung cho tất cả.



Hình 9.9: Kiến trúc tổng thể NPA [28].

9.3.2 Kiến trúc tổng thể

NPA gồm ba thành phần chính:

- **News Encoder:** ánh xạ mỗi bài báo thành vector đặc trưng r bằng CNN và attention cá nhân hóa theo người dùng.
- **User Encoder:** tổng hợp biểu diễn người dùng từ các bài báo đã đọc, cũng với personalized attention.
- **Click Predictor:** dự đoán xác suất click dựa trên độ tương đồng giữa u và r_c .

9.3.3 Bộ mã hóa tin tức (News Encoder)

Mỗi bài báo được biểu diễn bởi một chuỗi từ $\{w_1, w_2, \dots, w_T\}$, được ánh xạ thành embedding $\{e_1, e_2, \dots, e_T\}$. Chuỗi embedding này đi qua CNN để thu được các đặc trưng cục bộ h_i . Khác với NAML, cơ chế attention ở đây phụ thuộc vào từng người dùng:

$$\alpha_i^u = \frac{\exp((q_u)^\top h_i)}{\sum_{j=1}^T \exp((q_u)^\top h_j)}, \quad r = \sum_{i=1}^T \alpha_i^u h_i, \quad (9.7)$$

trong đó q_u là vector tham số riêng của người dùng u , học được thông qua một tầng embedding người dùng.

9.3.4 Bộ mã hoá người dùng (User Encoder)

Giả sử người dùng u đã đọc tập tin tức $\{r_1, r_2, \dots, r_N\}$. Tương tự như trên, personalized attention được dùng để gán trọng số cho các tin tức đã đọc:

$$\gamma_i^u = \frac{\exp((s_u)^\top r_i)}{\sum_{j=1}^N \exp((s_u)^\top r_j)}, \quad u = \sum_{i=1}^N \gamma_i^u r_i, \quad (9.8)$$

trong đó s_u là vector tham số người dùng, học từ embedding người dùng.

9.3.5 Bộ dự đoán click (Click Predictor)

Với biểu diễn người dùng u và bài báo ứng cử r_c , xác suất click được tính bằng:

$$\hat{y} = \sigma(u^\top r_c). \quad (9.9)$$

9.3.6 Hàm mất mát (Loss Function)

Tương tự như NAML, NPA sử dụng kỹ thuật *negative sampling*. Cho một bài báo được click c^+ và K bài báo không click c_k^- , xác suất dự đoán được định nghĩa:

$$P(c^+ | u) = \frac{\exp(u^\top r_{c^+})}{\exp(u^\top r_{c^+}) + \sum_{k=1}^K \exp(u^\top r_{c_k^-})}. \quad (9.10)$$

Hàm mất mát là log-likelihood âm:

$$\mathcal{L} = - \sum_{(u, c^+, \{c_k^-\})} \log P(c^+ | u). \quad (9.11)$$

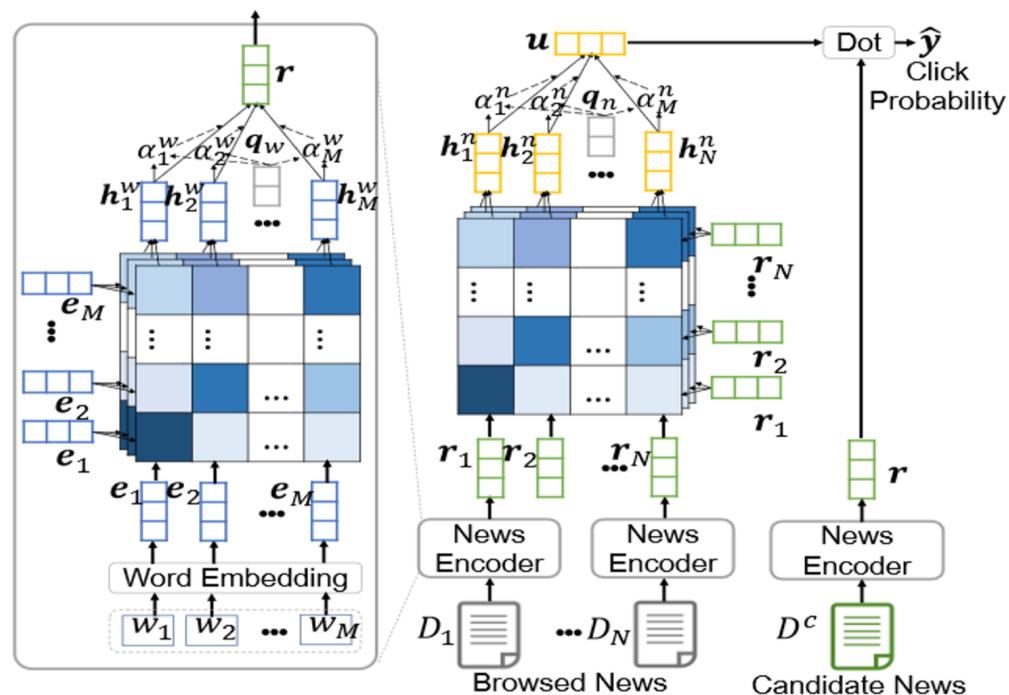
9.3.7 Tóm tắt

Điểm khác biệt chính của NPA so với NAML là việc sử dụng *personalized attention* dựa trên embedding người dùng. Cơ chế này cho phép mô hình học cách tập trung vào những từ và tin tức quan trọng khác nhau với từng người dùng, từ đó nâng cao khả năng cá nhân hóa trong gợi ý tin tức.

9.4 Mô hình gợi ý với cơ chế tự chú ý: NRMS [29]

9.4.1 Giới thiệu

Mô hình NRMS (*Neural News Recommendation with Multi-Head Self-Attention*) [29] được đề xuất nhằm khắc phục hạn chế của các phương pháp trước đó khi mã hóa tin tức và lịch sử đọc của người dùng. Khác với CNN hoặc RNN vốn chủ yếu nắm bắt ngữ cảnh cục bộ hoặc tuần tự, NRMS sử dụng *multi-head self-attention* để mô hình hóa quan hệ toàn cục giữa các từ trong tiêu đề tin tức và giữa các tin trong lịch sử người dùng. Điều này cho phép hệ thống học được sự phụ thuộc dài hạn và đa chiều trong dữ liệu, đồng thời tăng tính linh hoạt khi lựa chọn thông tin quan trọng cho quá trình gợi ý.



Hình 9.10: Kiến trúc tổng thể NRMS [27].

9.4.2 Kiến trúc mô hình

Kiến trúc NRMS gồm ba thành phần chính (Hình 9.10):

1. **News encoder:** mỗi tin tức (news) được biểu diễn bởi tiêu đề (title).

2. **User encoder:** biểu diễn người dùng được xây dựng từ các tin tức đã đọc trong lịch sử.
3. **Click predictor:** tính toán độ phù hợp giữa tin tức ứng viên và người dùng để dự đoán xác suất click.

News encoder. Giả sử một tin tức có tiêu đề gồm T từ. Mỗi từ w_t được ánh xạ thành vector embedding $e_t \in \mathbb{R}^d$. Ta thu được ma trận embedding:

$$E = [e_1, e_2, \dots, e_T] \in \mathbb{R}^{d \times T}.$$

Dể học ngữ cảnh giữa các từ, NRMS áp dụng *multi-head self-attention*. Với một head, attention được tính theo:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

trong đó $Q = EW_Q, K = EW_K, V = EW_V$, với $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$.

Multi-head self-attention với h head được biểu diễn:

$$\text{MultiHead}(E) = [H_1; H_2; \dots; H_h]W_O,$$

trong đó $H_i = \text{Attention}(EW_Q^i, EW_K^i, EW_V^i)$.

Kết quả sau self-attention được đưa vào một lớp *additive attention* để chọn lọc từ quan trọng:

$$\alpha_t = \frac{\exp(\tanh(We_t)^\top u)}{\sum_{j=1}^T \exp(\tanh(We_j)^\top u)}, \quad r = \sum_{t=1}^T \alpha_t e_t,$$

với u là vector ngữ cảnh học được. Vector r chính là biểu diễn tin tức.

User encoder. Giả sử một người dùng đã đọc N tin, mỗi tin có vector biểu diễn r_i . Ta có:

$$R = [r_1, r_2, \dots, r_N] \in \mathbb{R}^{d \times N}.$$

NRMS tiếp tục áp dụng multi-head self-attention để mô hình hóa quan hệ giữa các tin tức đã đọc, thu được ma trận $H \in \mathbb{R}^{d \times N}$.

Sau đó, một lớp additive attention được dùng để chọn tin tức quan trọng nhất cho sở thích người dùng:

$$\beta_i = \frac{\exp(\tanh(Wr_i)^\top u_u)}{\sum_{j=1}^N \exp(\tanh(Wr_j)^\top u_u)}, \quad u = \sum_{i=1}^N \beta_i r_i,$$

trong đó u là vector biểu diễn người dùng.

Click predictor. Với vector tin tức ứng viên r_c và vector người dùng u , xác suất click được tính bằng tích vô hướng:

$$\hat{y} = \sigma(u^\top r_c),$$

trong đó $\sigma(\cdot)$ là hàm sigmoid.

9.4.3 Hàm mất mát (Loss function)

NRMS được huấn luyện theo cách phân biệt giữa tin đã click (positive) và chưa click (negative). Cho một impression gồm một tin dương r^+ và K tin âm $\{r_1^-, \dots, r_K^-\}$, xác suất dự đoán được chuẩn hóa bằng softmax:

$$P(r^+|u) = \frac{\exp(u^\top r^+)}{\exp(u^\top r^+) + \sum_{k=1}^K \exp(u^\top r_k^-)}.$$

Hàm mất mát là *negative log-likelihood*:

$$\mathcal{L} = - \sum_{(u, r^+, \{r^-\})} \log P(r^+|u).$$

9.4.4 Tóm tắt

NRMS khai thác hiệu quả cơ chế multi-head self-attention ở cả cấp độ từ và cấp độ tin tức, giúp học được các mối quan hệ dài hạn và đa chiều. Nhờ đó, mô hình đạt hiệu suất cao hơn so với các phương pháp dựa trên CNN hoặc RNN truyền thống trong nhiều thí nghiệm trên tập dữ liệu MIND.

9.5 Mô hình dựa gợi ý dựa trên biểu diễn bộ nhớ dài hạn và ngắn hạn LSTUR [1]

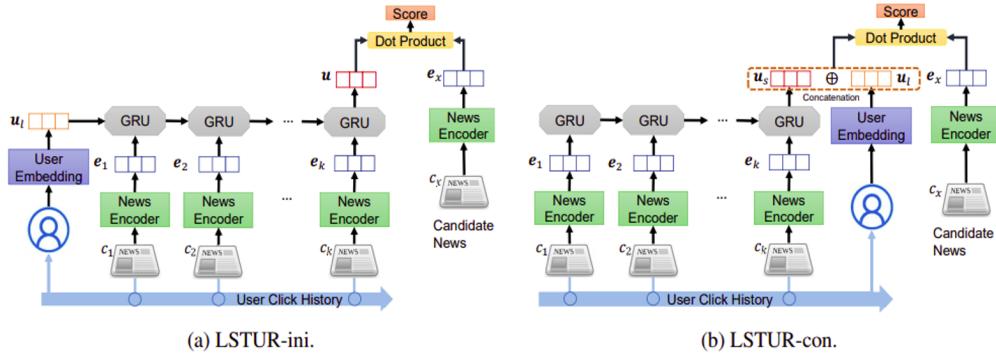
9.5.1 Giới thiệu

Trong gợi ý tin tức, hành vi người dùng thường bị chi phối đồng thời bởi *sở thích ngắn hạn* (các tin vừa đọc gần đây) và *sở thích dài hạn* (mẫu hành vi ổn định theo thời gian). Các mô hình trước đó như NAML hay NPA chủ yếu tập trung vào biểu diễn nội dung tin tức và cá nhân hóa qua attention, nhưng chưa khai thác rõ ràng mối quan hệ giữa ngắn hạn và dài hạn.

Mô hình **LSTUR (Neural News Recommendation with Short- and Long-term User Representations)** được An et al. (2019) [1] đề xuất nhằm giải

9.5 Mô hình dựa gợi ý dựa trên biểu diễn bộ nhớ dài hạn và ngắn hạn LSTUR [157]

quyết vấn đề này. Ý tưởng chính là kết hợp hai thành phần: (i) biểu diễn dài hạn của người dùng (long-term user embedding), và (ii) biểu diễn ngắn hạn thu được từ chuỗi tin tức gần đây mà người dùng đã đọc thông qua mạng tuần tự GRU.



Hình 9.11: Kiến trúc tổng thể LSTUR [1].

9.5.2 Kiến trúc tổng thể

LSTUR gồm ba thành phần (Hình 9.11):

- **News Encoder:** ánh xạ mỗi tin tức thành vector $r \in \mathbb{R}^d$.
- **User Encoder:** kết hợp sở thích ngắn hạn và dài hạn để sinh biểu diễn người dùng u .
- **Click Predictor:** dự đoán xác suất click dựa trên độ tương đồng giữa u và r_c .

9.5.3 Bộ mã hoá tin tức (News Encoder)

Tin tức được mã hoá tương tự như các mô hình trước (NAML, NPA). Một tin tức với chuỗi từ $\{w_1, \dots, w_T\}$ được ánh xạ sang embedding $\{e_1, \dots, e_T\}$, sau đó đi qua CNN để trích xuất đặc trưng ngữ nghĩa h_i . Attention được dùng để chọn lọc các từ quan trọng:

$$\alpha_i = \frac{\exp(q^\top h_i)}{\sum_{j=1}^T \exp(q^\top h_j)}, \quad r = \sum_{i=1}^T \alpha_i h_i, \quad (9.12)$$

trong đó q là vector tham số học được, r là biểu diễn tin tức.

9.5.4 Bô mã hoá người dùng (User Encoder)

Sở thích dài hạn. Mỗi người dùng u được gán một embedding tĩnh $e_u \in \mathbb{R}^d$, học được trong quá trình huấn luyện. Đây là đặc trưng dài hạn phản ánh sở thích tổng quát.

Sở thích ngắn hạn. Giả sử người dùng u đã đọc chuỗi tin tức gần đây $\{r_1, r_2, \dots, r_N\}$, trong đó r_i là vector tin tức từ News Encoder. Một mạng GRU được sử dụng để mô hình hoá phụ thuộc theo thời gian:

$$h_t = \text{GRU}(r_t, h_{t-1}), \quad u_s = h_N, \quad (9.13)$$

trong đó u_s là biểu diễn sở thích ngắn hạn.

Kết hợp. Biểu diễn người dùng cuối cùng được kết hợp từ u_s (ngắn hạn) và e_u (dài hạn). Trong LSTUR có hai biến thể:

$$u = \begin{cases} u_s + e_u, & \text{additive fusion,} \\ \text{concat}(u_s, e_u), & \text{concatenation fusion.} \end{cases} \quad (9.14)$$

9.5.5 Bô dự đoán click (Click Predictor)

Với biểu diễn người dùng u và tin ứng cử r_c , xác suất click được định nghĩa bởi:

$$\hat{y} = \sigma(u^\top r_c). \quad (9.15)$$

9.5.6 Hàm mất mát (Loss Function)

LSTUR cũng sử dụng *negative sampling* để huấn luyện. Cho một phiên duyệt với bài báo được click c^+ và K bài báo âm c_k^- , xác suất chuẩn hoá được tính bởi:

$$P(c^+ | u) = \frac{\exp(u^\top r_{c^+})}{\exp(u^\top r_{c^+}) + \sum_{k=1}^K \exp(u^\top r_{c_k^-})}. \quad (9.16)$$

Hàm mất mát cực đại log-likelihood:

$$\mathcal{L} = - \sum_{(u, c^+, \{c_k^-\})} \log P(c^+ | u). \quad (9.17)$$

9.5.7 Tóm tắt

LSTUR nổi bật ở việc mô hình hoá đồng thời sở thích ngắn hạn và dài hạn của người dùng. Nhờ đó, mô hình có khả năng vừa phản ánh xu hướng mới của người dùng (short-term), vừa duy trì tính ổn định theo thời gian (long-term), giúp cải thiện hiệu quả gợi ý tin tức trong bối cảnh hành vi người dùng thay đổi liên tục.

9.6 So sánh các mô hình MAML, NRMS, NPA và LSTUR

9.6.1 Tổng quan

Trong gợi ý tin tức dựa trên học sâu, nhiều mô hình đã được đề xuất nhằm cải thiện biểu diễn tin tức và người dùng. Bốn mô hình tiêu biểu gồm:

- **MAML (NAML, Wu et al. 2019a)**: học biểu diễn tin tức đa góc nhìn với attention nhiều cấp.
- **NRMS (Wu et al. 2019b)**: khai thác cơ chế self-attention để mô hình hoá quan hệ ngữ cảnh giữa các từ và tin tức.
- **NPA (Wu et al. 2019b)**: đưa vào personalized attention để cá nhân hoá việc chọn lọc từ và tin tức theo từng người dùng.
- **LSTUR (An et al. 2019)**: kết hợp sở thích ngắn hạn và dài hạn thông qua GRU và embedding người dùng.

9.6.2 So sánh chi tiết

Bảng 9.1 tóm tắt đặc điểm chính của từng mô hình.

9.6.3 Nhận xét

Các mô hình trên thể hiện sự tiến hoá từ việc tập trung vào *nội dung tin tức* (MAML, NRMS) sang *cá nhân hoá mạnh mẽ* (NPA) và cuối cùng là *kết hợp động giữa ngắn hạn và dài hạn* (LSTUR). MAML khai thác đa nguồn thông tin; NRMS nhấn mạnh quan hệ ngữ cảnh; NPA nâng cao mức độ cá nhân hoá; LSTUR cân bằng giữa ổn định và thích nghi. Tùy vào kịch bản ứng dụng (tin tức nhiều metadata, người dùng ít hay nhiều lịch sử, môi trường động), có thể chọn mô hình phù hợp hoặc kết hợp ưu điểm của nhiều mô hình để đạt hiệu quả tối ưu.

Bảng 9.1: So sánh các mô hình gợi ý tin tức tiêu biểu

Mô hình	Đặc điểm chính	Ưu điểm	Hạn chế
MAML (NAML)	Multi-view learning: tiêu đề, nội dung, category; attention nhiều cấp (word, view, news).	Biểu diễn tin tức phong phú; tận dụng nhiều loại thông tin.	Phụ thuộc nhiều vào metadata; tính cá nhân hoá chưa sâu.
NRMS	Dựa trên <i>multi-head self-attention</i> trong encoder tin tức và user encoder.	Học được quan hệ ngữ nghĩa dài hạn giữa các từ/tin tức; không cần CN-N/RNN.	Tính cá nhân hoá phụ thuộc nhiều vào lịch sử đọc, chưa khai thác embedding người dùng.
NPA	Personalized attention: tham số attention gắn với embedding người dùng.	Cá nhân hoá mạnh mẽ ở cả mức từ và mức tin tức.	Cần embedding người dùng rõ ràng; khó khai quát cho người dùng mới.
LSTUR	Kết hợp short-term (GRU trên lịch sử đọc) và long-term (user embedding).	Bắt được cả xu hướng tức thời và sở thích ổn định; hiệu quả trong môi trường động.	GRU có thể tốn kém tính toán; embedding dài hạn cần dữ liệu lớn để huấn luyện.

Chương 10

ỨNG DỤNG DEEP LEARNING: GỢI Ý DỰA TRÊN HỌC TƯƠNG PHẢN

10.1 Học tương phản (Contrastive Learning)	161
10.2 Học tương phản trong hệ gợi ý	167
10.3 Học tương phản cho gợi ý chuỗi CL4Rec [31]	169
10.4 Tăng cường dữ liệu trong học tương phản CoSeRec [18]	172
10.5 Các phương pháp mở rộng học tương phản trong hệ gợi ý	176

10.1 Học tương phản (Contrastive Learning)

Học tương phản là một trong những hướng tiếp cận nổi bật trong học máy hiện đại, đặc biệt trong bối cảnh học tự giám sát (*self-supervised learning*). Mục tiêu chính của học tương phản là xây dựng không gian biểu diễn (*representation space*) sao cho:

- Các điểm dữ liệu có tính chất hoặc ý nghĩa tương đồng sẽ được ánh xạ gần nhau.
- Các điểm dữ liệu khác biệt sẽ được đẩy ra xa nhau trong không gian biểu diễn.

Nhờ đặc điểm này, học tương phản cho phép mô hình học được biểu diễn giàu

ngữ nghĩa mà không cần dựa vào nhãn dữ liệu, từ đó tận dụng được nguồn dữ liệu khổng lồ chưa gắn nhãn. Điều này mở ra nhiều ứng dụng trong thị giác máy tính (CV), xử lý ngôn ngữ tự nhiên (NLP) và hệ thống gợi ý (recommender systems).

10.1.1 Quy trình học tương phản

Quy trình học tương phản thường bao gồm các bước chính sau:

1. Tạo dữ liệu huấn luyện

- **Cặp dương (positive pairs):** Là các cặp dữ liệu mang ý nghĩa tương đồng. Ví dụ: trong bài toán phân loại ảnh, hai ảnh thuộc cùng một lớp sẽ tạo thành cặp dương; hoặc trong học tự giám sát, một ảnh gốc và phiên bản được biến đổi bởi các phép *data augmentation* (xoay, cắt, thay đổi màu sắc, ...) cũng được coi là cặp dương.
- **Cặp âm (negative pairs):** Là các cặp dữ liệu khác biệt về ngữ nghĩa, chẳng hạn hai ảnh thuộc hai lớp khác nhau.

2. **Hàm mã hóa (Encoder)** Các dữ liệu đầu vào được đưa qua một mạng mã hóa (thường là CNN cho ảnh, Transformer cho văn bản) để học biểu diễn vector đặc trưng. Biểu diễn này sẽ là cơ sở để so sánh mức độ tương đồng hoặc khác biệt.
3. **Độ đo tương đồng** Để so sánh hai biểu diễn, ta sử dụng một độ đo tương đồng như *cosine similarity* hoặc khoảng cách Euclidean.
4. **Hàm mất mát tương phản (Contrastive Loss)** Hàm mất mát được thiết kế nhằm:

- Tối đa hóa độ tương đồng giữa các cặp dương.
- Tối thiểu hóa độ tương đồng giữa các cặp âm.

Các biến thể phổ biến bao gồm:

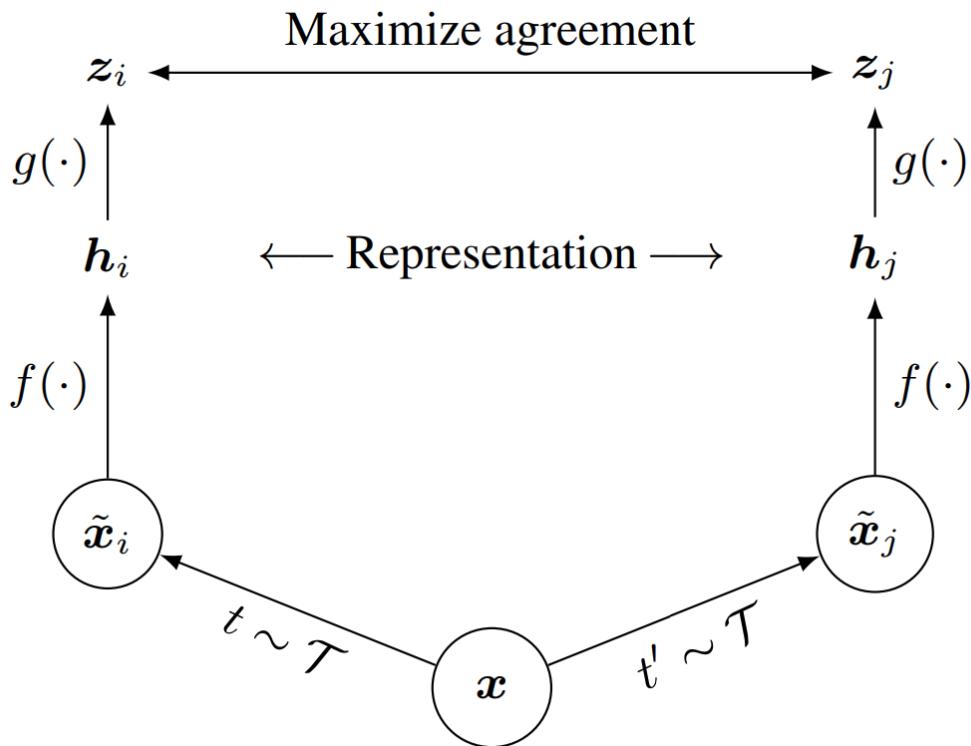
- **Triplet Loss:** dựa trên quan hệ giữa một điểm gốc (*anchor*), một điểm dương (*positive*) và một điểm âm (*negative*).
- **InfoNCE Loss:** sử dụng kỹ thuật phân biệt (*discriminative*) giữa một điểm dương và tập các điểm âm.

Ưu điểm và ứng dụng: Học tương phản đã chứng minh hiệu quả vượt trội trong việc học biểu diễn tổng quát và giàu ngữ nghĩa, từ đó cải thiện hiệu suất trên

nhiều bài toán hạ nguồn (*downstream tasks*) như phân loại, tìm kiếm, gợi ý, hoặc phân cụm dữ liệu. Điểm mạnh cốt lõi là khả năng khai thác dữ liệu không nhãn — một nguồn tài nguyên dồi dào nhưng trước đây chưa được tận dụng triệt để.

10.1.2 Mô hình SimCLR [5]

Một trong những mô hình học tương phản tiêu biểu trong thị giác máy tính là **SimCLR**, được đề xuất bởi Chen và cộng sự (2020) [5]. SimCLR chứng minh rằng một thiết kế đơn giản nhưng hợp lý có thể giúp học tự giám sát đạt kết quả ngang bằng hoặc thậm chí vượt qua học có giám sát trong nhiều tác vụ hạ nguồn.



Hình 10.1: Minh họa SimCLR.

Ý tưởng chính (Hình 10.1)

SimCLR xây dựng các cặp *positive* bằng cách áp dụng các phép biến đổi dữ liệu (*data augmentation*) lên cùng một ảnh gốc. Với mỗi ảnh $x \in \mathcal{X}$, ta áp dụng hai phép biến đổi ngẫu nhiên $t, t' \sim \mathcal{T}$ (với \mathcal{T} là phân phối của các phép biến đổi) để

thu được hai ảnh biến thể:

$$x_i^1 = t(x_i), \quad x_i^2 = t'(x_i).$$

Hai ảnh này tạo thành một *positive pair*. Tất cả các ảnh còn lại trong batch được coi là *negative samples*.

Kiến trúc mô hình

Pipeline của SimCLR bao gồm ba bước:

1. **Data augmentation:** tập hợp các phép biến đổi như cắt ngẫu nhiên, lật ngang, thay đổi màu sắc, làm mờ Gaussian.
2. **Encoder $f(\cdot)$:** ánh xạ ảnh đầu vào sang không gian biểu diễn. Ví dụ, dùng ResNet-50, ta thu được:

$$h_i^k = f(x_i^k), \quad k \in \{1, 2\}.$$

3. **Projection head $g(\cdot)$:** một mạng MLP nhỏ ánh xạ biểu diễn h sang không gian z nơi áp dụng hàm mất mát:

$$z_i^k = g(h_i^k) = g(f(x_i^k)).$$

Vector z_i^k thường được chuẩn hóa về độ dài 1 để thuận tiện cho việc tính *cosine similarity*.

Hàm mất mát

Với một batch gồm N ảnh gốc, sau data augmentation ta thu được $2N$ biểu diễn $\{z_1, z_2, \dots, z_{2N}\}$.

Cosine similarity. Độ tương đồng giữa hai vector u, v được định nghĩa:

$$\text{sim}(u, v) = \frac{u^\top v}{\|u\| \|v\|}.$$

InfoNCE loss. Với một cặp positive (z_i, z_j) , hàm mất mát được định nghĩa như sau:

$$\ell(i, j) = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)},$$

trong đó τ là **nhiệt độ** (*temperature parameter*) điều chỉnh độ nhạy trong phân phối softmax, và $\mathbf{1}_{[k \neq i]}$ là hàm chỉ báo để loại bỏ chính phần tử z_i trong mẫu so sánh.

Hàm mất mát cuối cùng cho toàn bộ batch là trung bình trên tất cả các cặp positive:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{2N} \ell(i, j(i)),$$

trong đó $j(i)$ là chỉ số của biến thể cùng nguồn gốc với z_i . Ví dụ, nếu z_{2k-1} và z_{2k} là hai biến thể sinh ra từ ảnh gốc x_k , thì $j(2k-1) = 2k$ và $j(2k) = 2k-1$.

Đóng góp chính

Các tác giả chỉ ra rằng hiệu năng của SimCLR phụ thuộc nhiều vào:

- Kích thước batch lớn (tăng số lượng negative samples).
- Data augmentation phong phú và mạnh.
- Projection head $g(\cdot)$ giúp cải thiện không gian biểu diễn so với chỉ dùng encoder.

Nhờ đó, biểu diễn học được từ SimCLR khi fine-tune cho nhiều tác vụ hạ nguồn (như phân loại ảnh trên ImageNet) đạt kết quả ngang bằng hoặc vượt trội so với huấn luyện có giám sát.

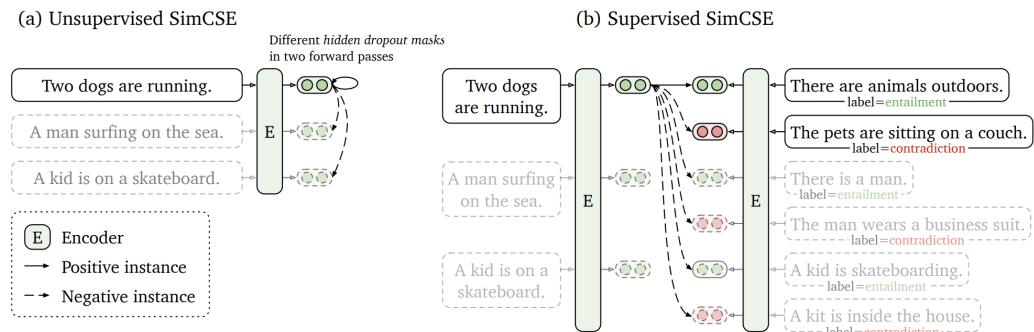
10.1.3 Mô hình SimCSE [7]

Trong xử lý ngôn ngữ tự nhiên, **SimCSE** (Simple Contrastive Sentence Embedding) được Gao và cộng sự (2021) [7] đề xuất như một phương pháp học biểu diễn câu bằng học tương phản. SimCSE đạt kết quả vượt trội trên các benchmark đánh giá ngữ nghĩa câu (Semantic Textual Similarity – STS) mà không cần thay đổi kiến trúc encoder gốc (BERT, RoBERTa, v.v.).

Ý tưởng chính (Hình 10.2)

Tương tự SimCLR trong thị giác máy tính, SimCSE xây dựng các *positive pairs* từ cùng một câu, và coi các câu khác trong batch là *negative samples*. Mô hình có hai biến thể:

- **SimCSE không giám sát (Unsupervised SimCSE):** dựa vào dropout ngẫu nhiên để tạo positive pairs.



Hình 10.2: Minh họa SimCSE.

- **SimCSE có giám sát (Supervised SimCSE):** sử dụng dữ liệu paraphrase từ tập NLI (SNLI, MNLI) để tạo positive và negative pairs.

Unsupervised SimCSE

Với một câu x_i , ta đưa nó qua encoder $f(\cdot)$ hai lần với dropout khác nhau, thu được hai biểu diễn:

$$h_i^1 = f(x_i; \text{dropout}_1), \quad h_i^2 = f(x_i; \text{dropout}_2).$$

Cặp (h_i^1, h_i^2) được coi là *positive pair*, còn tất cả các h_j với $j \neq i$ trong batch sẽ đóng vai trò *negative samples*.

Với một batch gồm N câu, ta thu được $2N$ embeddings $\{h_1, h_2, \dots, h_{2N}\}$. Hàm mất mát **InfoNCE** cho một anchor h_i được định nghĩa như:

$$\ell(i, j) = -\log \frac{\exp(\text{sim}(h_i, h_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(\text{sim}(h_i, h_k)/\tau)},$$

trong đó j là embedding sinh ra từ cùng một câu với h_i , $\text{sim}(\cdot, \cdot)$ là cosine similarity, và τ là tham số nhiệt độ.

Hàm mất mát cuối cùng cho batch:

$$\mathcal{L}_{\text{unsup}} = \frac{1}{2N} \sum_{i=1}^{2N} \ell(i, j(i)).$$

Supervised SimCSE

Trong thiết lập có giám sát, SimCSE sử dụng dữ liệu paraphrase từ các tập NLI. Với một câu gốc x_i (anchor), ta có:

- **Positive:** một câu paraphrase x_i^+ (cùng nghĩa).
- **Negatives:** các câu x_i^- khác trong cùng batch.

Sau khi mã hóa bằng encoder $f(\cdot)$, ta thu được biểu diễn h_i, h_i^+ và các negatives $\{h_k^-\}$. Hàm mất mát InfoNCE được định nghĩa:

$$\ell_{\text{sup}}(i) = -\log \frac{\exp(\text{sim}(h_i, h_i^+)/\tau)}{\exp(\text{sim}(h_i, h_i^+)/\tau) + \sum_{k=1}^K \exp(\text{sim}(h_i, h_k^-)/\tau)},$$

trong đó K là số lượng negatives trong batch.

Hàm mất mát cuối cùng cho batch là:

$$\mathcal{L}_{\text{sup}} = \frac{1}{N} \sum_{i=1}^N \ell_{\text{sup}}(i).$$

Đóng góp chính

SimCSE cho thấy rằng:

- Trong thiết lập **không giám sát**, chỉ cần dropout cũng đủ tạo ra những biến thể khác nhau của cùng một câu để huấn luyện contrastive learning.
- Trong thiết lập **có giám sát**, sử dụng cặp paraphrase làm positive pairs giúp mô hình học biểu diễn ngữ nghĩa tốt hơn.
- SimCSE đạt state-of-the-art trên nhiều tập benchmark STS, vượt trội so với các phương pháp học biểu diễn trước đó.

10.2 Học tương phản trong hệ gợi ý

Học tương phản đang ngày càng được áp dụng rộng rãi trong hệ gợi ý nhờ khả năng học được biểu diễn người dùng và sản phẩm giàu ngữ nghĩa từ dữ liệu tương tác quy mô lớn mà không cần nhãn bổ sung. Ý tưởng chính là đưa những người dùng có hành vi tương tác tương đồng lại gần nhau trong không gian biểu diễn, đồng thời đẩy xa những người dùng có hành vi khác biệt.

10.2.1 Nguyên lý cơ bản

Trong hệ gợi ý, mỗi người dùng u và sản phẩm i được ánh xạ sang một vector biểu diễn bằng các encoder (ví dụ: mạng embedding hoặc mạng neural sâu). Mục tiêu của học tương phản là:

- Với những người dùng có lịch sử tương tác tương đồng, vector biểu diễn từ encoder người dùng (*user encoder*) cần gần nhau trong không gian đặc trưng.
- Với những người dùng có hành vi khác biệt, biểu diễn cần cách xa nhau hơn.

Bằng cách này, không gian biểu diễn người dùng và sản phẩm phản ánh tốt hơn mối quan hệ ngữ nghĩa tiềm ẩn trong hành vi tương tác, từ đó cải thiện chất lượng gợi ý.

10.2.2 Tối ưu hoá biểu diễn người dùng qua các *views*

Một ý tưởng quan trọng trong học tương phản cho hệ gợi ý là tối ưu biểu diễn của cùng một người dùng theo các góc nhìn khác nhau (*views*). Thay vì chỉ dựa trên lịch sử tương tác ban đầu, ta tạo ra nhiều phiên bản (*views*) khác nhau của dữ liệu người dùng, sau đó buộc encoder học được các biểu diễn nhất quán cho cùng một người dùng.

Ký hiệu h_u^1 và h_u^2 là hai biểu diễn khác nhau của cùng một người dùng u từ hai views, hàm mất mát tương phản sẽ tối đa hóa độ tương đồng giữa (h_u^1, h_u^2) và tối thiểu hóa độ tương đồng với biểu diễn của các người dùng khác.

10.2.3 Các cách tạo views cho người dùng

Trong bối cảnh hệ gợi ý, nhiều phương pháp đã được đề xuất để xây dựng các views khác nhau của lịch sử tương tác người dùng. Ba cách tiếp cận điển hình gồm:

- **Cropping:** chọn ngẫu nhiên một đoạn con (subsequence) trong chuỗi lịch sử tương tác, coi như một view của người dùng.
- **Masking:** che khuất một phần các mục (items) trong lịch sử tương tác, để mô hình học cách biểu diễn người dùng dựa trên thông tin còn lại.
- **Reordering:** thay đổi ngẫu nhiên thứ tự một số tương tác trong chuỗi, nhằm tăng tính đa dạng và khả năng khai thác của encoder.

10.2.4 Hàm mất mát tương phản

Tương tự như SimCLR và SimCSE, hệ gợi ý thường áp dụng **InfoNCE loss**. Với một batch gồm N người dùng, mỗi người dùng u có hai views sinh ra embeddings h_u^1, h_u^2 . Hàm mất mát cho một anchor h_u^1 được định nghĩa:

$$\ell(u) = -\log \frac{\exp(\text{sim}(h_u^1, h_u^2)/\tau)}{\sum_{v=1}^N \exp(\text{sim}(h_u^1, h_v^2)/\tau)},$$

trong đó $\text{sim}(\cdot, \cdot)$ là cosine similarity, τ là tham số nhiệt độ. Hàm mất mát tổng thể là trung bình trên tất cả các người dùng trong batch:

$$\mathcal{L} = \frac{1}{N} \sum_{u=1}^N \ell(u).$$

10.2.5 Lợi ích trong hệ gợi ý

Học tương phản giúp hệ gợi ý:

- Tăng tính **khái quát** của biểu diễn người dùng, đặc biệt khi dữ liệu tương tác thưa thớt.
- Giảm hiện tượng **overfitting** bằng cách tận dụng các views được tạo ra từ dữ liệu gốc.
- Cải thiện **độ chính xác gợi ý** trong các tác vụ downstream như dự đoán click, xếp hạng sản phẩm hoặc gợi ý theo chuỗi.

Nhờ những ưu điểm trên, học tương phản đã trở thành một trong những hướng nghiên cứu quan trọng trong xây dựng hệ gợi ý thế hệ mới.

10.3 Học tương phản cho gợi ý chuỗi CL4Rec [31]

10.3.1 Động lực

Trong hệ gợi ý, một thách thức lớn là làm sao học được biểu diễn (representation) giàu thông tin cho người dùng và phiên tương tác khi dữ liệu thường thưa thớt và không đầy đủ nhãn. CL4Rec [31] khai thác ý tưởng học tương phản (contrastive learning), trong đó các biểu diễn từ những *views* khác nhau của cùng một chuỗi tương tác được tối ưu để gần nhau trong không gian embedding, trong khi các biểu diễn từ chuỗi khác biệt phải được đẩy xa nhau. Nhờ vậy, mô hình học được representation ổn định và giàu ngữ nghĩa hơn, hữu ích cho cả bài toán gợi ý ngắn hạn và dài hạn.

10.3.2 Chuẩn bị dữ liệu huấn luyện

Tương tự SASRec và BERT4Rec, dữ liệu đầu vào được biểu diễn dưới dạng chuỗi tương tác:

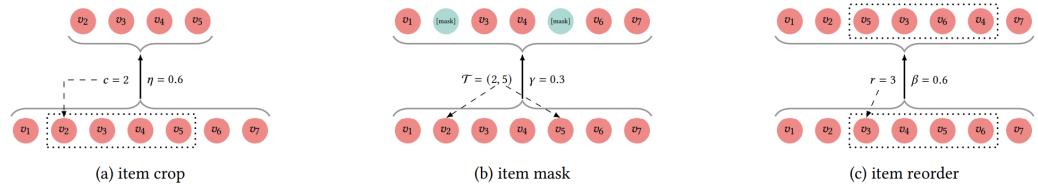
$$s_u = [v_1, v_2, \dots, v_T],$$

trong đó v_t là item mà người dùng u đã tương tác tại bước t .

Khác biệt then chốt của CL4Rec là từ một chuỗi gốc s_u , ta sinh ra nhiều *views* khác nhau thông qua các kỹ thuật **data augmentation**, sau đó tối ưu để các views này có embedding gần nhau.

10.3.3 Data Augmentation cho chuỗi phiên

CL4Rec đề xuất ba cách chính để tạo views (Hình 10.3):



Hình 10.3: Minh họa Data Augmentation.

- **Cropping:** Chọn một đoạn con liên tiếp từ chuỗi gốc. Ví dụ: $[v_1, v_2, v_3, v_4, v_5] \rightarrow [v_2, v_3, v_4]$. Mục tiêu là học được rằng các đoạn con đều phản ánh sở thích chung của người dùng.
- **Masking:** Ngẫu nhiên che (mask) một tỷ lệ item trong chuỗi bằng token đặc biệt `[MASK]`, tương tự như trong BERT4Rec. Ví dụ:

$$[v_1, v_2, v_3, v_4] \rightarrow [v_1, [\text{MASK}], v_3, v_4].$$

Cách này giúp tăng tính robust và tận dụng ngữ cảnh hai chiều.

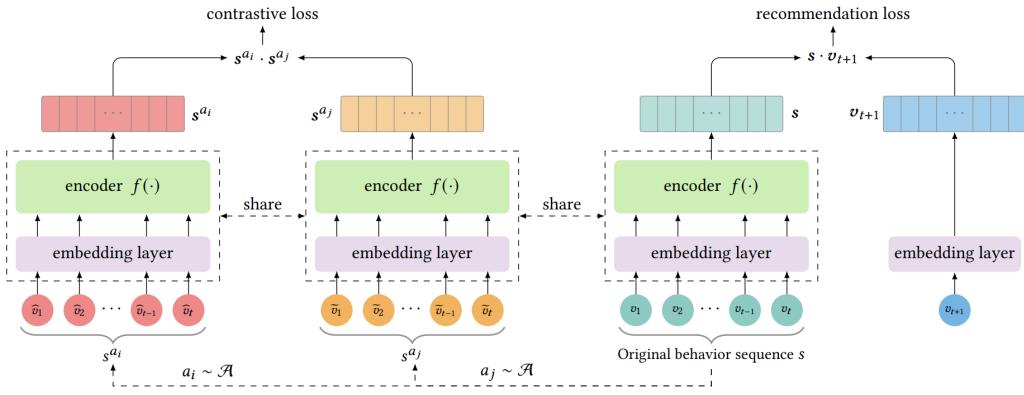
- **Reordering:** Hoán đổi thứ tự của một đoạn con trong chuỗi. Ví dụ:

$$[v_1, v_2, v_3, v_4, v_5] \rightarrow [v_1, v_3, v_2, v_4, v_5].$$

Điều này buộc mô hình học được tính bất biến với nhiễu nhỏ trong thứ tự hành vi.

Khi huấn luyện, từ chuỗi gốc s_u ta ngẫu nhiên áp dụng hai phép biến đổi để thu được hai views khác nhau:

$$s_u^{(1)} = \text{Augment}_1(s_u), \quad s_u^{(2)} = \text{Augment}_2(s_u).$$



Hình 10.4: Minh họa CL4Rec.

10.3.4 Kiến trúc mô hình

Kiến trúc CL4Rec được xây dựng dựa trên SASRec, bao gồm ba thành phần chính:

- **Embedding layer:** ánh xạ mỗi item ID và vị trí (position) trong chuỗi sang vector trong không gian \mathbb{R}^d . Lớp embedding này giúp mô hình hóa cả thông tin nội dung của item và thứ tự trong chuỗi.
- **Unidirectional Transformer encoder:** chuỗi embedding sau đó được đưa qua nhiều lớp self-attention với *causal mask*, đảm bảo rằng tại vị trí t mô hình chỉ có thể chú ý đến các item trong quá khứ ($\leq t$). Encoder này đóng vai trò là hàm mã hóa $f_\theta(\cdot)$, trích xuất biểu diễn ngữ cảnh hóa cho toàn bộ chuỗi hành vi.
- **Projection head:** để tối ưu trong không gian tương phản, embedding đầu ra của encoder $h_u^{(i)}$ được đưa qua một MLP nhỏ $g(\cdot)$ (thường gồm hai tầng tuyến tính kèm nonlinear activation và normalization). Projection head này giúp biểu diễn được chuẩn hóa và thích hợp hơn cho việc so sánh tương đồng trong học tương phản.

Cụ thể, với mỗi view $s_u^{(i)}$ được tạo từ chuỗi gốc s_u , ta thu được:

$$h_u^{(i)} = f_\theta(s_u^{(i)}),$$

trong đó $h_u^{(i)} \in \mathbb{R}^d$ là embedding đầu ra từ Transformer encoder. Tiếp đó, biểu diễn này được ánh xạ sang không gian tương phản nhờ projection head:

$$z_u^{(i)} = g(h_u^{(i)}).$$

Các vector $z_u^{(i)}$ sau khi chuẩn hóa sẽ được sử dụng trong hàm mất mát InfoNCE để tối ưu tương đồng giữa các views dương và phân tách khỏi các views âm.

10.3.5 Hàm mất mát

CL4Rec sử dụng hàm mất mát InfoNCE tương tự SimCLR (Hình 10.4). Với một batch gồm N chuỗi, mỗi chuỗi sinh ra hai views, tổng cộng có $2N$ biểu diễn $\{z_u^{(i)}\}$. Đối với một cặp views dương $(z_u^{(1)}, z_u^{(2)})$, hàm mất mát định nghĩa như sau:

$$\ell(z_u^{(1)}, z_u^{(2)}) = -\log \frac{\exp(\text{sim}(z_u^{(1)}, z_u^{(2)})/\tau)}{\sum_{v \neq u} \exp(\text{sim}(z_u^{(1)}, z_v^{(j)})/\tau)},$$

trong đó:

- $\text{sim}(a, b) = \frac{a^\top b}{\|a\|\|b\|}$ là cosine similarity,
- τ là hệ số nhiệt độ (temperature hyperparameter).

Hàm mất mát cuối cùng là trung bình trên tất cả các cặp dương trong batch:

$$\mathcal{L}_{CL} = \frac{1}{N} \sum_{u=1}^N \frac{1}{2} (\ell(z_u^{(1)}, z_u^{(2)}) + \ell(z_u^{(2)}, z_u^{(1)})).$$

10.3.6 Ưu điểm

- Khai thác được dữ liệu không nhãn thông qua học tương phản.
- Cải thiện độ robust của biểu diễn phiên nhờ data augmentation.
- Học được embedding giàu ngữ nghĩa, hỗ trợ tốt cho dự đoán tiếp theo trong hệ gợi ý.

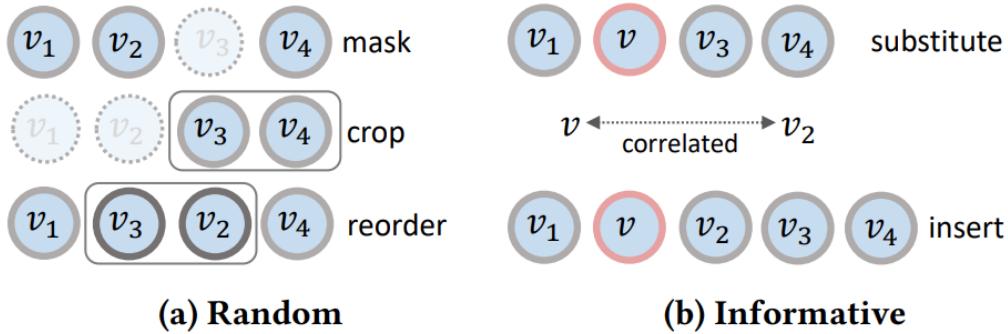
10.4 Tăng cường dữ liệu trong học tương phản CoSeRec [18]

10.4.1 Động lực

Các mô hình gợi ý tuần tự dựa trên self-attention như SASRec đã đạt được nhiều tiến bộ, tuy nhiên chúng vẫn phụ thuộc lớn vào dữ liệu gán nhãn (lịch sử tương tác rõ ràng). Điều này gây ra hạn chế trong bối cảnh dữ liệu khan hiếm, tương tác ngắn hoặc thiếu thông tin. CoSeRec (Liu et al., 2021) [18] được đề xuất nhằm tận dụng

học tương phản tự giám sát (contrastive self-supervised learning), giúp mô hình học biểu diễn mạnh mẽ hơn thông qua các chiến lược *data augmentation* trên chuỗi tương tác.

10.4.2 Chiến lược tăng cường dữ liệu trong CoSeRec (Hình 10.5)



Hình 10.5: Minh họa Data Augmentation trong CoSeRec.

Một đóng góp quan trọng của CoSeRec [18] là thiết kế các phép tăng cường dữ liệu (*data augmentation*) cho chuỗi người dùng, nhằm sinh ra các *views* đa dạng nhưng vẫn giữ được tính nhất quán ngữ nghĩa. CoSeRec đề xuất tổng cộng năm phép biến đổi, chia thành hai nhóm chính: **ngẫu nhiên** và **dựa trên thông tin**.

(i) Các phép biến đổi ngẫu nhiên

Nhóm này chủ yếu tác động ngẫu nhiên lên chuỗi gốc, phá vỡ một phần cấu trúc tuần tự để mô hình học được tính bất biến:

- **Crop (cắt):** Chọn một đoạn con liên tiếp ngẫu nhiên của chuỗi gốc $\mathbf{A} = [v_1, \dots, v_m]$. Kết quả là:

$$\mathbf{A}_{crop} = [v_s, v_{s+1}, \dots, v_{s+\lceil P \times m \rceil - 1}],$$

trong đó s được chọn ngẫu nhiên và P là tỉ lệ cắt.

- **Mask (che):** Một tập vị trí trong chuỗi được chọn ngẫu nhiên để thay bằng token đặc biệt `[MASK]`, thu được chuỗi \mathbf{A}_{mask} . Tỉ lệ mask được điều khiển bởi tham số P .

- **Reorder (xáo trộn):** Chọn một đoạn con liên tiếp và thay thế thứ tự các mục bằng một hoán vị ngẫu nhiên. Nếu chọn đoạn $[v_i, \dots, v_{i+k}]$, sau khi xáo trộn thu được:

$$[v_{\pi(i)}, v_{\pi(i+1)}, \dots, v_{\pi(i+k)}],$$

với π là một hoán vị ngẫu nhiên.

Các phép biến đổi này tạo ra sự đa dạng mạnh mẽ, song có thể làm giảm tính hữu ích của dữ liệu, đặc biệt trên các chuỗi ngắn.

(ii) Các phép biến đổi dựa trên thông tin

Nhằm khắc phục hạn chế của nhóm ngẫu nhiên, CoSeRec bổ sung hai phép biến đổi dựa trên quan hệ giữa các mục:

- **Substitute (thay thế):** Chọn một tập chỉ số $\{i_1, \dots, i_k\}$ (khoảng $k = r \times m$), và thay mỗi v_i bằng một mục v'_i có tương quan cao với v_i :

$$\mathbf{A}_{sub} = [v_1, \dots, v_{i_1-1}, v'_{i_1}, v_{i_1+1}, \dots, v'_{i_k}, v_{i_k+1}, \dots, v_m].$$

- **Insert (chèn):** Tại một số vị trí ngẫu nhiên j , chèn thêm các mục v'_j có tương quan cao ngay trước hoặc sau v_j , thu được chuỗi mở rộng \mathbf{A}_{ins} có độ dài xấp xỉ $(1 + r)m$.

(iii) Xác định các mục tương quan cao

Để chọn các mục thay thế/chèn (v'_i), CoSeRec tính *điểm tương quan* giữa các mục dựa trên hai phương pháp:

1. **ItemCF-IUF (memory-based):** Dựa trên lý thuyết lọc cộng tác theo mục. Độ tương quan giữa hai mục i và j được tính:

$$\text{simIUF}(i, j) = \sum_{u \in U_i \cap U_j} \frac{1}{\log(1 + |I_u|)},$$

trong đó U_i là tập người dùng đã tương tác với i , còn $|I_u|$ là số mục mà người dùng u đã tương tác. Thuật toán này giảm trọng số của người dùng có lịch sử quá dài.

2. **Model-based similarity:** Dựa trên biểu diễn vector của mục do mô hình học. Với $e_i, e_j \in \mathbb{R}^d$ là embedding của i và j , điểm tương quan được tính bằng cosine:

$$\text{sim}_{\text{model}}(i, j) = \frac{e_i^\top e_j}{\|e_i\| \cdot \|e_j\|}.$$

Điểm tương quan cuối cùng được chọn là:

$$\text{sim}(i, j) = \max(\text{sim}_{\text{IUF}}(i, j), \text{sim}_{\text{model}}(i, j)).$$

Các mục v'_i được chọn trong tập có $\text{sim}(i, j)$ cao nhất với mục gốc.

(iv) Kết hợp chiến lược theo độ dài chuỗi

CoSeRec áp dụng tỉ lệ khác nhau cho từng loại biến đổi tùy vào độ dài chuỗi:

- Với chuỗi ngắn: Ưu tiên phép thay thế và chèn (ít gây mất mát thông tin).
- Với chuỗi dài: Cho phép áp dụng crop, mask, reorder nhiều hơn để tạo ra views đa dạng.

Nhờ đó, CoSeRec vừa duy trì sự đa dạng trong dữ liệu tăng cường, vừa đảm bảo rằng các views sinh ra vẫn giữ tính hợp lý và giàu thông tin.

10.4.3 Kiến trúc mô hình

Kiến trúc CoSeRec kế thừa trực tiếp từ SASRec:

- **Embedding layer:** ánh xạ item ID và vị trí sang vector trong \mathbb{R}^d .
- **Unidirectional Transformer encoder (SASRec):** mỗi item chỉ được phép chú ý đến các item trước đó (causal mask), đảm bảo tính tuần tự. Đây chính là encoder $f_\theta(\cdot)$.
- **Projection head:** một MLP nhỏ $g(\cdot)$ để ánh xạ embedding cuối cùng sang không gian tương phản, giúp tối ưu hoá hàm mất mát InfoNCE.

Với mỗi view $s_u^{(i)}$, ta có:

$$h_u^{(i)} = f_\theta(s_u^{(i)}), \quad z_u^{(i)} = g(h_u^{(i)}),$$

trong đó $z_u^{(i)} \in \mathbb{R}^d$ là biểu diễn đã chuẩn hóa để tính toán độ tương đồng.

10.4.4 Hàm mất mát tương phản

CoSeRec sử dụng hàm mất mát InfoNCE để kéo gần các views dương và đẩy xa các views âm:

$$\mathcal{L}_{\text{contrast}} = - \sum_{u=1}^N \log \frac{\exp(\text{sim}(z_u^{(1)}, z_u^{(2)})/\tau)}{\sum_{v=1}^N \exp(\text{sim}(z_u^{(1)}, z_v^{(2)})/\tau)},$$

trong đó:

- $\text{sim}(\cdot, \cdot)$ là cosine similarity,
- τ là hệ số nhiệt độ (temperature),
- N là số lượng chuỗi trong batch.

10.4.5 Hàm mất mát kết hợp

Mô hình CoSeRec được huấn luyện với hai mục tiêu:

- **Loss dự đoán gợi ý:** cross-entropy giữa dự đoán item tiếp theo và item thật.
- **Loss tương phản:** tối ưu biểu diễn nhất quán giữa các views của cùng một người dùng.

Hàm mất mát cuối cùng:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \cdot \mathcal{L}_{\text{contrast}},$$

trong đó λ là hệ số điều chỉnh cân bằng giữa hai mục tiêu.

10.4.6 Ưu điểm

- Kết hợp chặt chẽ giữa *supervised learning* (dự đoán item) và *self-supervised contrastive learning*.
- Augmentation đa dạng và bền vững, giúp học được biểu diễn bất biến và giàu thông tin.
- Cải thiện đáng kể trên nhiều benchmark datasets so với SASRec và BERT4Rec.

10.5 Các phương pháp mở rộng học tương phản trong hệ gợi ý

10.5.1 Học tương phản có tính đổi biến cho gợi ý tuần tự ECL-SR [35]

Động lực và mục tiêu. ECL-SR [35] được đề xuất nhằm khắc phục hạn chế của các phương pháp học tương phản (CL) tuần tự hiện tại, vốn thường buộc biểu diễn phải bất biến với mọi loại biến đổi. Tuy nhiên, một số phép biến đổi xâm lấn (invasive augmentations) như thay thế mục trong chuỗi có thể làm thay đổi ngữ nghĩa hành vi người dùng. Do đó, ECL-SR chia augmentations thành hai loại: (i) *nhỏ* (mild) như dropout trên embedding, và (ii) *xâm lấn* như thay thế mục. Mục tiêu là học bất

biến (invariance) đối với augmentations nhẹ, nhưng học tương ứng (equivariance) đối với augmentations xâm lấn bằng cách dùng *conditional discriminator* để phát hiện thay đổi.

Kiến trúc mô hình. Mô hình gồm ba thành phần:

- **User Behavior Encoder (UBE):** dựa trên SASRec, mã hoá chuỗi hành vi người dùng để dự đoán mục tiếp theo.
- **Conditional Discriminator (CD):** cũng dùng SASRec, phân biệt vị trí bị thay thế trong chuỗi augment.
- **Generator (G):** dựa trên BERT4Rec, sinh các mục thay thế trong augment xâm lấn.

Cả ba thành phần chia sẻ cùng bảng embedding. Augmentation gồm: (i) dropout masking trong không gian embedding (*mild*), và (ii) item substitution qua Generator (*invasive*).

Chiến lược tăng cường dữ liệu.

- *Mild augmentation:* dropout hai lần độc lập trên embedding tạo ra hai view dương, dùng để học tính bất biến.
- *Invasive augmentation:* Generator thay thế một số mục trong chuỗi, sinh ra view xâm lấn. CD phải dự đoán chính xác vị trí bị thay thế, học tính tương ứng.

Hàm mất mát. Hàm tối ưu tổng hợp:

$$\mathcal{L} = \mathcal{L}_{rec} + \alpha \mathcal{L}_{CL} + \beta \mathcal{L}_{RID}.$$

Thành phần CL bất biến (InfoNCE):

$$\mathcal{L}_{CL} = -\frac{1}{|B|} \sum_{u \in B} \log \frac{\exp(\text{sim}(z_u, z_{u+})/\tau)}{\sum_{v \in B} \exp(\text{sim}(z_u, z_v)/\tau)},$$

với z_u, z_{u+} là embedding của cùng một chuỗi sau hai dropout khác nhau.

Thành phần RIDL (Replaced Item Detection Loss):

$$\mathcal{L}_{RID} = - \sum_{(u,i)} \left[y_{ui} \log \sigma(w^\top [z_u; z_i]) + (1 - y_{ui}) \log (1 - \sigma(w^\top [z_u; z_i])) \right],$$

trong đó $y_{ui} = 1$ nếu mục i trong chuỗi u bị thay thế, $[z_u; z_i]$ là embedding ghép.

Kết quả thực nghiệm. Trên bốn tập dữ liệu chuẩn (Sports, Toys, Yelp, ML-1M), ECL-SR vượt trội các baseline, đạt Recall@20 và NDCG@20 cao nhất. Điều này chứng minh việc kết hợp học bất biến và học tương ứng đem lại lợi ích rõ rệt. Mã nguồn được công bố tại GitHub (Tokkiu/ECL).

10.5.2 Học tương phản với gợi ý dựa trên đồ thị LightGCL [3]

Động lực và mục tiêu. Các phương pháp CL trên đồ thị gợi ý thường dựa vào augmentations ngẫu nhiên (loại bỏ cạnh/nút) hoặc heuristic (clustering), nhưng dễ gây nhiễu hoặc mất thông tin quan trọng. LightGCL [3] thay thế bằng một augmentation toàn cục dựa trên phân rã giá trị đơn (SVD) của ma trận kè, giữ lại các thành phần chính để bảo toàn cấu trúc hợp tác toàn cục.

Kiến trúc mô hình. LightGCL sử dụng LightGCN hai tầng. Đầu vào gồm hai đồ thị:

- Đồ thị gốc với ma trận kè A .
- Đồ thị augmented từ SVD: $A' = U_q \Sigma_q V_q^\top$, với q trị đơn lớn nhất.

Embedding người dùng/mục là tổng embedding từ các tầng trên cả hai đồ thị.

Chiến lược tăng cường dữ liệu. Thay vì loại bỏ cạnh, LightGCL dùng SVD để tái tạo đồ thị, giữ quan hệ cộng tác chính và lọc bỏ nhiễu. Đây là augmentation toàn cục, ổn định hơn nhiều so với ngẫu nhiên.

Hàm mất mát. Hàm tổng:

$$\mathcal{L} = \mathcal{L}_{BPR} + \lambda \mathcal{L}_{CL}.$$

Thành phần CL:

$$\mathcal{L}_{CL} = -\frac{1}{|B|} \sum_{u \in B} \log \frac{\exp(\text{sim}(z_u, z'_u)/\tau)}{\sum_{v \in B} \exp(\text{sim}(z_u, z'_v)/\tau)},$$

với z_u embedding trên đồ thị gốc, z'_u embedding trên đồ thị SVD-augmented.

Kết quả thực nghiệm. Trên năm tập dữ liệu lớn (Yelp, Gowalla, ML-10M, Amazon-Book, Tmall), LightGCL vượt trội so với các baseline (ví dụ Recall@20 trên Yelp đạt 0.0793 so với 0.0718 của SimGCL). Kết quả ablation chỉ ra LightGCL đặc biệt hiệu quả khi dữ liệu thừa và giảm bias theo độ phổ biến. Mã nguồn được công bố tại GitHub (HKUDS/LightGCL).

Tài liệu tham khảo

- [1] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. Neural news recommendation with long- and short-term user representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 336–345. Association for Computational Linguistics, 2019.
- [2] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
- [3] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. Lightgcl: Simple yet effective graph contrastive learning for recommendation. In *International Conference on Learning Representations (ICLR)*. ICLR, 2023.
- [4] C Aggarwal Charu. *Recommender systems: The textbook*. 2016.
- [5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020.
- [6] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. Neural multi-task recommendation from multi-behavior data. In *Proceedings of the 35th IEEE International Conference on Data Engineering (ICDE)*, pages 1554–1557. IEEE, 2019.
- [7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6894–6910. Association for Computational Linguistics, 2021.
- [8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for rec-

- ommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.
- [9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.
 - [10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.
 - [11] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*, pages 263–272. IEEE Computer Society, 2008.
 - [12] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.
 - [13] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge university press, 2010.
 - [14] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.
 - [15] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization for recommender systems. *Journal of Machine Learning Research*, 12(1):88–102, 2020.
 - [16] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 211–219. ACM, 2020.
 - [17] Nathan N. Liu, Evan W. Xiang, Min Zhao, and Qiang Yang. Unifying explicit and implicit feedback for collaborative filtering. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1445–1448. ACM, 2010.
 - [18] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S. Yu, Julian McAuley, and Caiming Xiong. Contrastive self-supervised sequential recommendation with robust

- augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2021. arXiv preprint arXiv:2108.06479.
- [19] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 54–61. AAAI, 1996.
 - [20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 452–461. AUAI Press, 2009.
 - [21] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2010.
 - [22] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.
 - [23] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformers. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1441–1450, 2019.
 - [24] Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 83–90, 2012.
 - [25] Quyen Tran, Lam Tran, Linh Chu Hai, Ngo Van Linh, and Khoat Than. From implicit to explicit feedback: A deep neural network for modeling sequential behaviours and long-short term preferences of online users. *Neurocomputing*, 479:89–105, 2022.
 - [26] Anh Phan Tuan, Nhat Nguyen Trong, Duong Bui Trong, Linh Ngo Van, and Khoat Than. From implicit to explicit feedback: A deep neural network for modeling the sequential behavior of online users. In *Asian Conference on Machine Learning*, pages 1188–1203. PMLR, 2019.

- [27] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. Neural news recommendation with attentive multi-view learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.
- [28] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. Npa: Neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2576–2584. ACM, 2019.
- [29] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. Neural news recommendation with multi-head self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6389–6394. Association for Computational Linguistics, 2019.
- [30] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3597–3606. Association for Computational Linguistics, 2020.
- [31] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 990–1000. ACM, 2021.
- [32] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, volume 17, pages 3203–3209. Melbourne, Australia, 2017.
- [33] Markus Zanker and Markus Jessenitschnig. Case-studies on exploiting explicit customer requirements in recommender systems. In *Proceedings of the 2006 ACM symposium on Applied computing (SAC '06)*, pages 784–788, New York, NY, USA, 2006. ACM.
- [34] Jiyong Zhang and Pearl Pu. A recursive prediction algorithm for collaborative filtering recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 57–64, 2007.

- [35] Peilin Zhou, Jingqi Gao, Yueqi Xie, Qichen Ye, Yining Hua, Jae Boum Kim, Shoujin Wang, and Sunghun Kim. Equivariant contrastive learning for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys)*, pages 129–140. ACM, 2023.