TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐỒ ÁN MÔN HỌC
KHAI PHÁ WEB

Đề tài: Hệ gợi ý cho thương mại điện tử

Giảng viên: **TS. Nguyễn Kiêm Hiếu**

Mã lớp: 128761 – Nhóm: 17

Họ và tên	MSSV
Nguyễn Minh Đức	20183500
Phan Thanh Long	20183587
Trần Đức Duy	20183515
Phạm Văn Nam	20183598

Hà Nội, tháng 1 năm 2022

MỤC LỤC

Hình ảnh

LỜI CẢM ƠN

Thực tế luôn cho thấy, sự thành công nào cũng đều gắn liền với những sự hỗ trợ, giúp đỡ của những người xung quanh dù cho sự giúp đỡ đó là ít hay nhiều, trực tiếp hay gián tiếp. Trong suốt thời gian từ khi bắt đầu làm bài tập lớn đến nay, chúng em đã nhận được sự quan tâm, chỉ bảo, giúp đỡ của thầy cô, gia đình và bạn bè xung quanh. Với tấm lòng biết ơn vô cùng sâu sắc, chúng em xin gửi lời cảm ơn chân thành nhất từ đáy lòng đến quý thầy cô của trường Đại học Bách khoa Hà Nội đã cùng dùng những tri thức và tâm huyết của mình để có thể truyền đạt cho chúng em trong vốn kiến thức quý báu suốt thời gian học tập tại trường. Đặc biệt, chúng em xin chân thành cảm ơn thầy Nguyễn Kiêm Hiếu đã tận tâm chỉ bảo hướng dẫn chúng em qua từng buổi học, từng tiết bài giảng trên lớp. Nhờ có những lời hướng giảng, dạy bảo đó, bài tập lớn của chúng em đã hoàn thành một cách tốt nhất. Một lần nữa, chúng em xin gửi lời cảm ơn chân thành đến thầy. Ban đầu còn nhiều bỡ ngỡ vì vốn kiến thức của còn có hạn. Do vậy, không tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp của quý thầy cô và các bạn học cùng lớp để bài tập lớn được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

Trân trọng, tháng 1 năm 2022

Nhóm 17

1. Giới thiệu bài toán

1.1. Bài toán thực tế

Hiện nay, với sự phổ biến của điện thoại thông minh, sự phát triển của các trang thương mại điện tử ngày càng mạnh mẽ. Trong một thị trường thương mại điện tử có rất nhiều các mặt hàng, và đôi khi người dùng không tìm được sản phẩm mình yêu thích. Từ đó, hệ gợi ý được áp dụng vào trong rất nhiều trang thương mai điện tử.

Các công ty thương mại điện tử thường hoạt động trên khắp các thị trường; ví dụ: Amazon đã mở rộng hoạt động và bán hàng của họ đến 18 thị trường (tức là các quốc gia) trên toàn cầu. Gợi ý liên thị trường liên quan đến vấn đề giới thiệu các sản phẩm phù hợp cho người dùng ở thị trường mục tiêu (ví dụ: thị trường khan hiếm tài nguyên) bằng cách tận dụng dữ liệu từ các thị trường có nguồn tài nguyên cao tương tự, ví dụ: sử dụng dữ liệu từ thị trường Hoa Kỳ để cải thiện các đề xuất trong thị trường mục tiêu. Tuy nhiên, thách thức chính là dữ liệu, chẳng hạn như dữ liệu tương tác của người dùng với các sản phẩm (nhấp chuột, mua hàng, đánh giá), có đặc trưng nhất định của từng thị trường. Do đó, các thuật toán được đào tạo trên một thị trường nguồn không nhất thiết sẽ hiệu quả trong một thị trường mục tiêu khác.

Mục tiêu của bài toán là cải thiện hệ thống gợi riêng lẻ tại các thị trường mục tiêu này bằng cách tận dụng dữ liệu từ các thị trường phụ trợ tương tự. Và từ đó tăng doanh số cho người bán hàng.

1.2. Phát biểu bài toán

Chúng ta có 3 thị trường nguồn và 2 thị trường mục tiêu với mỗi tập người dùng trong các thị trường không trùng nhau (nghĩa là mỗi người dùng chỉ tương tác với một thị trường). Mục tiêu là tạo ra một hệ thống gợi ý tốt nhất cho 2 thị trường mục tiêu.

- Input: Một người dùng và một tập sản phẩm trong các sàn thương mại điện tử trong 1 thị trường mục tiêu.
- Output: Top 10 sản phẩm đúng với sở thích của người dùng được sắp xếp theo thứ tự giảm dần.

2. Động lực lựa chọn phương pháp giải quyết

Hệ gợi ý đang được ứng dụng nhiều trong rất nhiều trong các dịch vụ online như thương mại điện tử, quảng cáo và mạng xã hội. Cốt lõi của gợi ý là có thể ước tính được sở thích của người dùng với một item dựa trên lịch sử yêu thích của user đó. Kỹ thuật lọc cộng tác (Collaborative filtering - CF) giải quyết vấn đề đó bằng cách giả sử các người dùng có những tương tác với hệ thống như nhau sẽ có sở thích như nhau.
Các công việc chính của CF model là:

- Embedding: học ra biểu diễn của user, item.
- Interaction modeling: học ra sự tương tác dựa trên embedding đó.

Một số kỹ thuật cho CF: matrix factorization (MF), neural collaborative filtering (NMF)
Các phương pháp trên chỉ học biểu diễn tương tác user-item dựa trên embedding của user, item chứ không học trên tương tác user- item khác (collaborative signal).
Mạng thần kinh (Neural Network) trong thập kỷ qua đã đạt được nhiều thành công, đánh dấu một bước tiến lớn trong lĩnh vực trí tuệ nhân tạo. Tuy nhiên, các biến thể ban đầu của mạng thần kinh chỉ có thể được triển khai trên dữ liệu thông thường hoặc Euclide, trong khi rất nhiều dữ liệu trong thế giới thực có cấu trúc không phải dạng Euclide như cấu trúc đồ thị. Điều đó đã dẫn đến sự phát triển của lĩnh vực mạng thần kinh đồ thị (Graph Neural Networks). Graph Convolution Network (GCN) là một biến thể cơ bản của mạng thần kinh đồ thị và đang trở thành phương pháp SOTA mới cho bài toán lọc cộng tác (Collaborative Filtering).
Neural Graph Collaborative Filtering (NGCF) mong muốn lan truyền các sự tương tác user-item thông qua high-order connectivity trong đồ thị user-item.
Tuy nhiên, hiện đang tồn tại vấn đề trong thiết kế của GCN là feature transformation và nonlinear activation không mang lại nhiều lợi ích mà ngược lại còn tăng độ phức tạp, khiến thời gian training lâu hơn và làm giảm độ chính xác của mô hình. LightGCN đề xuất một kiến trúc đơn giản hơn thay thế cho Neural Graph Collaborative Filtering (NGCF) – phương pháp dựa trên GCN cho bài toán Collaborative Filtering (CF) – bao gồm những thành phần thiết yếu nhất của GCN và đem lại kết quả vượt trội so với phương pháp ban đầu.

3. Kiến trúc mô hình

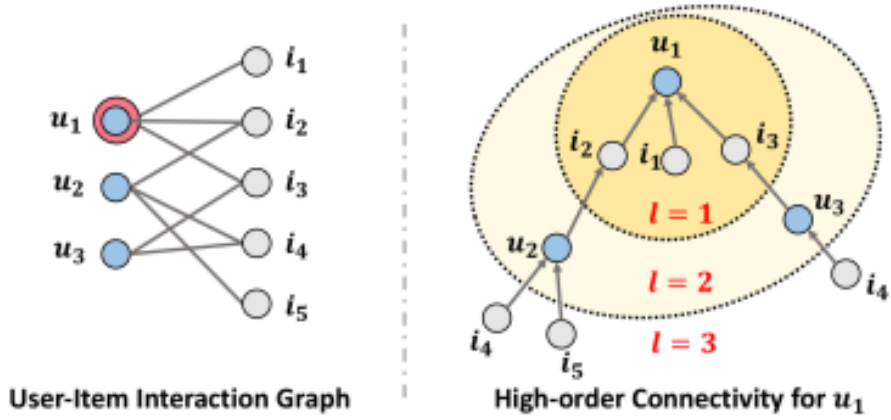
3.1. Neural Graph Collaborative Filtering (NGCF)

3.1.1. Giới thiệu

Học các biểu diễn vectơ (hay còn gọi là embedding) của người dùng và các sản phẩm nằm ở cốt lõi của các hệ thống gợi ý hiện đại. Thay đổi từ matrix factorization ban đầu đến các phương pháp dựa trên học sâu mới xuất hiện gần đây, những nỗ lực hiện tại thường thu được cách embedding của người dùng (hoặc của một mặt hàng) bằng cách ánh xạ từ các tính chất đã có trước đó mô tả người dùng (hoặc mặt hàng), chẳng hạn như ID và thuộc tính. Tuy nhiên, một nhược điểm cố hữu của các phương pháp như vậy là không có encoded embedding trong các tương tác giữa người dùng-sản phẩm và không được mã hóa trong quá trình embedding. Do đó, kết quả embedding có thể không đủ để thu được hiệu ứng lọc cộng tác (CF).
Vì vậy, phương pháp này đề xuất tích hợp các tương tác giữa người dùng và sản phẩm - cụ thể hơn là cấu trúc biểu đồ hai bên - vào quy trình embeding.

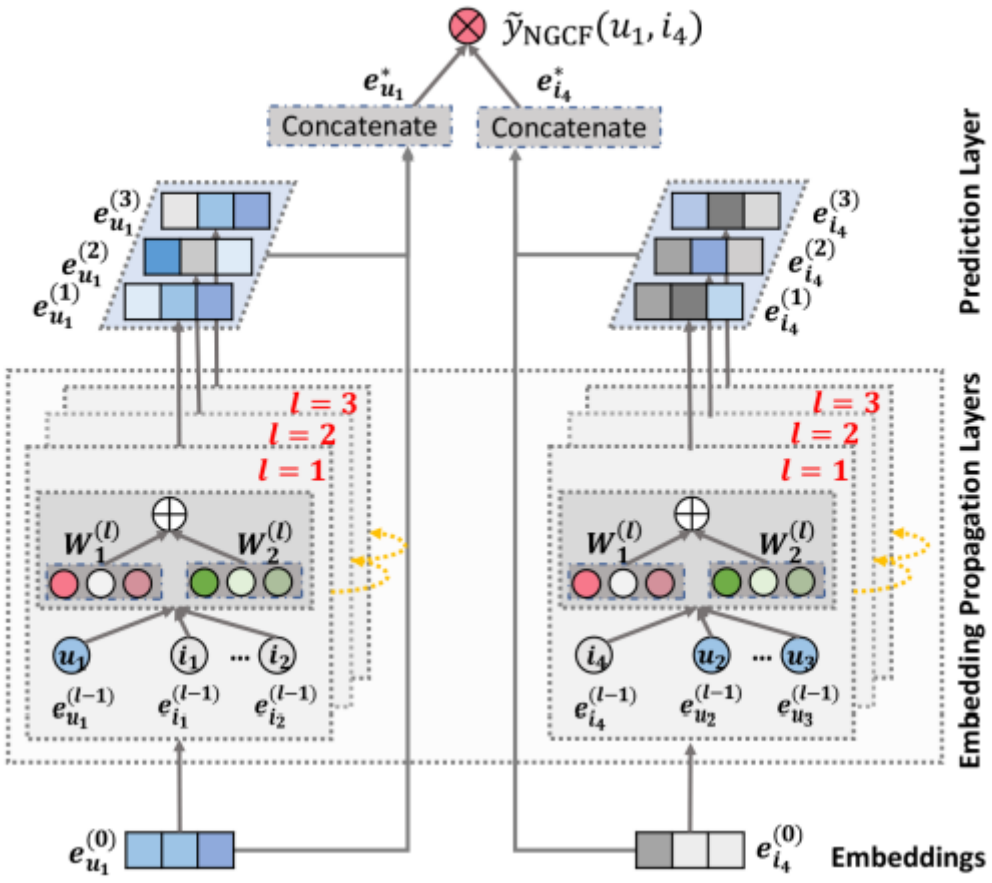
3.1.2. Kiến trúc

Ví dụ: Hình 1 minh họa khái niệm kết nối bậc cao. Kết nối cao hơn như vậy chứa nhiều ngữ nghĩa mang tín hiệu cộng tác. Ví dụ: đường dẫn $u_1 \leftarrow i_2 \leftarrow u_2$ cho biết sự giống nhau về hành vi giữa u1 và u2, vì cả hai người dùng đã tương tác với i2; đường dẫn dài hơn $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$ gợi ý rằng u_1 có khả năng sử dụng i_4, vì user u_2 tương tự của đã sử dụng i_4 trước đó. Hơn nữa, từ cái nhìn tổng thể của l = 3, mục i_4 có nhiều khả năng được u1 quan tâm hơn mục i_5, vì có hai đường dẫn kết nối $<i_4, u_1>$, trong khi chỉ có một đường dẫn kết nối $<i_5, u_1>$.



Hình 1. Hình minh họa về biểu đồ tương tác giữa người dùng-item và kết nối bậc cao. Nút u_1 là người dùng mục tiêu để cung cấp các đề xuất.

Tác giả thiết kế một phương pháp mạng nơron để truyền các phép embeding một cách đệ quy trên biểu đồ. Có thể được coi là xây dựng các luồng thông tin trong không gian embeding. Cụ thể, tạo ra một lớp lan truyền embeding, lớp này sẽ tinh chỉnh cách embeding của người dùng (hoặc một mặt hàng) bằng cách tổng hợp các lần embeding của các mặt hàng (hoặc người dùng) được tương tác. Bằng cách xếp chồng nhiều lớp lan truyền embeding, chúng ta có thể thực thi các quá trình embeding để thu tín hiệu cộng tác trong các kết nối bậc cao. Lấy Hình 1 làm ví dụ, xếp chồng hai lớp nắm bắt sự giống nhau về hành vi của $u_1 \leftarrow i_2 \leftarrow u_2$, xếp chồng ba lớp nắm bắt các đề xuất tiềm năng của $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$ và sức mạnh của luồng thông tin (được ước tính bởi trọng số có thể huấn luyện giữa các lớp) xác định mức độ ưu tiên gợi ý của i_4 và i_4.



Hình 2. Một minh họa về kiến trúc mô hình NGCF (các đường mũi tên thể hiện luồng thông tin). Các đại diện của user u_1 (trái) và item i_4 (phải) được tinh chỉnh với nhiều lớp lan truyền embedding, các đầu ra của chúng được nối với nhau để đưa ra dự đoán cuối cùng.

NGCF bao gồm 2 quá trình First-order Propagation và High-order Propagation tương ứng là quá trình lan truyền thông tin trong 1 layer và các layer với nhau.

First-order Propagation:

First-order Propagation học ra biểu diễn sự tương tác user-item dựa trên 2 operator là: message construction và message aggregation.
Message construction:

$$\mathbf{m}_{u\leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i + \mathbf{W}_2(\mathbf{e}_i \odot \mathbf{e}_u)\right),$$

Message embedding của item i lan truyền tới user u được kí hiệu là $m_{u\leftarrow i}$, tính theo W_1, W_2, là các ma trận cần học để biểu diễn message (share ma trận này với cùng một user) và e_u, e_i lần lượt là embedding của user và item đó . $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ đóng vai trò như decay factor để chọn lọc thông tin lan truyền, trong đó N là các hàng xóm lân cận của user và item.

Message Aggregation:

$$\mathbf{e}_u^{(1)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u} + \sum_{i\in\mathcal{N}_u}\mathbf{m}_{u\leftarrow i}\right),$$

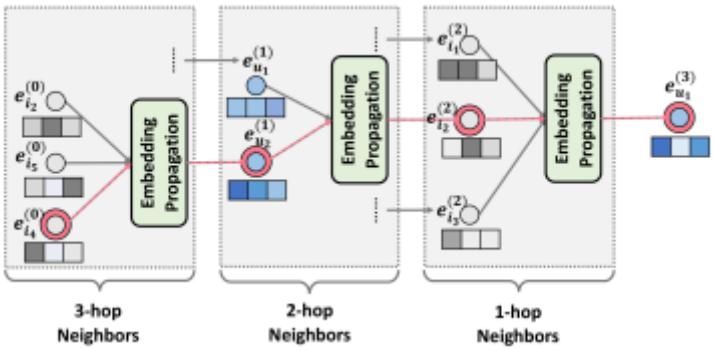
Sau khi tính được Message construction giữa một user và một item , ta tổng hợp lại thông tin cho biểu diễn của user u bằng tổng tất cả item hàng xóm của user u: $\sum m_{u\leftarrow i}$, cộng với một self-connection của chính nó là $m_{u\leftarrow u}$.

High-order Propagation
Tổng quát công thức tính biểu diễn của user qua một layer:

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u}\mathbf{m}_{u\leftarrow i}^{(l)}\right)$$

Với $e_u^{(l)}$ thì (l) chính là kí hiệu cho layer thứ l.

Bên cạnh đó sẽ lan truyền biểu diễn message construction qua các layer, lần lượt xen kẽ biểu diễn user, biểu diễn item, chính là High-order Propagation:



Hình 3. Minh họa về truyền embedding bậc ba cho người dùng u_1

Định nghĩa công thức lan truyền qua các layer:

$$\begin{cases} \mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right), \\ \mathbf{m}_{u\leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases}$$

Trong đó $m_{u\leftarrow i}^{(l)}$, $m_{u\leftarrow u}^{(l)}$ hình là Message construction từ item i tới u tương tự phần **First-order Propagation**, chỉ khác là thêm ở layer thứ l. p_{ui} (vẫn có thể lấy bằng $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$) đóng vai trò như một hệ số lan truyền, chọn lọc thông tin lan truyền qua các layer.

3.1.3. Loss

Sau khi có được biểu diễn của user và item qua các layer, có thể tiến hành tổng hợp lại thông tin giữa các layer bằng concat hoặc các cách thức khác như weighted average, max pooling, LSTM, …
Dự đoán khả năng yêu thích:

$$\hat{y}_{\text{NGCF}}(u,i) = \mathbf{e}_u^{*\top}\mathbf{e}_i^*.$$

Hàm loss sử dụng Bayesian Personalized Ranking Loss:

$$Loss = \sum_{(u,i,j)\in O} -\ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\|\Theta\|_2^2,$$

Trong đó λ là tham số của L_2 regularization để tránh overfitting,

Θ $=\{\mathrm{E}, \{\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}\}_{l=1}^{L}\}$ là tất cả các ma trận(tham số) cần học.

Cặp (u,i,j) tương ứng i là mẫu có tương tác, j là mẫu không có tương tác đối với user u.

3.2. Tổng quan về các mô hình

3.2.1. Collaborative Filtering (CF)

Lọc cộng tác (Collaborative Filtering) là một trong những hướng tiếp cận của hệ thống gợi ý (Recommender System) – có nhiệm vụ dự đoán ra người dùng sẽ tương tác (click, rate, purchase …) với những item nào. CF tập trung vào khai thác lịch sử tương tác của người dùng để nâng cao trải nghiệm cá nhân hóa trong gợi ý. Mục tiêu chính của CF là học được vectơ biểu diễn của user và item để đưa ra dự đoán dựa trên các vectơ đó.

3.2.2. Graph Convolution Network (GCN)

Ý tưởng cơ bản của của GCN là học biểu diễn của các nút (node) trong mạng. Để làm được điều đó, một nút sẽ được biểu diễn dựa trên sự kết hợp các đặc trưng của các nút hàng xóm (neighbor):

$$\mathbf{e}_u^{(k+1)} = \mathrm{AGG}(\mathbf{e}_u^{(k)}, \{\mathbf{e}_i^{(k)} : i \in \mathcal{N}_u\}).$$

Trong đó, AGG (aggregation function) đóng vai trò cốt lõi của mạng GCN, thể hiện rằng một nút ở layer thứ k được biểu diễn bởi các nút hàng xóm. AGG có thể là các hàm như weighted sum, LSTM, bilinear, … cho biểu diễn của nút khá tốt trong tác vụ phân loại tuy nhiên lại phức tạp quá mức cho bài toán CF.

3.2.3. Neural Graph Collaborative Filtering (NGCF)

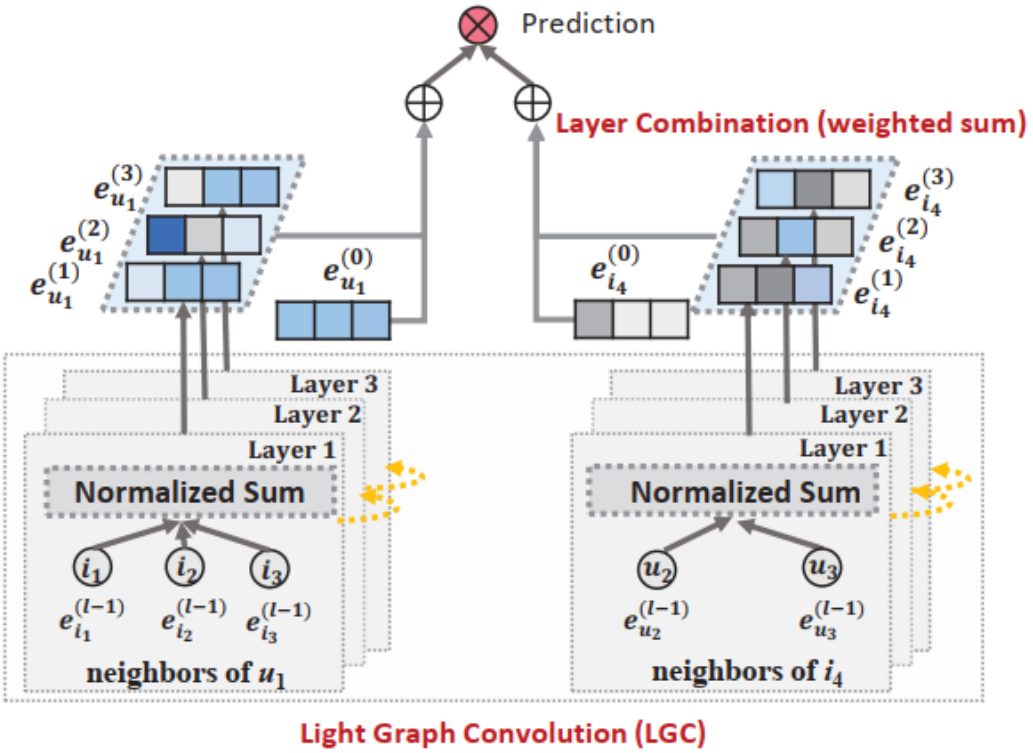
NGCF là phương pháp dựa trên GCN cho bài toán CF gồm ba thành phần chính: feature transformation, neighborhood aggregation và nonlinear activation.

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}))\right),$$

$$\mathbf{e}_i^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)}))\right),$$

Trong công thức trên, $e_u^{(k)}$ và $e_i^{(k)}$ là các biểu diễn (embedding) của user u và item i tại layer thứ k, σ là hàm kích hoạt phi tuyến (nonlinear activation function), N_u và N_i là tương ứng là tập hợp các item (user) tương tác với user (item), W_1 và W_2 là các ma trận trọng số. Khi lan truyền L layer, NGCF đạt được L+1 embedding của user ($e_u^{(0)}$, $e_u^{(1)}$, …, $e_u^{(L)}$) và item ($e_i^{(0)}$, $e_i^{(1)}$, …, $e_i^{(L)}$). Sau đó thực hiện nối (concatenate) L+1 embeeding đó để đạt được embeeding cuối cùng cho user và item phục vụ cho tác vụ dự đoán.

3.3. LightGCN



Hình 4. Kiến trúc mô hình LightGCN

3.3.1. Light Graph Convolution (LGC)

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}.$$

Trong LightGCN, các thành phần feature transformation và nonlinear activation của NGCF được loại bỏ, LightGCN sử dụng cách kết hợp weighted sum đơn giản hơn. Biểu thức chuẩn hóa đối xứng (symmetric normalization) $\frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}$ giúp tránh bị scale embedding.

3.3.2. Layer Combination and Model Prediction

Sau quá trình lan truyền K layer LGC và thu được L+1 embedding, chúng ta thực hiện kết hợp chúng để thu được final embedding theo công thức sau:

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)},$$

Trong đó $\alpha_k \geq 0$ là trọng số thể hiện mức độ quan trọng của embedding ở layer thứ k. α_k có thể là một siêu tham số (hyperparameter) để tuning thủ công, hoặc cũng có thể là tham số của mô hình được học tự động (như attention). Thực nghiệm cho thấy sử dụng một giá trị đồng nhất $\alpha_k = \frac{1}{K+1}$ cho hiệu suất khá tốt.

Cuối cùng thực hiện nhân hai final embedding để thu được prediction score:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i,$$

3.3.3. Matrix Form

Ma trận tương tác user-item $R \in R^{M \times N}$ với M và N là số lượng user và item, $\mathbf{R_{ui}}$ = 1 khi u có tương tác với i, ngược lại R_{ui} = 0. Từ đó thu được ma trận kề như sau:

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix},$$

Ma trận embedding tại layer thứ k là $E^{(k)} \in R^{(M+N) \times T}$ trong đó T là kích thước của embedding. Ta có biểu thức LGC dưới dạng ma trận:

$$\mathbf{E}^{(k+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)},$$

Trong đó, **D** là ma trận đương chéo có kích thước là (M+N)x(M+N), $\mathbf{D_{ii}}$ là số phần tử khác 0 của hàng thứ i ma trận A.

Cuối cùng ta có ma trận final embeeding:

$$\begin{aligned} \mathbf{E} &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_K\mathbf{E}^{(K)} \\ &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \alpha_2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + \alpha_K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}, \end{aligned}$$

Với $\tilde{A} = D^{\frac{-1}{2}}AD^{\frac{-1}{2}}$ là ma trận chuẩn hóa đối xứng (symmetric normalization matrix).

3.3.4. Phân tích mô hình

Để chứng minh tính đúng đắn của mô hình LightGCN, chúng ta tiến hành so sánh với một số mô hình hiện có để làm rõ ưu điểm của LightGCN trên các khía cạnh khác nhau.

3.3.4.1. Mối quan hệ với SGCN

Simplified GCN (SGCN) là một mạng GCN tuyến tính đơn giản đưa ra đề xuất bỏ nonlinear activation và thu gọn các ma trận trọng số về một ma trận duy nhất để giảm độ phức tạp quá mức của GCN:

$$\mathbf{E}^{(k+1)} = (\mathbf{D} + \mathbf{I})^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})(\mathbf{D} + \mathbf{I})^{-\frac{1}{2}}\mathbf{E}^{(k)},$$

Trong đó, $I \in R^{(M+N)(M+N)}$ là một ma trận định danh (identity matrix), được cộng vào ma trận A để tạo các kết nối self-connection. Để đơn giản hóa, LightGCN không sử self-connection, công việc chỉ là rescale lại các embedding. Trong SGCN, ma trận final embedding được biểu diễn như sau:

$$\begin{aligned}\mathbf{E}^{(K)} &= (\mathbf{A} + \mathbf{I})\mathbf{E}^{(K-1)} = (\mathbf{A} + \mathbf{I})^K\mathbf{E}^{(0)} \\ &= \binom{K}{0}\mathbf{E}^{(0)} + \binom{K}{1}\mathbf{A}\mathbf{E}^{(0)} + \binom{K}{2}\mathbf{A}^2\mathbf{E}^{(0)} + \ldots + \binom{K}{K}\mathbf{A}^K\mathbf{E}^{(0)}.\end{aligned}$$

Biểu thức trên cho thấy việc thêm các self-connection về cơ bản tương đương với việc sử dụng tổng trọng số như trong LightGCN.

3.3.4.2. Mối quan hệ với APPNP

Approximate Personalized Propagation of Neural Predictions (APPNP) bằng cách sử dụng kiến trúc Personalized PageRank giúp mạng lan truyền trong phạm vi dài mà không gặp phải hiện tượng oversmoothing (embedding của các nút trở nên giống nhau khi độ sâu của mạng tăng lên):

$$\mathbf{E}^{(k+1)} = \beta\mathbf{E}^{(0)} + (1 - \beta)\tilde{\mathbf{A}}\mathbf{E}^{(k)},$$

Trong đó β là xác suất để giữ lại các đặc trưng ban đầu khi thực hiện lan truyền, $\tilde{A}$ là ma trận kề đã được chuẩn hóa. Ma trận final embedding được tính như sau:

$$\begin{aligned}\mathbf{E}^{(K)} &= \beta\mathbf{E}^{(0)} + (1 - \beta)\tilde{\mathbf{A}}\mathbf{E}^{(K-1)}, \\ &= \beta\mathbf{E}^{(0)} + \beta(1 - \beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + (1 - \beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(K-2)} \\ &= \beta\mathbf{E}^{(0)} + \beta(1 - \beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \beta(1 - \beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + (1 - \beta)^K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}.\end{aligned}$$

Liên hệ với LightGCN, với việc sử dụng hệ số α_k cho mỗi layer, chúng ta có thể kiểm soát và khôi phục những embedding từ những tầng trước đó, giúp ngăn chặn được hiện tượng oversmoothing nếu chọn được giá trị α_k phù hợp.

3.3.4.3. Second-Order Embedding Smoothness

Để chứng minh các phép biến đổi tuyến tính đơn giản trong LightGCN đem lại hiệu quả tốt trong việc học được các biểu diễn của user và item. Lấy ví dụ với 2 layer của LightGCN:

$$\mathbf{e}_u^{(2)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{|\mathcal{N}_i|} \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}} \mathbf{e}_v^{(0)}.$$

Chúng ta có thể thấy rằng, nếu một người dùng khác cùng tương tác với người dùng u, smoothness strength của v trên u được tính bằng hệ số:

$$c_{v->u} = \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}} \sum_{i \in \mathcal{N}_u \cap \mathcal{N}_v} \frac{1}{|\mathcal{N}_i|}.$$

Công thức trên cho thấy độ ảnh hưởng của một hàng xóm bậc 2 v trên u được xác định bởi 3 yếu tố: tỉ lệ thuận với số lượng item cùng tương tác bởi u và v; mức độ phổ biến item đồng tương tác giảm dần (thể hiện tính cá nhân hóa của user) và cuối cùng là tỉ lệ nghịch với mức độ hoạt động của v. Khả năng diễn giải tốt như trên phục vụ tốt cho việc đo lường mức độ tương tự của các user trong bài toán CF. Các phân tích hoàn toàn tương tự với item do công thức đối xứng của LightGCN.

3.3.5. Model Training

Hàm loss sử dụng Bayesian Personalized Ranking (BPR) loss:

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||\mathbf{E}^{(0)}||^2$$

Trong đó λ là tham số L_2 regularization, hàm tối ưu sử dụng là Adam với mini-batch, một số chiến lược negative sampling cũng giúp cải thiện hiệu suất mô hình như hard negative sampling hay adversarial sampling. Dropout được sử dụng phổ biến trong GCN hay NGCF tuy nhiên không được sử dụng trong LightGCN do việc sử dụng L2 regularization đã đủ để ngăn chặn overfitting.

4. Mô tả tập dữ liệu

4.1. Giới thiệu

Bộ dữ liệu chúng em sử dụng trong bài toán này là Cross-Market Recommendation (XMRec). Trong đó có ba thị trường nguồn là s1, s2, s3 và hai thị trường mục tiêu t1, t2. Mục tiêu của bài toán là tối ưu hệ thống gợi ý trên thị trường mục tiêu t1, t2 có thể sử dụng dữ liệu của các thị trường nguồn s1, s2 và s3.

```
DATA
├── s1
│   └── [1.8M]  train_5core.tsv       /* train data */
│   └── [19.2M]  train.tsv       /* full train data */
│   └── [139K]  valid_qrel.tsv       /* validation positive samples */
│   └── [5.6M]  valid_run.tsv       /* list of validation items to be reranked */
├── s2
│   └── [1.1M]  train_5core.tsv       /* train data */
│   └── [2.6M]  train.tsv       /* full train data */
│   └── [153K]  valid_qrel.tsv       /* validation positive samples */
│   └── [6.2M]  valid_run.tsv       /* list of validation items to be reranked */
├── s3
│   └── [548K]  train_5core.tsv       /* train data */
│   └── [1.2M]  train.tsv       /* full train data */
│   └── [71K]  valid_qrel.tsv       /* validation positive samples */
│   └── [2.9M]  valid_run.tsv       /* list of validation items to be reranked */
├── t1
│   ├── [2.3M]  test_run.tsv    /* list of test items to be reranked */
│   ├── [1.4M]  train.tsv       /* full train data */
│   ├── [457K]  train_5core.tsv       /* train data */
│   ├── [58K]  valid_qrel.tsv   /* validation positive samples */
│   └── [2.3M]  valid_run.tsv    /* list of validation items to be reranked */
└── t2
    ├── [4.8M]  test_run.tsv    /* list of test items to be reranked */
    ├── [2.8M]  train.tsv       /* full train data */
    ├── [966K]  train_5core.tsv       /* train data */
    ├── [118K]  valid_qrel.tsv   /* validation positive samples */
    └── [4.8M]  valid_run.tsv    /* list of validation items to be reranked */
```

- train.tsv: bao gồm userId, itemId và rating từ 1 đến 5
- train_5score: đã được tiến hành một số bước tiền xử lý để tạo điều kiện thuận lợi cho gánh nặng tiền xử lý dữ liệu. Dữ liệu được chuẩn hóa thành xếp hạng 0 và 1.

- Valid_qrel: chứa các đánh giá tích cực về sản phẩm của người dùng
- Valid_run: chứa các đánh giá khẳng định và phủ định của người dùng. Một người dùng có 100 đánh giá về 100 sản phẩm, cứ 1 đánh giá tích cực thì có 99 đánh giá tiêu cực
- Tập test_qrel và test_run tương tự valid_qrel và valid_run nhưng chỉ sử dụng làm tập test.

➢ Như chúng ta thấy, bộ dữ liệu này vô cùng đa dạng các thuộc tính để có thể khai thác và kết hợp nhằm tăng độ chính xác cho mô hình.

4.2. EDA data

4.2.1. Train.tsv

- Thống kê độ lớn của mỗi tập dữ liệu:
  - S1 train.tsv: 840212 đánh giá, 141491 user, 34648 item
  - S2 train.tsv: 115549 đánh giá, 118875 user, 4960 item
  - S3 train.tsv: 54202 đánh giá, 8601 user, 2797 item
  - T1 train.tsv: 60168 đánh giá, 9742 user, 3429 user
  - T2 train.tsv: 120320 đánh giá, 18242 user, 8834

➢ Các tập nguồn nhỏ dần nhằm tạo sự tiện lợi cho quá trình huấn luyện mô hình phù hợp với nhiều loại phần cứng. Nếu có nhiều tài nguyên thì có thể sử dụng nhiều bộ dữ liệu và ngược lại. Trong bài toán này, chúng ta sử dụng cả ba bộ dữ liệu để huấn luyện mô hình.

- Các trường thuộc tính trong tập train: Cột thứ nhất là ID của người dùng, cột thứ hai là ID của sản phẩm và cột thứ ba là xếp hạng của người dùng cho sản phẩm.

	userId	itemId	rating
0	s1U1031263	P1004242	1.0
1	s1U1010792	P1028037	1.0
2	s1U1080963	P1011533	4.0
3	s1U1023573	P1006496	5.0
4	s1U1125010	P1029882	5.0

- Xem xét ratings của từng tập S1, S2, S3, T1, T2:

```
s1
              rating
count  840212.000000
mean        4.282325
std         1.188025
min         1.000000
25%         4.000000
50%         5.000000
75%         5.000000
max         5.000000
```

```
s2
              rating
count  115549.000000
mean        4.407308
std         1.135887
min         1.000000
25%         4.000000
50%         5.000000
75%         5.000000
max         5.000000
```

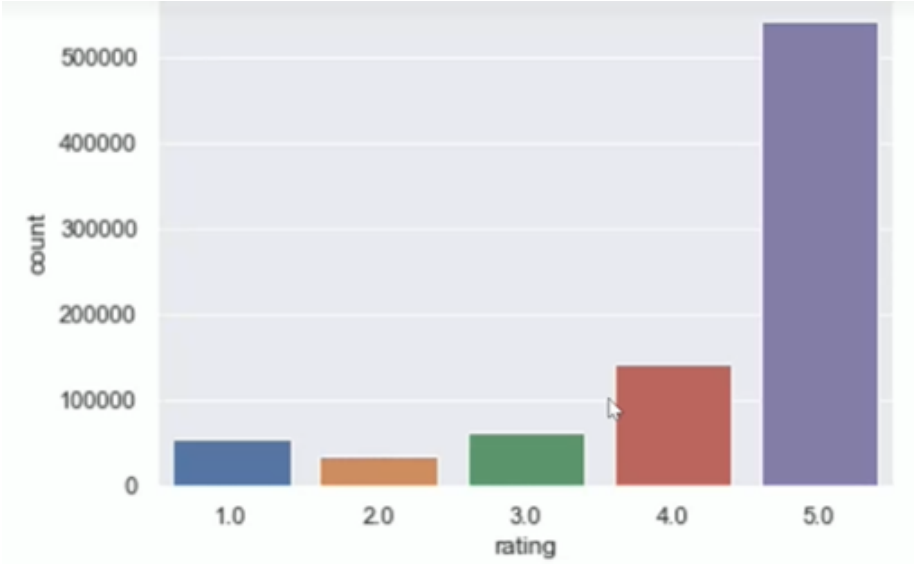
```
s3
              rating
count  54202.000000
mean       4.404468
std        1.042800
min        1.000000
25%        4.000000
50%        5.000000
75%        5.000000
max        5.000000
```

```
t1
              rating
count  60168.000000
mean       4.415337
std        1.123550
min        1.000000
25%        4.000000
50%        5.000000
75%        5.000000
max        5.000000
```

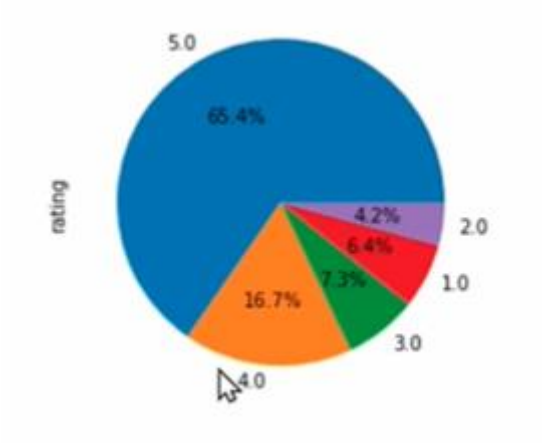
```
t2
              rating
count  120320.000000
mean        4.431799
std         1.063725
min         1.000000
25%         4.000000
50%         5.000000
75%         5.000000
max         5.000000
```

Nhận xét: Các tập đều có mean ratings rất lớn 4.2 – 4.4, và không có phương sai quá lớn, tầm 1.0 – 1.1 → Ratings từ cả 5 tập có xu hướng lệch, đa số là ratings tốt

- Chúng ta khảo sát với tập train của thị trường nguồn S1:



Hình 5. Số lượng của mỗi loại Rating

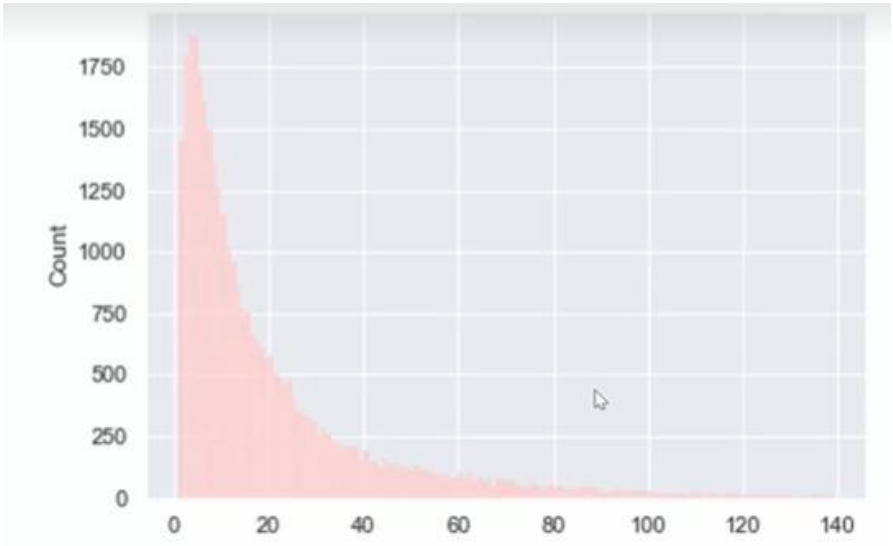


Hình 6. Biểu đồ tỉ lệ của mỗi loại Rating

- Như chúng ta đã thấy ở hai biểu đồ trên, số lượng Rating là 5 chiếm chủ yếu (65,4%). Rating trung bình là 4.3, đây là một con số khá cao. Cho thấy rằng trong thị trường s1 này khi người dùng đã mua và đánh giá sản phẩm thì thường đánh giá khá cao.

Hình 7. Biểu đồ thống kê số lượng đánh giá của mỗi người dùng

- Mỗi user đánh giá từ 4-137 item và trung bình mỗi user đánh giá 5,7 item. Bộ dữ liệu cũng đã rất cố gắng thu thập những người dùng đánh ít nhất 4 sản phẩm để tránh trường hợp quá ít dữ liệu để gợi ý cho người dùng ấy.



Hình 8. Biểu đồ thống kê số lượng vote của mỗi item

4.2.2. train_5score và valid_quel

Đây là 2 tập dữ liệu nhóm sử dụng để xây dựng mô hình bài toán. Trong đó train_5core.tsv dùng để huấn luyện mô hình còn valid_qrel.tsv dùng để tuning mô hình. Đặc điểm chung của 2 bộ dữ liệu là chỉ có những tương tác tốt, rating chỉ có 1.0. Ngoài ra valid_qrel.tsv cũng chỉ có ở tập t1 và t2 do phục vụ cho tuning

```
userId  itemId  rating
t1U1007418  P1012632    1.0
t1U1005548  P1027537    1.0
t1U1002792  P1008806    1.0
t1U1007846  P1008776    1.0
t1U1009113  P1026544    1.0
t1U1006112  P1034611    1.0
t1U1004231  P1027433    1.0
t1U1004410  P1025338    1.0
```

- Thống kê về độ lớn của các tập:
  - S1 train_5core.tsv: 77173 đánh giá, 6466 user, 9762 item
  - S2 train.tsv: 48302 đánh giá, 7109 user, 2198 item
  - S3 train.tsv: 23367 đánh giá, 3328 user, 1245 item
  - T1 train.tsv: 19615 đánh giá, 2697 user, 1357 user
  - T2 train.tsv: 41226 đánh giá, 5482 user, 2917

5. Các độ đo đánh giá

Trong bài toán này chúng ta sử dụng nDCG@10 và HR@10 (lấy top 10) để đánh giá độ chính xác của hai thị trường mục tiêu t1 và t2.

5.1. NDCG (Normailized Discounted Cumulative Gain)

DCG là một độ đo chất lượng xếp hạng. Đo lường dựa trên vị trí của đối tượng trong danh sách trả về.

Ý tưởng đằng sau NDCG khá đơn giản: 1 hệ gợi ý trả về 1 danh sách gợi ý gồm các *item* và ta muốn tính toán chất lượng của danh sách đó. Mỗi *item* có một điểm phù hợp (relevance score), thường là một số không âm, gọi là *gain*. Đối với những *item* mà không có đánh giá từ *user* ta thường đặt *gain* bằng 0.

Cumulative gain (CG) được định nghĩa là tổng các *gain* của các *item* trong danh sách gợi ý. CG ở vị trí *k* được xác định:

$$CG_k = \sum_{i=1}^{k} G_i$$

trong đó: G_i *là gain* của *item* ở vị trị thứ *i* trong danh sách gợi ý.

Rõ ràng, CG không cân nhắc đến thứ tự của các *item* trong danh sách gợi ý, nó không bị ảnh hường khi đổi chỗ thứ tự của 2 *item* bất kỳ. Đây là một vấn đề khi thứ tự xếp hạng là quan trọng, chẳng hạn, trong kết quả tìm kiếm của Google Search, ta sẽ không muốn các trang web liên quan nhất với câu truy vấn được đặt ở bên dưới kết quả tìm kiếm.

Để "phạt" các *item* có độ phù hợp cao hơn bị đặt thấp hơn trong danh sách gợi ý, ta chia *gain* của *item* cho xếp hạng của nó trong danh sách gợi ý. DCG (Discounted Cumulative Gain) được giới thiệu:

$$DCG_k = \sum_{i=1}^{k} \frac{G_i}{log_2(i+1)}$$

Vấn có hạn chế với DCG, đó là DCG phụ thuộc vào độ dài của danh sách gợi ý, độ dài danh sách gợi ý khác nhau với các *user* khác nhau. Do đó, ta không thể so sánh hiệu suất hệ gợi ý một cách nhất quán từ *user* này sang *user* khác nếu chỉ dùng DCG. Vì vậy, ta cần chuẩn hóa DCG tại mỗi vị trí ứng với giá trị *k* đã chọn trên các *user*.

Thực hiện điều này bằng cách sắp xếp tất cả các *item phù hợp* trong kho *item* theo mức độ phù hợp của chúng, giúp tạo ra DCG *lớn nhất có thể* thông qua vị trí *k*, hay còn gọi là Ideal DCG (IDCG) thông qua vị trí *k*. Đối với một *user*, NDCG được xác định như sau:

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

trong đó, IDCG được xác định:

$$IDCG_k = \sum_{i=1}^{|I_k|} \frac{G_i}{log_2(i+1)}$$

và I_k biểu diễn danh sách các *item (*được sắp xếp theo mức độ phù hợp) trong kho *item* đến vị trí k.

Giá trị NDCG cho tất cả *user* được lấy trung bình để làm thước đo hiệu suất trung bình của hệ gợi ý. Một hệ gợi ý hoàn hảo có DCG giống với IDCG và NDCG bằng 1.

Khó khăn chính trong việc sử dụng NDCG là *không có sẵn một thứ tự lý tưởng của kết quả* khi chỉ có *một phần* đánh giá của *user* về độ phù hợp của các *item.*

5.2. HR (Hit Ratio)

HR là tỷ lệ *user* có *item chính xác* được đưa vào danh sách đề xuất có độ dài L.

$$HR = \frac{|U_{hit}^L|}{|U_{all}|}$$

trong đó: U_{hit}^L ∨ là số lượng *user* có *item chính xác được đưa vào danh sách đề xuất top L*, U_{all} ∨ là tổng số người dùng trong tập dữ liệu kiểm thử.

Có thể thấy, L càng lớn thì HR càng cao, bởi vì càng có nhiều khả năng *item chính xác* được đưa vào danh sách gợi ý. Vì vậy, cần chọn một giá trị L thích hợp.

6. Thiết lập thí nghiệm và đánh giá kết quả thực nghiệm

Thí nghiệm 1: Thay đổi bộ dữ liệu sử dụng để huấn luyện mô hình.

Với bộ dữ liệu này chúng ta có ba thị trường nguồn s1, s2, s3 và hai thị trường mục tiêu t1 và t2. Chúng ta sẽ xây dựng ra hai mô hình lần lượt cho hai thị trường t1, t2 và turning tham số với từng mô hình:

- Thị trường t1: đầu tiên chỉ huấn luyện một mô hình gợi ý với tập train của thị trường t1. Chúng ta tăng bộ dữ liệu huấn luyện lên dần dần và xây dựng các mô hình mới bằng cách thêm thị trường s1, s1 và s2, cả ba thị trường nguồn. Từ đó chúng ta cũng có thể so sánh được sự khác biệt giữa việc có sử dụng thị trường nguồn cho thị trường mục tiêu hay không sử dụng.
- Thị trường t2: Cũng tương tự thị trường t1. Chúng ta xây dựng 4 mô hình lần lượt sử dụng các bộ dữ liệu là t2; t2 và s1; t2, s1 và s2; t2, s1, s2 và s3.
- Nhóm chúng em sẽ so sánh với mô hình GMF++, mô hình mà bài toán với bộ dữ liệu này đề xuất.

Source / **Target**	**None**		**S1**		**S1-S2**		**S1-S2-S3**	
	ndcg10	hr10	ndcg10	hr10	ndcg10	hr10	ndcg10	hr10
T1(GMF++)	0.3194	0.4976	0.2919	0.4928	None	None	0.2967	0.5083
T1(lightGCN)	0.5730	0.7278	**0.5829**	0.7353	0.5706	0.7379	0.5706	**0.7419**
T2(GMF++)	0.3203	0.4850	0.2963	0.4670	None	None	0.1257	0.1476
T2(lightGCN)	0.5179	0.6505	**0.5250**	0.6600	0.5245	0.6605	0.5228	**0.6627**

T1+T2(GMF+)	0.3200	0.4892	0.2949	0.4755	None	None	0.2301	0.3695
T1+T2(lightGCN)	0.5361	0.6760	0.**5540**	0.6849	0.5397	0.6860	0.5386	**0.6889**

- Sau khi thiết lập thí nghiệm chúng ta có thể nhận thấy rằng ở mỗi thị trường, khi sử dụng dữ liệu đánh giá của thị trường nguồn s1 sẽ thu được kết quả tốt nhất. Chúng ta sẽ sử dụng bộ dữ liệu có kết quả tốt nhất này để huấn luyện mô hình cho các thí nghiệm sau.

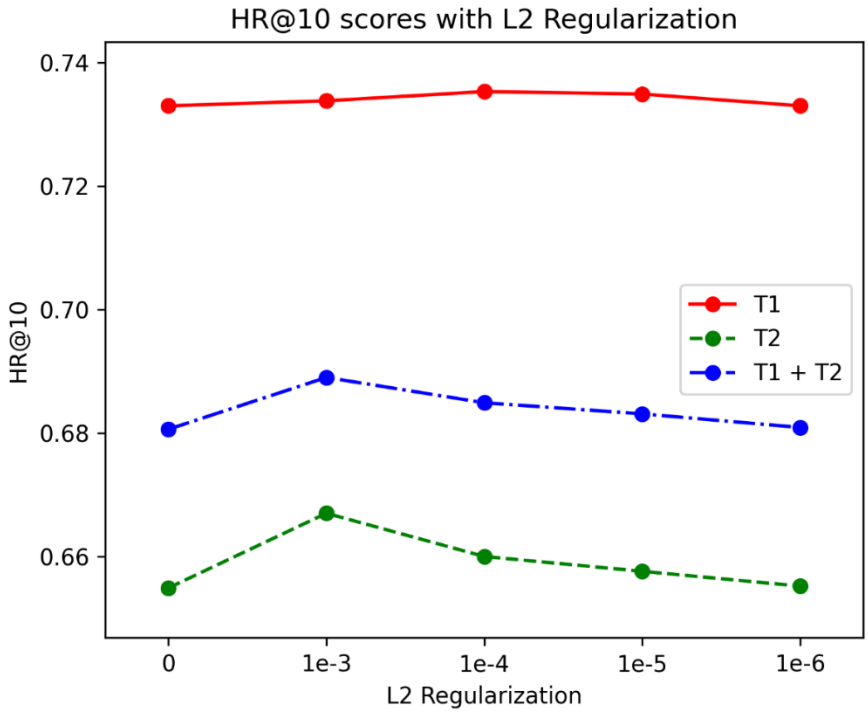
Thí nghiệm 2: Thay đổi tham số λ (tham số L_2 regularization) trong hàm loss Bayesian Personalized Ranking (BPR). Đây là một tham số rất quan trọng trong mô hình LightGCN này. L2 regularization nhằm giúp mô hình tránh bị overfitting.

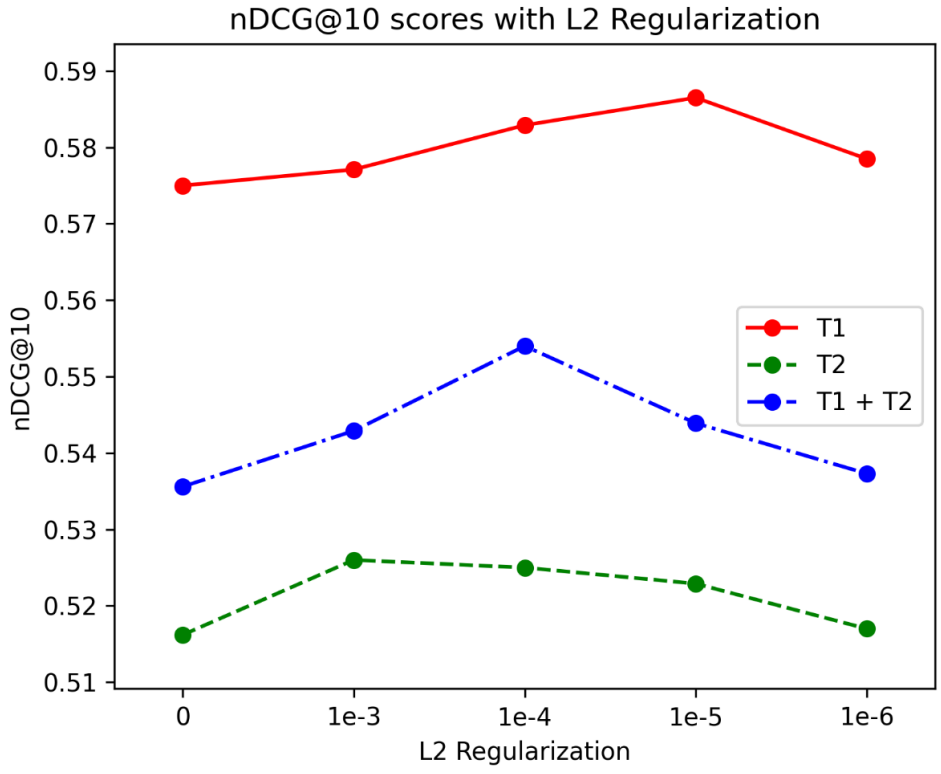
Với mô hình của từng thị trường, chúng ta sẽ khảo sát tham số λ trong những giá trị $\{ 1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3} \}$

L2_reg / Target	0		1e-3		1e-4		1e-5		1e-6	
	ndcg10	hr10	ndcg10	hr10	ndcg10	hr10	ndcg10	hr10	ndcg10	hr10
T1	0.5750	0.7330	0.5771	0.7338	0.5829	**0.7353**	**0.5865**	0.7349	0.5785	0.7330
T2	0.5162	0.6549	**0.5260**	**0.6670**	0.5250	0.6600	0.5229	0.6576	0.5170	0.6552
T1+T2	0.5356	0.6806	0.5429	**0.6890**	**0.5540**	0.6849	0.5439	0.6831	0.5373	0.6809

- Sau khi xây dựng mô hình với từng giá trị của tham số λ, chúng ta thu được mô hình đạt kết quả tốt nhất với $\lambda = 1e^{-4}$

Trực quan kết quả tuning L2_reg:





Sau hai thí nghiệm, mô hình tốt nhất của mỗi thị trường chúng em nhận được có bộ dữ liệu sử dụng cả thị trường mục tiêu và ba thị trường nguồn, tham số λ có giá trị là $1e^{-4}$. Kết quả tốt nhất của các độ đo mà mô hình chúng em đạt được là:

T1+T2 nDCG@10 (test)	T1+T2 HR@10 (test)	T1+T2 nDCG@10 (validation)	T1+T2 HR@10 (validation)	T1 nDCG@10 (test)	T2 nDCG@10 (test)
0.5472	0.6863	0.8180	0.9880	0.5845	0.5288

7. Kết luận và hướng phát triển

Mục tiêu của LightGCN là tận dụng các kết nối bậc cao trong biểu đồ sản phẩm-người dùng. Chìa khóa của LightGCN là lớp lan truyền embedding. Các thử nghiệm trên bộ tập dữ liệu trên chứng minh tính hợp lý và hiệu quả của việc đưa cấu trúc graph user-item vào quy trình embedding. LightGCN đã cải tiến và loại bỏ thiết kế phức tạp không cần thiết của GCN để lọc cộng tác. LightGCN bao gồm hai thành phần thiết yếu - light graph convolution và layer combination. Trong light graph convolution loại bỏ feature transformation và nonlinear activation - hai phép toán tiêu chuẩn trong GCN nhưng chắc chắn sẽ làm tăng độ khó đào tạo. Trong kết hợp layer, lần embedding cuối cùng của node dưới dạng tổng trọng số của các lần embedding của nó trên tất cả các layer. Điểm mạnh của LightGCN là đơn giản: dễ đào tạo hơn, khả năng khái quát hóa tốt hơn và hiệu quả hơn.

Với sự phổ biến của dữ liệu đồ thị được liên kết trong các ứng dụng thực tế, các mô hình dựa trên đồ thị ngày càng trở nên quan trọng trong việc hệ gợi ý; bằng cách khai thác rõ ràng các mối quan hệ giữa các thực thể trong mô hình dự đoán, chúng có lợi cho học có giám sát truyền thống. Ví dụ, một xu hướng gần đây là khai thác các thông tin bổ trợ như đồ thị kiến thức item, mạng xã hội và nội dung đa phương tiện để gợi ý, trong đó các GCN đã thiết lập SOTA.

Trong bài toán này, chúng em tìm hiểu rất nhiều paper những chỉ xây dựng mô hình theo một paper đạt SOTA để có thể tìm hiểu sâu và turning một cách kỹ lưỡng các siêu tham số của mô hình nhằm đạt kết quả tốt nhất.

Hướng phát triển trong tương lai:

- Trong tương lai, có thể cải thiện LightGCN hơn nữa bằng cách kết hợp cơ chế attention
- Chúng ta còn có thể tùy biến lượng layer theo từng người dùng. Ví dụ: người dùng ít tương tác có thể cần nhiều tín hiệu hơn từ những người hàng xóm cấp cao hơn trong khi người dùng nhiều tương tác yêu cầu ít hơn.
- Để khai thác một cách triệt để bộ dữ liệu nhằm tạo ra mô hình gợi ý tốt nhất cho thị trường mục tiêu, chúng ta còn cần phải kết hợp các loại dữ liệu và tạo ra một mô hình tổng quát

Tài liệu tham khảo

1. Paper: Neural Graph Collaborative Filtering, Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua, [Submitted on 20 May 2019 (v1), last revised 3 Jul 2020 (this version, v2)]
2. Paper: LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, Meng Wang, Submitted on 6 Feb 2020 (v1), last revised 7 Jul 2020 (this version, v4)
3. https://towardsdatascience.com/ranking-evaluation-metrics-for-recommender-systems-263d0a66ef54
4. https://en.wikipedia.org/wiki/Discounted_cumulative_gain
5. https://www.dataminingapps.com/2016/04/what-are-popular-ways-of-evaluating-recommender-systems/