TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA TOÁN TIN - HỌC

-----ꕥ🕮ꕥ----

BÁO CÁO ĐỒ ÁN

Hệ thống Tư vấn - MTH10623

Đề tài: "Xây dựng hệ thống tư vấn sản phẩm dựa trên đồ thị"

Giáo viên hướng dẫn: ThS. Nguyễn Thanh Sơn

Nhóm thực hiện:

1. Huỳnh Quang Trung - 20280108
2. Nguyễn Thị Huyền – 20280048
3. Trần Tuấn Thái - 20280082

TP. Hồ Chí Minh - 2023

MỤC LỤC

I. GIỚI THIỆU

1. Giới thiệu chung và mục tiêu của đề tài

Hệ thống gợi ý đóng một vai trò quan trọng trong việc giới thiệu sản phẩm trên các thị trường trực tuyến và kết nối trên mạng xã hội. Sự đề xuất chất lượng cao không chỉ là yếu tố chính để tăng cường sự hài lòng của người tiêu dùng và tăng doanh thu bán hàng, mà còn là yếu tố quyết định sự tương tác của người dùng.

Các phương pháp hệ thống gợi ý truyền thống sử dụng các độ đo tương đồng để đề xuất các sản phẩm khác thông qua các phương pháp lọc dựa trên nội dung và cộng tác. Tuy nhiên, dữ liệu sản phẩm có thể được biểu diễn dưới dạng đồ thị phi Euclid với các mối quan hệ như sản phẩm được mua cùng nhau hoặc xem cùng nhau. Các phương pháp hệ thống gợi ý truyền thống không xem xét các mối quan hệ đồ thị giữa các nút sản phẩm theo cách mà các mạng nơ-ron đồ thị làm. So với phương pháp truyền thống, cách tiếp cận dựa trên biểu đồ để đề xuất sản phẩm đang đưa ra một bước tiến quan trọng. Thay vì chỉ xem xét số liệu tương tự, nó thực sự tính đến mối quan hệ đồ thị giữa các nút sản phẩm, tương tự như cách mạng thần kinh đồ thị hoạt động. Bằng cách này, việc sử dụng đầy đủ mối quan hệ giữa các nút sản phẩm giúp tạo ra các phần nhúng sản phẩm cải tiến, biểu thị một cách chính xác cấu trúc đồ thị phức tạp của dữ liệu sản phẩm.

Mục tiêu của đề tài là áp dụng mạng lưới thần kinh đồ thị để giải quyết những thách thức phức tạp và quan trọng của hệ thống gợi ý. Chúng tôi sẽ sử dụng 2 kiến trúc Graph Neural Netwwork (GNN) là Graph Convolutional Networks (GCN) và GraphSAGE để có thể sử dụng mối quan hệ đồ thị của dữ liệu sản phẩm tạo ra các embedding chính xác và mạnh mẽ cho việc gợi ý sản phẩm.

II. NỀN TẢNG LÝ THUYẾT

1. Graph Neural Network (GNN)

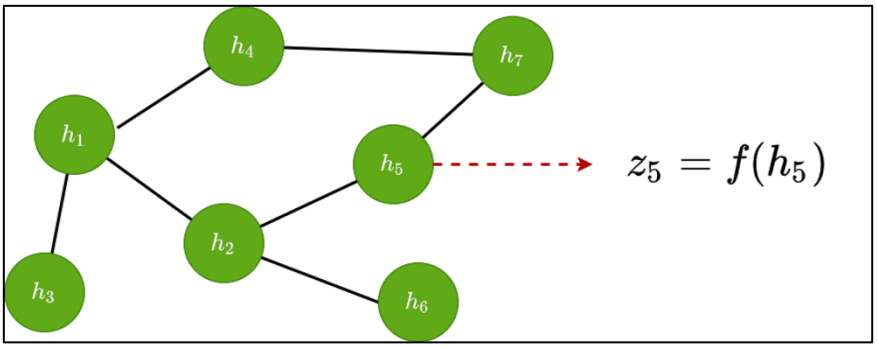
Mạng nơ-ron đồ thị (Graph Neural Networks - GNNs) là một dạng mạng nơ-ron nhân tạo được thiết kế để xử lý dữ liệu có cấu trúc dạng đồ thị. Chúng có khả năng học từ các mô hình đồ thị phức tạp và được sử dụng rộng rãi trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, dự đoán đồ thị, phân tích mạng xã hội và học máy hóa hóa học.

GNNs sử dụng các lớp nơ-ron được thiết kế đặc biệt để lan truyền thông tin giữa các đỉnh hoặc cạnh trong đồ thị. Qua nhiều lần lặp, chúng có khả năng tự cập nhật và học cách thức hoạt

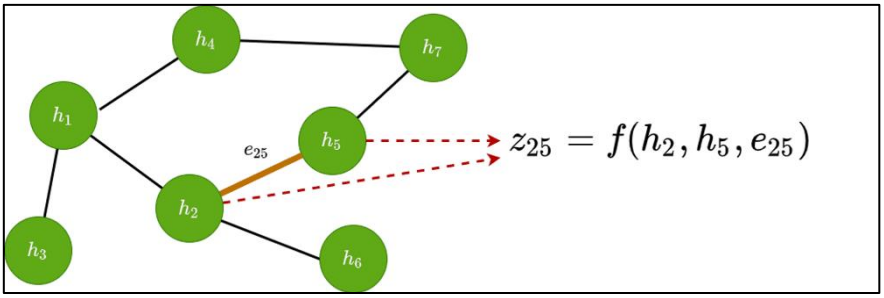
động của mạng dữ liệu. Ưu điểm lớn của GNNs là khả năng xử lý thông tin cấu trúc từ các đồ thị phức tạp, giúp chúng trở thành một công cụ mạnh mẽ trong việc hiểu và dự đoán các mô hình dữ liệu phức tạp, đặc biệt là trong việc phân tích mạng xã hội, dự đoán tương tác hóa học, và các ứng dụng về dự đoán đồ thị. Mục tiêu chính của kiến trúc GNN là học các biểu diễn ý nghĩa (embedding) cho mỗi nút trong đồ thị dựa trên mối quan hệ và thuộc tính của nó. Chúng ta có thể sử dụng các embedding này để giải quyết nhiều vấn đề khác nhau, bao gồm gán nhãn nút, dự đoán nút và cạnh, và nhiều hơn nữa.

GNN là mạng thần kinh được thiết kế để đưa ra dự đoán ở cấp độ nút, cạnh hoặc toàn bộ đồ thị:

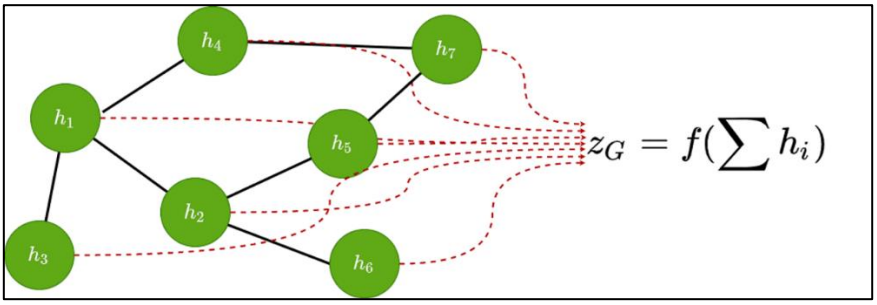
- **Cấp độ nút** có thể giải quyết một nhiệm vụ như phát hiện thư rác.



- **Cấp độ cạnh:** Để thực hiện điều này, chúng ta thường cần cả vectơ nút liền kề cũng như các đặc điểm cạnh nếu chúng tồn tại. Nhiệm vụ dự đoán cạnh có thể là dự đoán liên kết, một tình huống phổ biến trong các hệ thống tư vấn.

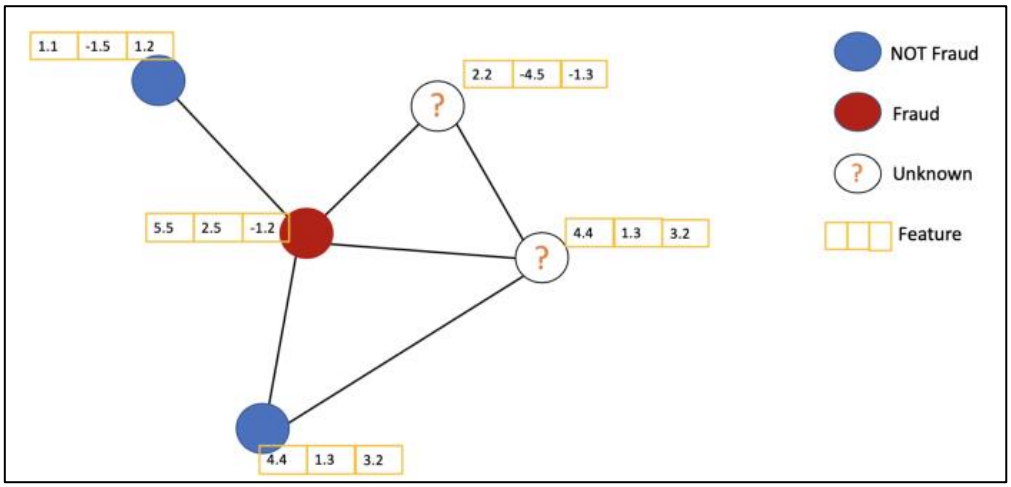


- **Cấp độ toàn đồ thị**: Bằng cách tổng hợp tất cả các đặc trưng của nút và áp dụng một hàm thích hợp f. Ví dụ: Nhiệm vụ dự đoán dựa trên đồ thị có thể là **dự đoán tính chất hóa học của đồ thị phân tử.**



2. Graph Convolution Network (GCN)

Mạng nơ-ron đồ thị (Graph Convolutional Network - GCN) là một loại mạng nơ-ron được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc dưới dạng đồ thị. Trong GCN, việc áp dụng các phép tích chập trên đồ thị giúp mô hình học được cách tương tác giữa các đỉnh và cạnh trong đồ thị.

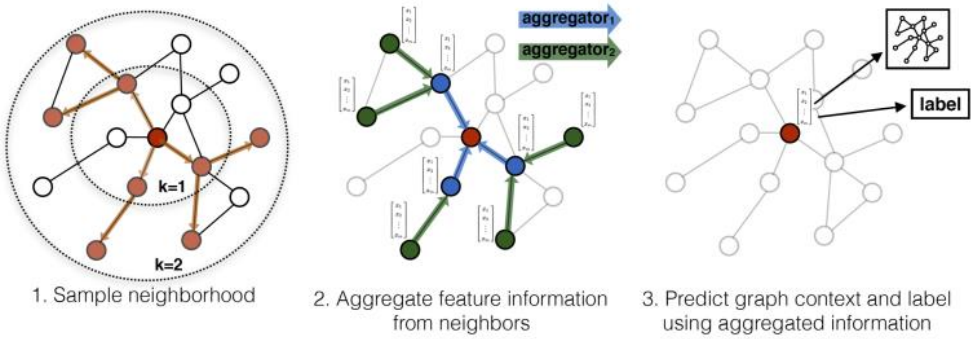


GCN hoạt động dựa trên một nguyên tắc cơ bản: Thông tin đặc trưng của mỗi nút được tổng hợp từ chính nó và tất cả các nút lân cận của nó. Để minh họa, chúng ta có thể xem xét việc sử dụng hàm trung bình (Average()). Quy trình này được thực hiện đồng thời cho tất cả các nút trong biểu đồ. Cuối cùng, các giá trị trung bình thu được từ quá trình này sẽ được đưa vào mạng nơ-ron để tiếp tục xử lý.

3. Thuật toán GraphSAGE

GraphSAGE là một giải thuật inductive learning dùng cho dữ liệu dạng đồ thị. Trong đó, inductive learning là quá trình học từ các tập dữ liệu huấn luyện sau đó suy ra các quy luật chung rồi áp dụng các quy luật này vào tập dữ liệu kiểm thử. Với cách huấn luyện này chúng ta không cần huấn luyện lại mạng đồ thị khi có dữ liệu mới, chưa từng xuất hiện nên phù hợp giải cho các bài toán tổng quát hóa cao như bài toán về mạng xã hội.

Mô hình GraphSAGE là một biến thể nhỏ của mô hình tích chập đồ thị GCN. Mô hình GraphSAGE lấy mẫu các hàng xóm của một nút mục tiêu và các đặc trưng của chúng, sau đó tổng hợp tất cả lại với nhau để học và dự đoán các đặc trưng của nút mục tiêu. Điều này cho phép thuật toán mở rộng độ sâu của thông tin thu thập từ hàng xóm, từ đó tạo ra các biểu diễn có ý nghĩa về cấu trúc và tính chất của đồ thị. GraphSAGE có thể được sử dụng cho các nhiệm vụ như phân loại đồ thị hoặc nút, dự đoán các thuộc tính của đỉnh, hay thậm chí là nhận diện cấu trúc của đồ thị.



Một điểm đặc biệt của GraphSAGE là khả năng huấn luyện cả ở chế độ không giám sát và giám sát. Trong quá trình huấn luyện, **thuật toán này cũng tối ưu hóa các hàm tổng hợp thông tin** từ hàng xóm cùng với các ma trận trọng số, giúp mô hình tự học cách kết hợp thông tin từ các nút láng giềng một cách hiệu quả. Điều này giúp GraphSAGE trở thành một công cụ mạnh mẽ trong việc biểu diễn đồ thị và xử lý dữ liệu đồ thị lớn.

Giải thuật thực hiện bằng cách huấn luyện một tập hợp các hàm gọi là aggregator function giúp tổng hợp các thông tin đặc trưng từ các nút hàng xóm. Mỗi aggregator function tổng hợp đặc trưng

Giải thuật sinh ma trận embedding: Các ký hiệu quy ước:

- **K**: Search depth, số lớp của mô hình
- **AGGREGATE**$_k$,$\forall k \in \{1,...,\mathbf{K}\}$: K hàm aggregator
- $\mathbf{W}^k$,$\forall k \in \{1,...,\mathbf{K}\}$: Ma trận trọng số dùng cho mỗi lớp của mô hình
- $\mathbf{N}_v$: tập hợp các nút hàng xóm của nút v

- $\mathbf{X}_v, \forall v \in \mathbf{V}$: feature của toàn bộ các nút

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Hàm mất mát của giải thuật GraphSage: Các hàm mất mát dựa trên cơ chế khuyến khích các nút gần nhau sẽ có biểu diễn tương tự nhau trong khi các nút khác nhau sẽ có biểu diễn khác nhau.

$$J_{\mathcal{G}}(z_u) = -log(\sigma(z_u^{\bar{\tau}} z_v)) - Q \cdot E_{v_n \sim P_n(v)} log(\sigma(-z_u^{\bar{\bar{\tau}}} z_{v_n}))$$

Trong đó:

- v là một nút gần u được lấy ngẫu nhiên theo random walk có chiều dài cố định.
- $\mathbf{P}_n$: negative sampling distribution
- **Q**: số lượng negative samples hay các nút không kề nhau.

Ta có thể thấy với việc sử dụng hai nút không kề nhau một cách ngẫu nhiên đã khiến cho mô hình có tính tổng quát cao hơn đối đặc biệt khi dự đoán vào các dữ liệu chưa được huấn luyện.

Các kiến trúc hàm tổng hợp phổ biến: Không giống với các dạng dữ liệu như câu văn, ảnh v.v.., các nút hàng xóm không có thứ tự, do đó các hàm tổng hợp phải có tính đối xứng tức là bất biến đối với sự thay đổi thứ tự trong input đầu vào trong khi vẫn có thể huấn luyện và có

khả năng biểu diễn, trích xuất đặc trưng cao. Một số loại hàm tổng hợp: Mean aggregator, Pooling aggregator, LSTM aggregator.

III. XÂY DỰNG HỆ THỐNG TƯ VẤN DỰA TRÊN ĐỒ THỊ

1. Giới thiệu về tập dữ liệu

Bộ dữ liệu Amazon product data của Julian McAuley từ UCSD là một trong những bộ dữ liệu lớn và phổ biến được sử dụng trong lĩnh vực học máy và khai phá dữ liệu. Bộ dữ liệu này rất hữu ích để dùng trong nghiên cứu về các mô hình dự đoán đánh giá sản phẩm, phân loại sản phẩm và xây dựng các hệ thống gợi ý sản phẩm cho người dùng.

Với 142,8 triệu đánh giá kéo dài từ tháng 5 năm 1996 đến tháng 7 năm 2014 (Đối với phiên bản 2014), bộ dữ liệu này chứa các đánh giá sản phẩm (bao gồm điểm số rating, các đoạn văn bản, số lượt bình chọn hữu ích) và metadata (bao gồm mô tả, thông tin danh mục, giá cả, thương hiệu và đặc trưng hình ảnh của các sản phẩm) từ Amazon.

Toàn bộ bộ dữ liệu được chia thành các bộ dữ liệu con theo từng danh mục, ví dụ như Amazon-Books, Amazon-Instant Video và Amazon-Electronics. Các bộ dữ liệu con trong Amazon thường được sử dụng để kiểm tra hiệu suất trong việc lọc cộng tác người dùng-sản phẩm và đề xuất tuần tự.

Tập dữ liệu được sử dụng trong bài làm này chính là tập dữ liệu metadata meta_Office_Products (phiên bản năm 2014 của Amazon Product data), nằm trong danh mục văn phòng phẩm.

Đường link lưu trữ bộ dữ liệu: https://jmcauley.ucsd.edu/data/amazon/links.html

2. Tiền xử lý dữ liệu

Đây là tập dữ liệu trong định dạng JSON, dữ liệu mô tả thông tin về một sản phẩm trên Amazon bao gồm các thuộc tính:

```
{'asin': '043950578X',
'categories': [['Office Products', 'Office & School Supplies']],
'description': 'Reproducible pages on the back of each chart!',
'title': 'Scholastic Reading Is Succeeding Incentive Chart
(TF2204)',
'price': 2.49,
'imUrl': 'http://ecx.images-
amazon.com/images/I/61vJYM%2BMq2L._SY300_.jpg',
'brand': 'Scholastic',
'related': {'also_bought': ['B00207MG4Y', 'B00207MGDA'],
'also_viewed': ['B0007WXPWQ', 'B003VQH562']}}
```

Trong đó:

- asin: Là một mã số định danh duy nhất cho từng sản phẩm
- title: Tên sản phẩm
- price: Giá sản phẩm theo đơn vị USD
- imUrl: Đường link hình ảnh của sản phẩm
- related: Các sản phẩm có liên quan, bao gồm: also bought, also viewed, bought together, buy after viewing.
- salesRank: Thông tin xếp hạng bán hàng
- brand: Nhãn hiệu sản phẩm
- categories: Danh sách chứa các danh mục của sản phẩm

Do đó, trước tiên chúng ta sẽ chuyển dữ liệu từ định dạng JSON sang dạng pandas DataFrame.

Trong bộ dữ liệu metadata, các sản phẩm được xác định bằng các số nhận dạng chuẩn duy nhất của Amazon, hay còn gọi là ASINs. Chúng tôi loại bỏ tất cả các dòng dữ liệu bị thiếu (NaN), vì đây là tập dữ liệu chứa các đặc điểm của sản phẩm nên không thể sử dụng một phương pháp ước lượng nào để điền vào các điểm dữ liệu bị khuyết.

Sau đó, chúng tôi tiến hành lọc dữ liệu để giữ lại các dòng sản phẩm có chứa mục thông tin also bought trong cột related. Trong đó, also bought mô tả tất cả các ASINs của các sản phẩm được mua cùng với sản phẩm cụ thể đó. Những sản phẩm trong mục also_bought sẽ là các nút mục tiêu trong danh sách cạnh của chúng tôi.

Đồng thời chúng tôi cũng sử dụng thông tin also_bought trong cột related để tạo ra một cột mới độc lập trong dataframe là also_bought và loại bỏ tất cả các sản phẩm also_bought không tồn tại trong tập dữ liệu. Nhưng trong quá trình loại bỏ các ASIN nằm bên ngoài tập dữ liệu, sẽ vô tình làm mất đi một số do ASIN sản phẩm nằm trong dữ liệu ban đầu do sử dụng phương thức drop NaN của pandas. Như minh họa ở hình bên dưới, sản phẩm C đã vô tình bị mất đi trong quá trình xử lý. Nhiệm vụ của chúng tôi là tái tạo lại thông tin của sản phẩm C, với also_bought là sản phẩm A.

ASIN	Also bought
A	B, E, D, C
B	E, H, D
C	I, U
D	A, B

→

ASIN	Also bought
A	B, D, C
B	D
~~C~~	
D	A, B

Tiếp theo, thông qua cột danh mục - categories, bằng cách trích xuất ra một danh mục con, chúng tôi có được danh mục cụ thể của sản phẩm nhằm mục đích tạo label cho sản phẩm này. Cuối cùng, chúng tôi tạo lại những dòng sản phẩm đã vô tình bị mất đi trong quá trình xử lý như đã đề cập ở trước đó.

3. Xây dựng đồ thị (Graph)

Trong ngữ cảnh của đề tài này, chúng tôi chỉ xây dựng một đồ thị đồng nhất (homogenous graph) bao gồm các sản phẩm dưới dạng các nút và các cạnh được kết nối với nhau với xem xét các nút đó có được mua cùng nhau hay không:

- Node: Sản phẩm
- Edge: Cạnh nối các sản phẩm được mua cùng nhau
- Feature node: Tiêu đề (title), mô tả (description) của sản phẩm.

Đầu tiên chúng tôi lấy ra danh sách các cạnh trong đồ thị với node nguồn là sản phẩm ở trong cột asin và node đích là sản phẩm nằm trong cột also_bought. Bởi vì ASINs/Product IDs của chúng tôi là kiểu chuỗi nên không thể nhập chúng vào mô hình, vì vậy tiếp theo chúng tôi sẽ ánh xạ tất cả các ASINs trong bộ dữ liệu thành một số nguyên duy nhất.

Tạo đặc trưng nút: Chúng tôi thực hiện việc biểu diễn các chuỗi văn bản từ 2 cột 'description' và 'title' thành các biểu diễn số hóa dựa trên TF-IDF, sau đó kết hợp các biểu diễn này thành một ma trận kết hợp để sử dụng cho việc huấn luyện mô hình.

Tạo node label: Tiếp theo chúng tôi thực hiện trích xuất node label là danh mục sản phẩm (niche categoty).

Sau khi trích xuất xong node label, node feature và danh sách các cạnh trong đồ thị, chúng tôi bắt đầu xây dựng đồ thị một đồ thị đồng nhất gồm các sản phẩm làm nút và đưa đồ thị và các đặc trưng của đồ thị này vào mô hình. Đồ thị của chúng tôi bao gồm 3863 node và 24998 cạnh.

Chúng tôi chia các cạnh thành hai loại, positive edge và negative edge. Positive edge mô tả mối quan hệ thực sự giữa hai sản phẩm được mua cùng nhau, và các negative edge thì ngược lại, mô tả một mối quan hệ hoàn toàn không có giữa hai nút.

4. Các thành phần quan trọng của quá trình huấn luyện mô hình.

4.1. Tập huấn luyện và tập kiểm thử.

Chúng tôi sẽ dựa vào các cạnh trong đồ thị để chia ra thành hai tập huấn luyện và thử nghiệm. Chúng tôi thực hiện phân chia theo tỉ lệ 90% các cạnh positive nằm trong tập huấn luyện và 10% các cạnh positive nằm trong tập thử nghiệm. Đối với các cạnh negative, chúng tôi thực hiện lấy ngẫu nhiên ra số cạnh negative bằng đúng với số cạnh hiện có của đồ thị và cũng thực hiện chia 90% cạnh negative trong tập huấn luyện và 90% cạnh negative trong tập thử nghiệm.

Trong các mô hình GNN, chúng ta thường tính toán biểu diễn cho từng nút trong đồ thị. Tuy nhiên, trong bài toán dự đoán liên kết, chúng ta cần tính toán biểu diễn cho các cặp nút, không chỉ riêng lẻ từng nút. Để giải quyết vấn đề này, thư viện mà chúng tôi sử dụng - DGL (Deep Graph Library), đề xuất xem các cặp nút như là một đồ thị mới. Bằng cách này, chúng ta có thể mô tả một cặp nút bằng một cạnh trong đồ thị mới này. Trong dự đoán liên kết, có hai đồ thị: một đồ thị tích cực (positive graph) chứa tất cả các ví dụ tích cực (những cặp nút có mối quan hệ liên kết), và một đồ thị tiêu cực (negative graph) chứa tất cả các ví dụ tiêu cực (những cặp nút không có mối quan hệ). Cả hai đồ thị này sẽ có cùng tập hợp các nút như đồ thị ban đầu, điều này giúp việc truyền các đặc trưng của nút giữa các đồ thị trở nên dễ dàng hơn. Sau này, chúng ta có thể trực tiếp sử dụng các node representations được tính toán trên toàn bộ đồ thị để tính toán điểm số cho các cặp nút trong cả positive graph và negative graph. Lợi ích khi xử lý các cặp nút như một đồ thị là chúng tôi có thể sử dụng phương thức DGLGraph.apply_edges, thuận tiện để tính toán các đặc trưng cạnh mới dựa trên các đặc trưng của các nút liên quan và các đặc trưng cạnh ban đầu (nếu có).

Chúng tôi đã xây dựng một hàm `train_test_split` để thực hiện tất cả các công việc kể trên. Sau cùng, chúng tôi sẽ làm việc trên các đồ thị sau:

- Training graph: Là đồ thị dùng để học các embedding thông qua mô hình, đã bị loại bỏ các cạnh có trong đồ thị testing.
- Training positive và Training negative graph: Lần lượt là các đồ thị chứa các cạnh positive và negative của tập training, dung trong việc dự đoán xác suất xuất hiện các cạnh và tối ưu hóa hàm loss.
- Testing positive và Testing negative graph: Lần lượt đồ thị lần lượt chứa các cạnh positive và negative của tập testing, dùng để kiểm thử mô hình
- Tất cả các đồ thị này sẽ có cùng tập hợp các nút như đồ thị ban đầu, chỉ khác biệt ở tập hợp cạnh.

4.2. Tính toán khả năng xuất hiện của một cạnh

Chúng tôi xây dựng một hàm để tính toán các đặc trưng cạnh mới cho đồ thị g dựa trên thông tin biểu diễn của các nút. Cụ thể, nó thực hiện việc tính tích vô hướng của các biểu diễn của các nút liên quan để tạo ra các đặc trưng cạnh mới cho từng cạnh trong đồ thị. Dựa vào kết quả của tích vô hướng này, chúng tôi có được một xác suất tồn tại của một cạnh và dùng nó trong quá trình huấn luyện mô hình sau này.

4.3. Hàm loss và và các chỉ số đánh giá mô hình

Hàm loss mà chúng tôi sử dụng là binary cross entropy loss và chỉ số đánh giá là AUC:

$$\mathcal{L} = \sum_{u\sim v \in D} \left(y_{u\sim v} log(\hat{y}_{u\sim v}) + (1 - y_{u\sim v}) log(1 - \hat{y}_{u\sim v}) \right)$$

Đồng thời, để đánh giá các đề xuất sản phẩm từ mô hình, chúng tôi đã sử dụng một độ đo được gọi là tỉ lệ trúng đích (hit rate) cho top-K sản phẩm được đề xuất với K = 20. Với một sản phẩm chưa được nhìn thấy, mô hình sẽ tạo ra các nút sản phẩm được embedding gần nhất trong top-K để đưa ra đề xuất. Nếu ít nhất một trong số các sản phẩm được đề xuất thật sự có kết nối với sản phẩm chưa được nhìn thấy này, chúng tôi sẽ coi đó là một trường hợp trúng đích. Chúng tôi tính toán tỉ lệ trúng đích (hit rate) tại K bằng cách lấy số lượng trường hợp trúng đích chia cho số lượng sản phẩm kiểm thử chưa được xem xét.

5. Định nghĩa và huấn luyện mô hình GCN và GraphSAGE

5.1. Định nghĩa mô hình GCN

Mô hình GCN của chúng tôi được xây dựng bằng Python sử dụng thư viện DGL (Deep Graph Library). Với cấu trúc 2 lớp, mô hình này thể hiện sự tích hợp của đồ thị trong quá trình học và trích xuất đặc trưng từ dữ liệu đồ thị. Lớp GCN được xác định bằng cách kế thừa từ lớp nn.Module, lớp này nhận đầu vào là số đặc trưng đầu vào in_feats và số đặc trưng ẩn h_feats. Phương thức forward thực hiện lan truyền thuận qua mạng. Đầu tiên, nó áp dụng lớp GCN thứ nhất vào đồ thị g với đặc trưng đầu vào in_feat. Kết quả sau đó được đưa qua hàm ReLU trước khi tiếp tục truyền qua lớp GCN thứ hai. Kết quả cuối cùng của mô hình là các đặc trưng được trích xuất từ lớp cuối cùng trong mạng GCN này.

5.2. Định nghĩa mô hình GraphSAGE

Mô hình này bao gồm hai lớp GraphSAGE trong đó mỗi lớp thực hiện các phép tính convolution trên dữ liệu đầu vào. Lớp đầu tiên nhận đầu vào là đồ thị (g) cùng với các đặc trưng ban đầu của các đỉnh (in_feat). Lớp này sử dụng phép tính SAGEConv để tạo ra các biểu diễn mới cho các đỉnh trong đồ thị. Kết quả sau khi đi qua lớp này được đưa qua hàm kích hoạt ReLU để tăng tính phi tuyến tính. Tiếp theo, h được đưa vào lớp thứ hai, cũng là một lớp SAGEConv, để tiếp tục thực hiện phép tính convolution trên biểu diễn đã được trích xuất từ lớp trước đó. Kết quả cuối cùng sau khi đi qua lớp này sẽ là biểu diễn cuối cùng của các đỉnh trong đồ thị. Mỗi lớp tổng hợp thông tin từ các nút hàng xóm bằng cách sử dụng hàm tổng hợp giá trị trung bình.

5.3. Quá trình huấn luyện mô hình

Cả hai mô hình GCN và GraphSAGE của chúng tôi đều hoạt động chỉ dựa trên đặc trưng của các nút (bao gồm tiêu đề, mô tả của sản phẩm) và các mối quan hệ thông qua các cạnh của các sản phẩm được mua cùng nhau.

Tại mỗi bước của mô hình, chúng tôi sẽ cho mô hình học và trích xuất các đặc trưng trên đồ thị huấn luyện (training graph), sau đó sử dụng các đặc trưng này để tính tích vô hướng của các nút nguồn và nút đích trên các đồ thị training positive và training negative để có được xác suất sự xuất hiện của các cạnh này thông qua hàm DotPrediction. Chúng tôi sẽ so sánh với kết quả thực tế và đưa ra giá trị hàm loss qua hàm mất mát binary cross entropy. Chúng tôi sẽ thực hiện lặp lại các bước này qua từng vòng lặp để tối ưu hóa giá trị của hàm mất mát.

Chúng tôi tiến hành training cả hai mô hình này với 100 epoch, kích thước đầu ra của đặc trưng là 32, trình tối ưu là Adam, learning rate là 0.002. Riêng mô hình GraphSAGE, chúng tôi sử dụng thêm L2 regularization là 0.0025 để tránh tình trạng overfitting trên mô hình này.

IV. KẾT QUẢ VÀ TỔNG KẾT

1. Kết quả thực nghiệm

Sau đây là bảng tổng hợp kết quả thực nghiêm thu được từ hai mô hình:

	Train loss	Test loss	Test AUC	Hit rate	Training time (s)
GCN	0.3427	0.3332	0.9339	0.6922	75.138
GraphSAGE	0.3507	0.3152	0.9504	0.7411	83.918

2. Nhận xét

Khi xét trên tiêu chí AUC thì kết quả của hai mô hình GCN và GraphSAGE đều khá tốt, với AUC trên tập test đều trên 90%. Tuy nhiên, nhìn chung thì GraphSAGE có kết quả tốt hơn so với GCN ở phần lớn các tiêu chí khác đó là test loss, AUC và hit rate. Cụ thể, AUC của GraphSAGE là 0.9504, cao hơn AUC của GCN là 0.9339. Tỷ lệ trúng đích hit rate của GraphSAGE là 0.7411, cao hơn hit rate của GCN là 0.6922. Giá trị hàm mất mát của GraphSAGE cũng bé hơn GCN kiểm tra mặc dù trên tập huấn luyện thì GCN là mô hình có giá trị lối thấp hơn. Thời gian training của mô hình GCN cũng nhanh hơn mô hình GraphSAGE, 75.138 giây so với 83.918 giây, tuy nhiên đây là lượng thời gian không đnags kể (cả 2 đều được huấn luyện qua 100 epochs).

Như vậy, thông qua bảng tổng kết trên thì ta có thể nhận xét rằng GraphSAGE là mô hình có hiệu quả tốt hơn GCN trong bài toán này, đặc biệt là dựa trên chỉ số hit rate cho thấy sự đề xuất hiệu quả hơn của GraphSAGE. Mặc dù đây không phải là kết quả tối ưu nhất, nhưng kết quả của chúng tôi đã cho thấy rằng việc đề xuất sản phẩm có thể được thực hiện được bằng cách sử dụng mạng neural đồ thị.

3. Tổng kết

Chúng tôi đã chỉ ra cách mạng nơ-ron tích chập đồ thị có thể được sử dụng trong ngữ cảnh hệ thống đề xuất, trong đó có thể sử dụng dữ liệu dạng đồ thị để tận dụng được mối quan hệ của các đối tượng bên cạnh các đặc trưng được xem xét bởi các phương pháp truyền thống. Bằng cách sử dụng bộ dữ liệu văn phòng phẩm từ Amazon, chúng tôi đã xây dựng một đồ thị sản phẩm đồng nhất với các cạnh giữa các sản phẩm mua cùng nhau và áp dụng 2 mạng nơ-ron

đồ thị khác nhau là GCN và biến thể nhỏ của nó GraphSAGE để đề xuất sản phẩm. Mô hình GraphSAGE đã vượt trội hơn một chút so với mô hình GCN, nhưng cả hai mô hình này đều có thể được cải thiện, vì chúng tôi chưa đạt được kết quả mà chúng tôi mong đợi.

Đối với dự án này, những hướng tối ưu trong tương lai có thể được chúng tôi xem xét là thu thập thêm các đặc trưng của người dùng hoặc sử dụng thêm các đặc trưng sản phẩm như giá cả, nhãn hiệu, danh mục sản phẩm. Ngoài ra, bên cạnh tập dữ liệu metadata mà chúng tôi sử dụng trong dự án này, thì kho dữ liệu sản phẩm của Amazon còn cung cấp các tập dữ về đánh giá của người dùng như văn bản đánh giá, điểm đánh giá của người dùng (rating) và đặc trưng hình ảnh của sản phẩm. Chúng tôi có thể sử dụng thêm những đặc trưng nói trên để cải thiện hiệu suất của mô hình. Một xem xét khác là xây dựng một đồ thị bao gồm nhiều quan hệ hơn thay vì chỉ bao gồm một quan hệ sản phẩm được mua cùng nhau như chúng tôi đã làm, các quan hệ này có thể là sản phẩm được xem cùng nhau (also view), được mua sau khi xem (buy after viewing), hoặc một đồ thị hợp nhất sẽ sử dụng cả các cạnh item-item và cạnh user-item và sử dụng một thuật toán hiện đại và tiên tiến hơn.

Kho lưu trữ các báo cáo và mã nguồn của dự án:
https://drive.google.com/drive/folders/1RfvaDkYad7wuvJ2qAQq75bRZJf5JT-tj?usp=drive_link

V. TÀI LIỆU THAM KHẢO

1. T. N. Kipf and M. Welling. 2017. Semi-supervised Classification with Graph Convolutional Networks.
2. W. L. Hamilton, R. Ying, and J. Leskovec. 2017. Inductive Representation Learning on Large Graphs.
3. Nathan Tsai and Abdullatif Jarkas. 2021. Graph-Based Product Recommendation (DSC180B Capstone Project)