Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів побудови рекомендаційних систем на основі графових нейронних мереж
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-2
Количева П.А.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Волощук О.Б.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Количевої Поліни Андріївни
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів побудови рекомендаційних систем на основі графових нейронних мереж

затверджена наказом університету від 24 березня 20 22 р. № 414Ст

2. Термін подання студентом роботи до екзаменаційної комісії 11 травня 20 22 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки та дослідження рекомендаційних систем, дані відомих наукових проектів щодо розробки та дослідження алгоритмів рекомендаційних систем, набір даних музичних плейлистів

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі та постановка задачі, графові нейронні мережі, LightGCN модель, програмна реалізація.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1 – Загальна процедура реалізації моделі GNN, Рисунок 2 – Репрезентативні структури графів у рекомендаційних системах, Рисунок 3 – Архітектура моделі LightGCN, Рисунок 4 – Початкові характеристики графу, Лістинг 1 – Створення K-core графу, Рисунок 5 – Характеристики K-core графу, Лістинг 2 – Поділ датасету, Рисунок 6 – Результати навчання оригінальної LightGCN моделі, Рисунок 7 –Графік втрат оригінальної LightGCN моделі, Рисунок 8 –Графік recall@k оригінальної LightGCN моделі, Рисунок 9 – Результати навчання зміненої LightGCN моделі, Рисунок 10 –Графік втрат зміненої LightGCN моделі, Рисунок 11 –Графік recall@k зміненої LightGCN моделі.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	28.03.2022	Виконано
2	Аналіз предметної області і постановка завдання	29.03.2022-02.04.2022	Виконано
3	Вибір та підготовка вхідних даних	03.04.2022-06.04.2022	Виконано
4	Вибір алгоритмів для реалізації	07.04.2022-08.04.2022	Виконано
5	Створення программного коду	09.04.2022-20.04.2022	Виконано
6	Тестування і відладка програмного коду	21.04.2022-22.04.2022	Виконано
7	Обробка і оформлення результатів	23.04.2022-24.04.2022	Виконано
8	Оформлення пояснювальної записки	01.04.2022-28.04.2022	Виконано
9	Надання пояснювальної записки на перевірку	29.04.2022	Виконано
10	Нормоконтроль	30.04.2022	Виконано
11	Захист перед ЕК	11.05.2022	Виконано

Дата видачі завдання 28 березня 20 22 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 75 с., 11 рис., 2 дод., 16 джерел.

ГРАФОВА НЕЙРОННА МЕРЕЖА, КОЛЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КОНТЕНТ-ОРІЄНТОВАНІ МЕТОДИ, РЕКОМЕНДАЦІЙНА СИСТЕМА.

Об'єкт дослідження – система рекомендаційного типу з використанням графової нейронної мережі.

Предмет дослідження – методи, які використовуються для побудови рекомендаційних систем, метрики оцінок точності рекомендацій.

Мета роботи – створення рекомендаційної системи з використанням графової нейронної мережі з власним варіантом реалізації update-методу.

Методи дослідження – теоретичний (збір та структурування теоретичного матеріалу), експериментальний (розробка рекомендаційною системи в хмарному сервісі Google Colab засобами мови Python), аналіз отриманих результатів.

В результаті виконання магістерської атестаційної роботи була розроблена рекомендаційна система на основі графової нейронної мережі, використовуючи LightGCN модель з власним варіантом реалізації update-методу, проведено аналіз результатів оцінювання точності систем і виявлено, що точність рекомендацій розробленої системи перевищує результати роботи оригінальної моделі.

ABSTRACT

Explanatory note: 75 p., 11 fig., 2 ann., 16 sources.

GRAPH NEURAL NETWORK , COLLABORATIVE FILTERING, CONTENT-BASED METHODS, RECOMMENDATION SYSTEM.

Object of research – a recommendation-type system based on a graph neural network.

Subject of research – methods used to build recommendation systems, metrics for assessing the accuracy of recommendations.

Purpose of work – creation of a recommendation system using a graph neural network with owm implementation of the update-method.

Research methods – theoretical (collection and structuring of theoretical material), experimental (development of a recommendation system in the cloud service Google Colab using the Python language), analysis of the obtained results.

As a result of master's thesis the recommendation system based on the graph neural network was developed using the LightGCN model with its own version of implementation of the update-method. The analysis of the results of the evaluation of the accuracy of the system was conducted and it was found that the accuracy of the recommendations of the developed system exceeds the results of the original model.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

РС – рекомендаційна система;

GCN – Graph convolutional network – графова згорткова мережа;

GNN – Graph neural network– графова нейронна мережа;

LightGCN – Light graph convolutional network – модель графової нейронної мережі.

ВСТУП

Тема рекомендаційних систем набула все більшого значення у дев'яностих роках, оскільки Інтернет став важливим середовищем для бізнесу та електронної комерції. Інтернет став надавати небачені можливості для персоналізації, які не були доступні в інших каналах обміну інформацією.

Як наслідок виникла досить проста ідея – використовувати думки мільйонів людей в Інтернеті, намагаючись допомогти нам усім знаходити більш корисний та цікавий контент. Пройшло більше п’ятнадцяти років з моменту першої розробки програмних систем, які сьогодні називаються «системами рекомендацій». З тих пір дослідники постійно розробляли нові підходи до впровадження систем рекомендацій, і сьогодні більшість із нас звикли підтримуватись рекомендаційними службами, такими, як на Amazon.com, Netflix, LinkedIn та ін.

Історично рекомендаційні системи привертали багато уваги, застосовуючи методи від штучного інтелекту до фільтрації інформації – тобто рекомендувати веб-сайти або сортувати та класифікувати новини. Сфери застосування систем рекомендацій виходять за рамки чистих методів інформатизації, і в даний час рекомендаційна технологія пропонує рішення в таких різних сферах, як фінансові продукти, нерухомість, електронні товари для споживачів, фільми, книги, музика, новини та веб-сайти та багато іншого.

Релевантні рекомендації скорочують час, необхідний для пошуку товарів і послуг, і значно збільшують вірогідність попадання в поле зору користувача інших об'єктів, які зможуть його зацікавити. В результаті підвищується лояльність і задоволеність користувачів веб-сервісами. Саме тому підвищення точності та актуальності результатів роботи рекомендаційних систем є актуальним завданням для дослідження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Основна ідея рекомендаційних систем

Збільшення значення Інтернету як засобу здійснення електронних і ділових операцій послужило рушійною силою розвитку технології рекомендаційних систем. Важливим каталізатором в цьому відношенні є та легкість, з якою Інтернет дозволяє користувачам надавати відгуки про свої уподобання. Наприклад, розглянемо сценарій використання контент-провайдера – Netflix. У ньому користувачі можуть легко надавати відгуки простим клацанням миші. Типова технологія надання відгуків полягає в формі виставлення оцінок, в якій користувачі вибирають числові значення зі спеціальної рейтингової системи (наприклад, системи оцінювання «5 зірок»), в якій вказують їх «лайки» і «дізлайки» для різного контенту.

Інші форми зворотного зв'язку не настільки явно виражені, але їх ще легше зібрати в рамках веб-орієнтованої концепції. Наприклад, простий факт придбання або перегляду товару користувачем може розглядатися як вираження схвалення для цього товару. Такі форми зворотного зв'язку зазвичай використовуються такими інтернет-магазинами, як Amazon.com, і збір таких даних абсолютно не представляє складності з точки зору дій, що вимагаються від покупця .

Основна ідея рекомендаційних систем полягає в тому, щоб за допомогою цих різних джерел даних визначати інтереси клієнтів. Сутність, якій дається рекомендація, є користувачем, а рекомендований продукт – товаром. Отже, аналіз рекомендацій часто ґрунтується на попередніх взаємодіях між користувачами і товарами, оскільки минулі інтереси і схильності часто є хорошими показниками майбутнього вибору. Примітним винятком є системи, засновані на знаннях, в яких рекомендації пропонуються на основі вимог, визначених користувачем,

а не на основі його минулої діяльності.

1.2 Мета рекомендаційних систем

Рекомендаційні системи, в основному, використовуються інтернет-магазинами для збільшення продажів. Рекомендуючи ретельно відібрані товари користувачам, системи привертають увагу користувачів до актуальних товарів. Це збільшує обсяг продажів і прибуток магазинів. Незважаючи на те, що основною метою рекомендаційної системи є збільшення доходів, це часто досягається менш очевидними способами, ніж може здатися на перший погляд. Для досягнення більш масштабної, орієнтованої на бізнес мети збільшення доходів, загальні цілі рекомендаційних систем полягають в наступному:

- актуальність: Найбільш очевидною практичною метою рекомендаційних систем є рекомендація товарів, які є актуальними для конкретного користувача. Користувачі з більшою ймовірністю будуть купувати цікаві для них товари. Незважаючи на те, що релевантність є основною операційною метою системи, вона не є вичерпною сама по собі;

- новизна: Рекомендаційні системи дійсно корисні, коли рекомендований товар це те, що користувач не бачив раніше. Наприклад, популярні фільми улюбленого жанру рідко будуть новинками для користувача. Повторювані рекомендації популярних товарів також можуть привести до скорочення продажів;

- інтуїція: Пов'язаним з цим поняттям є поняття «винахідливість», при якій рекомендовані товари є дещо несподіваними, і тому існує незначний фактор у вигляді вдалого виявлення, на відміну від очевидних рекомендацій. Інтуїція відрізняється від новизни тим, що рекомендації викликають у користувача здивування, а не просто є чимось, про що він раніше не знав. Часто буває так, що конкретний користувач може

користуватися предметами тільки певного типу, хоча може існувати прихований інтерес до предметів інших типів, про які користувач сам може не підозрювати. На відміну від новизни, в центрі уваги інтуїтивних методів знаходяться такі рекомендації;

- підвищення різноманітності рекомендацій: Рекомендації, як правило, передбачають перелік товарів, які перебувають у верхній частині списку. Коли всі ці рекомендовані товари дуже схожі, це збільшує ризик того, що користувачеві може не сподобатися жоден з цих товарів. З іншого боку, якщо список рекомендацій містить товари різних типів, існує велика ймовірність того, що користувачеві сподобається хоча б один з цих товарів. Різноманітність має перевагу, що полягає в тому, що повторні рекомендації схожих предметів не дадуть користувачеві нудьгувати.

Крім цих конкретних цілей, ряд «soft» цілей також досягається в процесі рекомендацій як з точки зору користувача, так і з точки зору продавця. З точки зору користувача, рекомендації можуть допомогти підвищити загальну задоволеність користування сайтом. Наприклад, користувач, який постійно отримує актуальні рекомендації від Amazon.com, буде більш задоволений своїм досвідом і з більшою ймовірністю знову скористається сайтом. Це може підвищити лояльність користувачів і ще більше збільшити продажі на сайті. З боку продавця рекомендації можуть дати уявлення про потреби користувачів і допомогти ще більше вдосконалити процес роботи з сайтом. Крім того, часто буває корисно надати користувачеві пояснення того, чому рекомендується той чи інший товар. Наприклад, у випадку з Netflix рекомендації надаються разом з раніше переглянутих фільмами. Деякі алгоритми рекомендацій краще підходять для надання пояснень, ніж інші.

Існує велика різноманітність видів продукції, рекомендованих такими системами. Деякі рекомендаційні системи, такі як Facebook, не

рекомендують продукцію безпосередньо. Швидше, вони можуть рекомендувати соціальні зв'язки, які побічно сприяють сайту, підвищуючи його зручність використання і якість реклами.

1.3 Загальна постановка задачі рекомендацій

Існують різні способи формулювання завдання рекомендації. Дві основні моделі наступні:

- прогнозована версія завдання: Перший підхід полягає в передбаченні значення оцінки для комбінації «користувач – товар». Передбачається, що є дані для навчання, які вказують на переваги користувача по окремих товарах. Для m користувачів і n товарів це відповідає неповної матриці m×n, в якій для навчання використовуються задані (або спостерігаються) оцінки. Пропущені (або не зазначені) оцінки прогнозуються за допомогою цієї навчальної моделі. Це завдання також називають завданням завершення навчання по матриці, так як у нас є неповна матриця оцінок, а інші оцінки передбачаються за допомогою алгоритму навчання;

- рейтинговий варіант завдання: На практиці немає необхідності передбачати оцінки для окремих товарів для того, щоб давати рекомендації користувачам. Замість цього, магазин може порекомендувати топ-к товарів для конкретного користувача, або визначити топ-к користувачів, для рекомендації певного товару. Визначення топ-к товарів більш поширене, ніж визначення топ-к користувачів, хоча методи в цих двох випадках абсолютно аналогічні. Цю проблему також називають проблемою рекомендації top-k, і вона є рейтинговим формулюванням проблеми рекомендації. У другому випадку не важливі значення прогнозованих оцінок. Перше формулювання є більш загальним, так як рішення другого випадку можуть бути отримані шляхом рішення першого формулювання для

різних комбінацій користувачів – товарів, а потім ранжирування рекомендацій. Однак у багатьох випадках простіше і природніше розробити методи безпосереднього вирішення рейтингової версії проблеми.

1.4 Принцип роботи рекомендаційних алгоритмів

Основний принцип рекомендацій полягає в тому, що існує значна залежність між користувачем і діяльністю, орієнтованою на товар. Наприклад, користувач, який цікавиться історичним документальним фільмом, швидше за все, зацікавиться іншим історичним документальним фільмом або освітньою програмою, ніж бойовиком. У багатьох випадках, різні категорії товарів можуть показувати значущі взаємозв'язки, які можуть бути використані для вироблення більш точних рекомендацій. В якості альтернативи, залежності можуть проявлятися в більш чіткій деталізації окремих товарів, а не в категоріях. Ці залежності можуть бути вивчені на основі даних з матриці оцінок, а результуюча модель використовується для складання прогнозів для цільових користувачів. Чим більша кількість оцінюваних товарів є у користувача, тим легше зробити достовірні прогнози про майбутню поведінку користувача. Для виконання цього завдання можна використовувати безліч різних моделей навчання. Наприклад, спільне замовлення або однакова оцінка однакових товарів від різних користувачів може бути використана для створення груп схожих користувачів, які зацікавлені в схожих товарах. Інтереси і дії цих груп можуть бути використані для складання рекомендацій окремим членам цих груп.

Описане вище засновано на дуже простому сімействі рекомендаційних алгоритмів, які називаються моделями сусідів. Це сімейство відноситься до більш широкого класу моделей, званих

коллаборативною фільтрацією. Термін «коллаборативна фільтрація» відноситься до використання оцінок від декількох користувачів для спільного прогнозування пропущених оцінок. На практиці рекомендаційні системи можуть бути більш складними і насиченими даними з великою різноманітністю додаткових типів даних. Наприклад, в рекомендаційних системах, заснованих на контенті, контент грає основну роль в процесі вироблення рекомендацій, в якому оцінки і атрибути опису товарів використовуються для складання прогнозів. Основна ідея полягає в тому, що інтереси користувачів можуть бути змодельовані на основі властивостей (або атрибутів) товарів, які вони оцінювали або переглядали в минулому. Різноманітні системи, засновані на знаннях, в яких користувачі інтерактивно визначають свої інтереси, а специфікація користувача поєднується зі знанням області для надання рекомендацій. У просунутих моделях можуть бути використані контекстні дані, такі як тимчасова інформація, зовнішні знання, інформація про місцезнаходження, соціальна інформація, або інформація про соціальні зв’язки [1].

1.5 Проблеми рекомендаційних алгоритмів

Рекомендаційна система стикається з багатьма проблемами, які необхідно досліджувати, і в цьому розділі описані деякими з них.

Проблема холодного старту: проблема холодного старту виникає в основному при появі нового користувача на сайті або при додаванні нового товару в систему. По-перше, як би ми порекомендували товар новому користувачеві, для якого ми не знаємо його інтересів, і він поки не оцінив жодного товару. По-друге, як ми можемо рекомендувати цей новий товар іншим, адже ніхто не оцінював цей товар ні позитивно, ні негативно, тому ймовірність того, що він буде цікавий користувачеві, невелика.

Проблема нестачі оцінок: Проблема нестачі оцінок – це важлива проблема в рекомендаційних системах, вона виникає, коли користувач має велику матрицю, що містить куплені товари, або переглянуті фільми, або музику. Проблема нестачі оцінок проявляється тоді, коли користувач не оцінює ці товари і при цьому рекомендаційна система залежить від того, як користувачі оцінюють товари в матриці щоб рекомендувати їх іншим користувачам.

Масштабованість: масштабованість визначає здатність системи ефективно працювати з високою продуктивністю при одночасному збільшенні обсягу інформації. Рекомендаційна система повинна рекомендувати товари для користувачів без змін, в той час як кількість користувачів збільшується або кількість товарів також збільшується. Для цього потрібно більше обчислень і витрат на них.

Проблема надмірної спеціалізованості: Рекомендації для користувачів засновані на вже відомих або визначаються тільки профілями користувачів, без виявлення нових товарів та інших доступних варіантів.

Різноманітність: переконайтеся, що результати рекомендацій охоплюють якомога більше простору товарів, а не всі з однієї групи.

Новизна: рекомендовані товари повинні містити нові.

Інноваційність: крім новизни, метою може бути і те, що деякі рекомендовані товари не тільки незнайомі, але і несподівані для користувача, про існування яких він раніше не замислювався.

Конфіденційність: конфіденційність є однією з важливих проблем в рекомендаційних системах. Користувачі повинні знати, яка інформація необхідна для того, щоб рекомендувати йому більш бажані товари, і як вона застосовується.

Шилінгові атаки: трапляються, якщо зловмисник або конкурент входить в систему і починає давати неправдиві оцінки деяких товарах або з метою підвищення популярності товару, або з метою зниження

його популярності.

Сірі вівці: сірі вівці зустрічаються в системах коллаборативної фільтрації, де думки користувача не прирівнюються до жодної групи і, отже, не можуть отримати користь від рекомендацій [2].

1.6 Відомі існуючі реалізації рекомендаційних систем

Багато систем, які ми використовуємо щодня, пропонують рекомендації своїм користувачам: наприклад, групи, вакансії і контакти можуть бути рекомендовані LinkedIn, Facebook рекомендує друзів, такі системи, як Last.fm, рекомендують музику, а такі сайти, як Forbes.com, рекомендують сюжети новин.

1.6.1 Amazon.com

Рекомендаційні системи широко використовуються інтернет-магазинами. На веб-сайтах, таких як Amazon.com (або інших аналогічних інтернет-магазинах), користувачам пропонуються рекомендації щодо продуктів, які вони, можливо, побажають купити. Алгоритми рекомендацій широко відомі завдяки їх використанню на сайтах інтернет-магазинів, де інтереси клієнтів використовуються в якості вхідних даних для складання списків рекомендованих товарів. У багатьох додатках використовуються тільки відомості про товари, які клієнти придбали і оцінили безпосередньо, з метою представлення цих інтересів, однак можуть бути включені і інші характеристики, такі як списки ознайомлених товарів, демографічні дані та улюблені артисти. Amazon.com використовує алгоритми рекомендацій для персоналізації інтернет-магазину для кожного клієнта. Магазин радикально змінюється в залежності від інтересів клієнта, показуючи програмні продукти, наприклад, інженерам-програмістам, а дитячі іграшки – молодим мамам.

Алгоритм, який використовується Amazon.com, базується на елементній коллаборативній фільтрації. Використовувані алгоритмом онлайн-розрахунки не залежать від кількості клієнтів і кількості товарів в каталозі. Алгоритм дає рекомендації в режимі реального часу і підходить для використання з великою кількістю даних.

Amazon.com використовує рекомендації в якості цільового інструменту маркетингу. Перейшовши за посиланням «Your Amazon», користувачі потрапляють в розділ, в якому вони можуть відфільтрувати рекомендації по типу та/або категорії товару, оцінити рекомендації по товарах, оцінити попередні покупки і зрозуміти, чому певні товари були рекомендовані їм. Крім того, список рекомендацій в кошику пропонує товари клієнтам на підставі його вмісту. Ця функція діє аналогічно імпульсивним покупкам на касі в супермаркеті, за винятком того, що у випадку з Amazon.com ці запропоновані «імпульсивні покупки» орієнтовані на індивідуальних клієнтів. Amazon.com використовує рекомендаційні алгоритми для персоналізації свого сайту, виходячи з конкретних інтересів кожного клієнта.

Замість того, щоб зіставляти профілі користувачів зі схожими користувачами, коллаборативна фільтрація по товарах зіставляє кожен товар, придбаний і/або оцінений користувачем зі схожими товарами, а потім об'єднує ці схожі товари в список рекомендацій. Щоб визначити, які збіги найбільш схожі на даний товар, алгоритм будує матрицю схожих товарів, щоб визначити товари, які клієнти часто купують разом. Отримавши матрицю схожих товарів, алгоритм виявляє товари, схожі на кожен товар, придбаний і/або оцінений користувачем, об'єднує ці товари, а потім рекомендує найбільш популярні або найтісніше пов'язані між собою товари. Ці обчислення дуже швидкі, так як вони залежать тільки від кількості товарів, придбаних і/або оцінених користувачем.

1.6.2 Netflix

Netflix – це онлайн сервіс прокату фільмів, який дозволяє користувачам орендувати фільми за щомісячну плату, ґрунтуючись на списку фільмів, які вони хочуть подивитися в заданому порядку. Фільми потім розсилаються користувачам або транслюються в потоковому режимі. Використовуючи опцію DVD, користувачі просто відправляють диск назад в компанію, а наступний фільм надсилається поштою без плати за перевезення.

Тривалість перебування підписників на сервісі прив'язана до кількості фільмів, які вони дивляться і отримують задоволення від перегляду. Якщо користувачі не можуть знайти фільми, які вони хочуть подивитися, вони, як правило, йдуть з сервісу. Тому, як для компанії, так і для підписників, дуже важлива наявність ефективної системи рекомендацій, щоб гарантувати, що клієнти знайдуть фільми, які їм сподобаються. Компанія заохочує підписників до оцінювання фільмів, які вони дивляться. В даний час Netflix володіє більш ніж 1,9 мільярдами оцінок, отриманих від більш ніж 11,7 мільйона підписників на більш ніж 85 000 найменувань фільмів з жовтня 1998 року. Щодня служба отримує більше 2 мільйонів оцінок.

Гібридна рекомендаційна система, яка використовується Netflix, іменована Cinematch, аналізує накопичені оцінки фільмів і використовує ці оцінки для складання сотень мільйонів персоналізованих прогнозів для користувачів на щоденній основі, з урахуванням особистих смаків. Система Cinematch автоматично аналізує накопичені оцінки фільмів по щотижневому графіку, використовуючи варіацію коефіцієнта кореляції Пірсона до всіх інших фільмів, щоб побудувати список «схожих» фільмів, які, ймовірно, сподобаються тим же користувачам. Так як користувачі надають оцінки, онлайн, в режимі реального часу, система використовує ці оцінки для обчислення багатовимірної регресії на

підставі цих значень кореляцій для того, щоб створити унікальний, персоналізований прогноз для кожного рекомендованого фільму. У разі відсутності персоналізованих рекомендацій використовується середня оцінка, привласнена фільмам. Ці прогнози відображаються на Веб-сайті з використанням червоних зірочок.

Для Netflix питання точності прогнозу результатів настільки важливе, що з 2006 року розробникам пропонуються певні призи, наприклад, виплата 1 мільйона доларів за перший алгоритм, який буде на 10% перевищувати по точності систему рекомендацій Netflix того часу. Приз був виграний в 2009 році дослідницької командою, відомої під назвою «BellKor's Pragmatic Chaos», через більш ніж 3 роки після початку конкурсу [3].

1.7 Постановка задачі дослідження

Як зазначалося раніше, точність рекомендацій відіграє важливу роль для задоволення потреб клієнтів, а отже і для рівня продажів онлайн-магазинів. За останній рік значної популярності набули рекомендаційні системи, які побудовані на основі графових нейронних мереж. Наразі існує вже безліч таких моделей. Серед них має сенс відзначити LightGCN модель – вона поєднує в собі високу точність та відносно низьку ресурсноємність.

LightGCN використовує агреговане представлення сусідів як нове представлення центрального вузла, тобто повністю відкидає початкову інформацію про вузол користувача/об'єкта, що може привести до втрати власних вподобань користувача або власних властивостей об'єкту. Для уникнення цієї проблеми в деяких дослідженнях об'єднують ці два представлення лінійно за допомогою операції підсумовування або середнього підсумовування.

Саме тому, враховуючи вищезгадане, пропонується вирішити

задачу надання рекомендацій взявши за основу LightGCN модель з власним варіантом реалізації update-методу для досягнення більшої точності рекомендацій шляхом збереження власних властивостей користувача/об'єкта.

Для досягнення поставленої мети необхідно розглянути наступні питання:

- провести аналіз існуючих методів створення РС;
- проаналізувати існуючі проблеми в даній предметній області і визначити шляхи їх вирішення;
- провести попередню обробку даних;
- створити РС на основі LightGCN моделі;
- створити РС на основі LightGCN моделі з власною реалізацією update-методу;
- провести оцінку та аналіз отриманих результатів;
- перевірити на практиці поведінку системи.

2 ГРАФОВІ НЕЙРОННІ МЕРЕЖІ

2.1 Основна ідея GNN

З стрімкою появою величезних обсягів графових даних, як-от соціальні мережі, молекулярні структури і графи знань, останніми роками виникла хвиля досліджень графових нейронних мереж (GNN). Виникнення GNN в основному пов'язане з розвитком згорткових нейронних мереж (CNN) та навчання уявленню графів (GRL). Коли CNN застосовується до звичайних евклідових даних, таких як зображення або тексти, вона надзвичайно ефективна у видаленні локальних ознак. Однак для неевклідових даних, таких як графи, CNN вимагає генералізації, щоб впоратися з ситуаціями, коли об'єкти обробки (наприклад, пікселі на зображеннях або вузли на графах) мають нефіксований розмір. Що стосується GRL, то його метою є генерація низькорозмірних векторів для вузлів графа, ребер або підграфів, які є складними структурами зв'язків графів. Наприклад, новаторська робота DeepWalk вивчає уявлення вузлів за допомогою SkipGram на згенерованому шляху зі випадковими блуканнями по графах. Комбінуючи CNN та GRL, були розроблені різні GNN для обробки структурної інформації та навчання високорівневих уявлень. Основні етапи проектування моделі GNN до виконання завдань на графах, показано на рисунку 2.1.

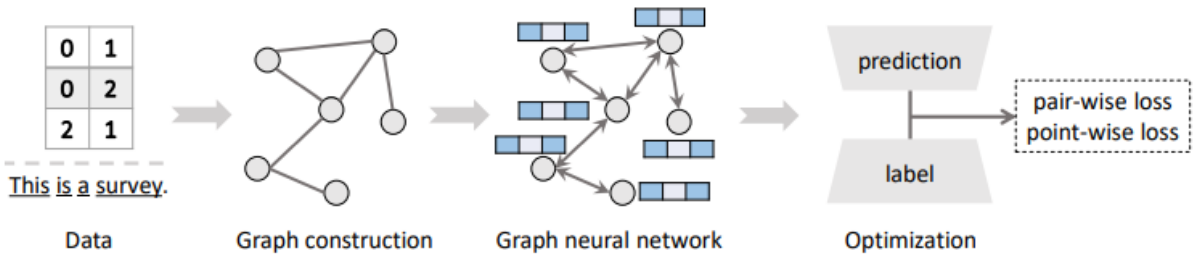


Рисунок 2.1 – Загальна процедура реалізації моделі GNN

2.1.1 Побудова графа

Спочатку використовуємо уніфіковану формулу визначення графа, тобто. G = (V , E), де V і E позначають безліч вузлів та ребер графа відповідно, а кожне ребро в E з'єднує будь-яке з будь-яким числом вузлів в V . В останні роки моделі на основі GNN фокусуються на розробці спеціалізованих мереж для наступних трьох категорій графів:

- однорідний граф, в якому кожне ребро з'єднує лише два вузли, і існує лише один тип вузлів та ребер;

- гетерогенний граф, в якому кожне ребро з'єднує лише два вузли, і існує кілька типів вузлів чи ребер;

- гіперграф, у якому кожне ребро з'єднує понад два вузли.

Нині у багатьох інформаційних системах реляційні дані природно представлені як графи. Наприклад, неявні відносини у соціальних мережах можна розглядати як єдиний граф, вузли якого представляють окремих людей, а ребра з'єднують людей, які підписані одне на одного. Однак, оскільки неструктуровані дані, такі як зображення та тексти, не містять графів у явному вигляді, для побудови графів необхідно визначати вузли та ребра вручну. Якщо взяти як приклад текстові дані, що використовуються в обробці природної мови (NLP), то слова/документи описуються як вузли, а ребра між ними будуються відповідно до частоти слів – зворотної частоти документа (IF-ITF). Крім того, напрямом досліджень, що розвиваються, у галузі навчання представлення на графах є граф знань (KG), який є типовим випадком гетерогенного графа. KG об'єднує безліч атрибутів даних та відносин, в якому вузли та ребра перевизначені як сутності та відносини, відповідно. Зокрема, сутності у KG можуть охоплювати широкий спектр об'єктів, включаючи людей, фільми, книги тощо. Відносини використовуються для опису того, як об'єкти пов'язані один з одним. Наприклад, фільм може бути пов'язаний із персонами (наприклад,

акторами чи режисерами), країнами, мовами тощо. Крім звичайних графів, останнім часом для гнучкої обробки складніших даних (наприклад, крім попарних відносин і множинних модальностей) досліджується гіперграф, у якому кожне ребро може поєднувати більше двох вузлів.

В цілому, побудова графів вимагає або вже наявних даних про графи, або абстрагування поняття вузлів і ребер графа з неструктурованих даних.

2.1.2 Проектування мережі

В цілому, моделі GNN можна поділити на спектральні та просторові. Спектральні моделі розглядають графи як сигнали та обробляють їх за допомогою згортки графів у спектральній області. Зокрема, сигнали графів спочатку перетворюються на спектральну область за допомогою перетворення Фур'є, визначеного на графах, потім застосовується фільтр, і, нарешті, оброблені сигнали перетворюються назад у просторову область. Формула обробки графового сигналу x за допомогою фільтра g має вигляд:

$$\mathbf{g} \star \mathbf{x} = \mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{g}) \odot \mathcal{F}(\mathbf{x})\right), \tag{2.1}$$

де F – перетворення Фур'є графа.

На противагу цьому просторові моделі здійснюють згортку безпосередньо на графових структурах для отримання локалізованих особливостей шляхом зваженого агрегування, як CNN. Незважаючи на те, що ці два типи моделей починають з різних місць, вони підкоряються тому самому принципу – ітеративно збирати інформацію про сусідів, щоб вловити кореляції високого порядку між вузлами та ребрами графа. Тут «інформація» представлена як ембедінги, тобто низькорозмірні

вектори. З цією метою основним і ключовим завданням GNN є поширення ембедингів за графами відповідно до структурних зв'язків, включаючи агрегування ембедингів сусідів та об'єднання їх з цільовим ембедингом (вузлом або ребром) для оновлення ембедингів графа шар за шаром.

2.1.3 Революційні моделі GNN

GCN – це типова спектральна модель, що об'єднує згортку графа та нейронні мережі для вирішення графової задачі напівконтрольованої класифікації. Зокрема, GCN апроксимує фільтр у згортці першим порядком, слідуючи [4]. Потім ембединги вузлів оновлюються таким чином:

$$\mathbf{H}^{l+1} = \delta(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l}\mathbf{W}^{l}), \tag{2.2}$$

де H^{l} – матриця ембедингів вузлів графа в l-му шарі згортки;

D – розмірність ембедингу;

$\tilde{A}$ – матриця суміжності графа із самозв'язком.

GraphSAGE – це інноваційна просторова модель GNN, яка вибирає сусідів цільового вузла, агрегує їх ембединги та поєднує з цільовим ембедингом для оновлення.

$$\begin{aligned} \mathbf{h}_{\mathcal{N}_i}^{l} &= \mathrm{AGGREGATE}_l\left(\{\mathbf{h}_j^l, \forall j \in \mathcal{N}_i\}\right), \\ \mathbf{h}_i^{l+1} &= \delta\left(\mathbf{W}^l\left[\mathbf{h}_i^l \| \mathbf{h}_{\mathcal{N}_i}^l\right]\right), \end{aligned} \tag{2.3}$$

де $\mathrm{N}i$ – обраних сусідів цільового вузла i.

Функція AGGREGATE має різні варіанти, такі як MEAN, LSTM тощо [5].

GAT – це просторова GNN–модель, яка вирішує кілька ключових проблем спектральних моделей, таких як низька здатність узагальнення від конкретної структури графа до іншого та складне обчислення зворотної матриці. GAT використовує механізми уваги для агрегування ознак сусідів (ембедингів) шляхом завдання різних ваг для різних вузлів. Зокрема, поширення формулюється таким чином:

$$\mathbf{h}_i^{l+1} = \delta\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}^l \mathbf{h}_j^l\right),$$
$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}^l \mathbf{h}_i^l \| \mathbf{W}^l \mathbf{h}_j^l\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}^l \mathbf{h}_i^l \| \mathbf{W}^l \mathbf{h}_k^l\right]\right)\right)}, \quad (2.4)$$

де αij – коефіцієнт поширення від вузла j до вузла i;

Ni – множина сусідів вузла i, включаючи сам вузол i.

Як показано в другому рівнянні, механізм уваги реалізується через повністю пов'язаний шар, параметризований навчальним вектором a, за яким слідує функція softmax.

HetGNN – це просторовий GNN, пристосований для гетерогенних графів. Враховуючи, що гетерогенний граф складається з декількох типів вузлів та ребер, HetGNN спочатку ділить сусідів на підмножини відповідно до їх типів. Після цього для кожного типу сусідів виконується агрегаторна функція для збирання локалізованої інформації, що поєднує операції LSTM та MEAN.

Крім того, різні типи інформації про сусідів агрегуються з урахуванням механізму уваги. Детальні формули опускаються, оскільки реалізація слідує вищезгаданим роботам.

HGNN – це спектральна модель, що реалізує GNN на гіперграфі. Згортка визначається наступним чином:

$$\mathbf{H}^{l+1} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{E}\mathbf{D}_e^{-1}\mathbf{E}^T\mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}^l\mathbf{W}^l, \quad (2.5)$$

де E_{ui} – чи містить гіпер-ребро u вузол i;

D_v – у скільки гіпер-ребер включений вузол;

D_e – скільки вузлів включає гіпер-ребро.

В цілому, цю операцію згортки можна розглядати як два етапи поширення сусідніх ембедингів: 1) поширення від вузлів до гіпер-ребра, що з'єднує їх, і 2) поширення від гіпер-ребра до вузла, з яким вони з'єднуються.

Для того, щоб додатково захопити структурну інформацію високого порядку на графі, згортка або розповсюдження ембедингів, згадані вище, виконуватимуться L раз. У більшості випадків $L \leq 4$, оскільки GNN страждають від проблеми надмірного згладжування, що оновлені ембединги матимуть невеликі коливання, коли кількість слоїв поширення стає більшою.

2.1.4 Оптимізація моделі

Після обробки розробленої мережі утворюються загальні ембединги вузлів або ребер, що кодують семантику ознаки, а також структури графа. Для виконання наступних завдань навчання графів ці ембединги будуть перетворені на цілі (наприклад, ймовірність того, що вузол належить до класу) за допомогою нейронних мереж загального типу (наприклад, MLP).

В основному існують завдання класифікації, передбачення та регресії на графах, що включають три рівні: вузол, ребро та підграф. Незважаючи на відмінність різних завдань, існує стандартна процедура оптимізації моделі. Зокрема, відповідні ембединги зіставляються і постачаються мітками для формулювання функції втрат, а потім для

навчання моделі використовуються існуючі оптимізатори. Після цього процесу існує кілька типів функцій відображення (наприклад, MLP, внутрішній добуток) та функції втрат (наприклад, парні, точкові), які можна вибрати для конкретних завдань. Для парної функції втрат заохочується поділ між позитивними та негативними вибірками, і типове формулювання BPR таке:

$$\mathcal{L} = \sum_{p,n} -\ln\sigma(s(p) - s(n)), \tag{2.6}$$

де $\sigma(\cdot)$ – сигмоїдна функція;

p – позитивні об'єкти;

n – негативні об'єкти;

$s(\cdot)$ – вимірювання об'єктів.

Для точкової функції втрат, вона включає середню квадратичну помилку (MSE) втрати, втрати крос-ентропії і так далі.

Для кращого розуміння ми розглянемо завдання прогнозування зв'язків і класифікації вузлів як приклади, щоб докладніше розповісти про те, як оптимізується модель GNN. Для передбачення зв'язків потрібно визначити ймовірність того, чи існує ребро між двома вузлами i, j. Технічно вона зазвичай розраховується на основі подібності до ембедингів вузлів у кожному шарі поширення:

$$s(i, j) = f(\{\mathbf{h}_i^l\}, \{\mathbf{h}_j^l\}), \tag{2.7}$$

де $f(\cdot)$ – функція маппінгу.

Більш того, ми можемо побудувати навчальні дані як O = $\{(i, j, k)\}$, що складаються з позитивних і випадково обраних негативних зразків, (i, j) і (i, k), відповідно. Зокрема, вузол i з'єднується із j на графі,

але не з k. У рекомендаційній системі вибірки будуть вказувати на те, що користувач i взаємодіяв з елементом j, але не взаємодіяв з елементом k, де k – вибірка з усіх інших елементів, з якими u не взаємодіяв раніше.

Тут і надалі, якщо вибраний BPR з попарними втратами, об'єктом оптимізації буде:

$$\mathcal{L} = \sum_{(i,j,k)\in O} -\ln\sigma(s(i,j) - s(i,k)) \quad . \tag{2.8}$$

З точки зору класифікації вузлів, ембединг вузла буде перетворений на розподіл ймовірності, яке представляє, до якого класу належить, показане нижче:

$$\mathbf{p}_i = f(\{\mathbf{h}_i^l\}), \tag{2.9}$$

де p_i – розподіл.

Зазвичай для задачі класифікації вибирається точкова функція втрат, така як втрата перехресної ентропії, яка формулюється як:

$$\mathcal{L} = -\sum_{(i,\mathbf{y}_i)\in O} \mathbf{y}_i^T \log \mathbf{p}_i. \quad . \tag{2.10}$$

В цілому, оптимізація в моделях на основі GNN розглядає представлення, узагальнені GNN як вхідні дані, а структури графа (наприклад, ребра, класи вузлів) як мітки, і функції втрат визначаються для навчання [6].

2.2 Переваги використання GNN в рекомендаційних задачах

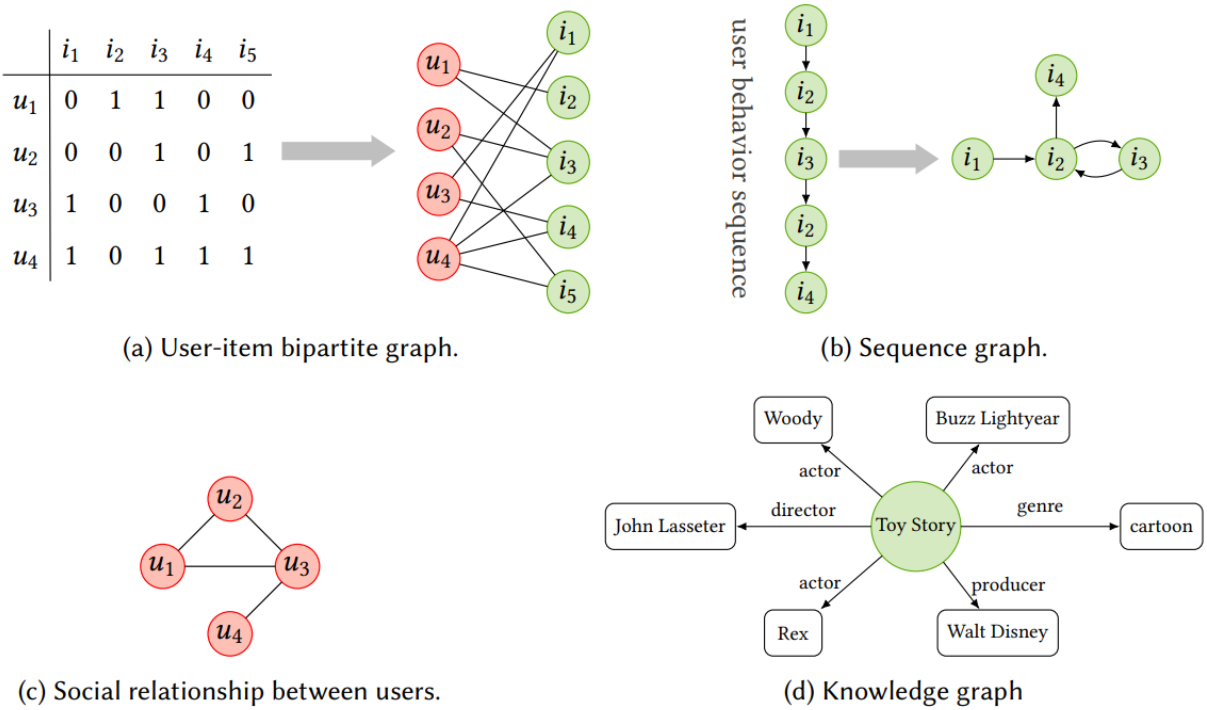
За останні кілька років було запропоновано безліч робіт із рекомендаційних систем на основі GNN. Перш ніж поринути у деталі останніх розробок, корисно зрозуміти мотиви застосування GNN у рекомендаційних системах.

Найбільш інтуїтивна причина полягає в тому, що методи GNN продемонстрували свою ефективність у навчанні представлень для графових даних у різних галузях, а більшість даних у рекомендаціях мають в основному графову структуру, як показано на рисунку 2.2.

Для загальної рекомендації дані про взаємодію можуть бути подані дводольними графом (як показано на рис. 2.2a) між вузлами користувача та об'єкта, де зв'язок являє собою взаємодію між відповідним користувачем та об'єктом.

Для послідовної рекомендації послідовність елементів може бути перетворена на послідовний граф, де кожен елемент може бути пов'язаний з одним або декількома наступними елементами.

На рисунку 2.2b показаний приклад послідовного графа, де між послідовними елементами є ребро. Порівняно з вихідними послідовними даними, граф послідовності забезпечує більшу гнучкість зв'язків між елементами. Крім того, деяка додаткова інформація також має природну графову структуру, наприклад, граф соціальних відносин та знань, як показано на рисунках 2.2c та 2.2d.



(a) User-item bipartite graph. (b) Sequence graph.

(c) Social relationship between users. (d) Knowledge graph

Рисунок 2.2 – Репрезентативні структури графів у рекомендаційних системах

Традиційно для загальних рекомендацій дослідники намагаються вивчити представлення користувача та об'єкту на основі парних взаємодій. Що стосується послідовних рекомендацій, дослідники використовують моделі послідовності для виявлення динамічних вподобань користувача на основі послідовної поведінки та передбачають, з чим він/вона, швидше за все, буде взаємодіяти наступного разу. У зв'язку з особливостями різних типів даних у рекомендаціях було запропоновано безліч моделей для ефективного вивчення їх шаблону для отримання кращих результатів рекомендацій, що є великою проблемою для розробки моделі. Розглядаючи інформацію у рекомендації з погляду графа, можна використовувати єдину структуру GNN на вирішення всіх цих задач. Завдання загальної рекомендації у вивченні ефективних представлень вузлів, тобто представлень користувача/об'єкта, та подальше прогнозування вподобань користувача. Завдання послідовної рекомендації у вивченні

інформативного представлення графа, тобто представлення послідовності. Як представлення вузлів, і представлення графа може бути вивчені за допомогою GNN. Крім того, воно зручніше і гнучкіше для включення додаткової інформації (якщо вона доступна), порівняно з неграфовою перспективою [7].

Крім цього, для загальних рекомендацій GNN може явно кодувати важливі спільні сигнали взаємодії користувача та предмета, щоб покращити представлення користувача та предмета через процес розповсюдження. Використання спільних сигналів для кращого навчання представлень не є абсолютно новою ідеєю. Наприклад, SVD++ включає представлення взаємодіючих елементів для збагачення представлень користувача. ItemRank будує граф елемент-елемент із взаємодій та використовує алгоритм випадкового проходу для ранжування елементів відповідно до уподобань користувача. Зверніть увагу, що SVD++ можна розглядати як використання однокрокових сусідів (тобто елементів) для покращення представлень користувача, тоді як ItemRank використовує двокрокових сусідів для покращення представлень елементів. У порівнянні з неграфовою моделлю, GNN є більш гнучкою та зручною для моделювання багатоланцюгових зв'язків при взаємодії користувача з предметом, а захоплені сигнали CF у високоланцюжкових сусідах продемонстрували свою ефективність для рекомендацій.

Для послідовних рекомендацій перетворення даних про послідовність у граф послідовності забезпечує більшу гнучкість вихідного переходу вибору елементів. Дублювання елементів у послідовній поведінці користувача – поширене явище, що може призвести до циклічної структури у графі послідовності. На рисунку 2.2b показаний приклад послідовності натиснутих елементів та відповідний граф послідовності. З погляду послідовності останній елемент $i4$ знаходиться під впливом попередніх чотирьох елементів. З

погляду графа послідовності, існує два шляхи до $i4$ через циклічну структуру між $i2$ і $i3$. Моделі послідовності суворо підпорядковуються тимчасовому порядку послідовності, у той час як GNN може відобразити складні вподобання користувача неявно в послідовній поведінці, завдяки циклічній структурі.

2.3 Загальні рекомендації на основі взаємодії користувачі-об'єкти

Загальні рекомендації спрямовані на моделювання уподобань користувача шляхом використання даних про взаємодію користувача та об'єкта, і таким чином надати список рекомендацій, що відображають довгострокові статичні інтереси кожного користувача.

Завдяки перевазі GNN у навчанні на графових даних, з'явилися зусилля зі створення рекомендаційних систем, що використовують архітектуру GNN для моделювання завдання рекомендації з погляду графа. Основна ідея полягає в тому, щоб використовувати елементи, з якими взаємодіяли користувачі, для покращення представлення користувачів та використовувати користувачів, які колись взаємодіяли з елементами для збагачення представлення елементів. Таким чином, багатошарові методи GNN дозволяють моделювати процес поширення інформації та більш ефективно використовувати зв'язки високого порядку від взаємодії користувача та елемента [8].

За наявності дводольного графа «користувач-елемент» основне завдання полягає в тому, як поширити інформацію про взаємодіючі елементи/користувачі на користувача/елемент та вивчити остаточні представлення користувача/елемента для прогнозування. Щоб використати всі переваги методів GNN на дводольному графі, необхідно вирішити чотири основні проблеми:

- побудова графа. Структура графа необхідна для визначення обсягу та типу інформації, що розповсюджується. Вихідний дводольний

граф складається з набору вузлів користувача/предмета та взаємодій між ними. Чи слід застосовувати GNN до неоднорідного дводольного графа або відновити однорідний граф на основі двокрокових сусідів? Враховуючи ефективність обчислень, як вибрати репрезентативних сусідів для розповсюдження графа замість оперувати повним графом?

- агрегація сусідів. Як агрегувати інформацію від сусідніх вузлів? Зокрема, чи слід розрізняти важливість сусідів, моделювати схожість між центральним вузлом та сусідами чи взаємодію між сусідами?

- оновлення інформації Як інтегрувати представлення центрального вузла та агреговане представлення його сусідів?

- кінцеве представлення вузла. Прогнозування вподобань користувача щодо елементів вимагає загального представлення користувача/елемента. Чи використовувати представлення вузла в останньому шарі або комбінацію представлень вузлів у всіх шарах як кінцеве представлення вузла?

2.3.1 Побудова графа

У більшості робіт GNN застосовують безпосередньо до вихідної структури дводольного графа «користувач-елемент». Існують дві проблеми з вихідним графом: одна – в ефективності, оскільки вихідна структура графа може бути недостатньою для навчання представлень користувача/предмета; інша – у непрактичності поширення інформації на графі з мільйонами і навіть мільярдами вузлів.

Для вирішення першої проблеми існуючі праці збагачують вихідну структуру графа шляхом додавання ребер чи віртуальних вузлів. Крім графа користувач-елемент, Multi-GCCF і DGCF додають ребра між двоходовими сусідами у вихідний граф, щоб отримати граф користувач-користувач і граф елемент-елемент. Таким чином, інформація про близькість між користувачами та елементами може бути явно включена

у взаємодію користувача та елемента. З огляду на те, що в попередніх роботах ігнорувалися наміри користувачів щодо вибору елементів, DGCF вводить віртуальні вузли намірів, які декомпозують вихідний граф на відповідний підграф для кожного наміру. З урахуванням обмежень незалежності різних намірів, неузгоджені представлення для різних намірів можуть бути вивчені шляхом ітеративного розповсюдження інформації за підграфом, що враховує наміри. Остаточне представлення – це інтеграція цих неузгоджених представлень, яка представляє вузол з різних сторін і має більш виразну силу.

Щодо другого питання пропонуються стратегії вибірки, щоб зробити GNN ефективною та масштабованою для великомасштабних завдань рекомендацій на основі графів. PinSage використовує метод вибірки на основі випадкових ходінь для отримання фіксованого розміру околиць з найбільшою кількістю відвідувань. Таким чином, ті вузли, які є безпосередніми сусідами центрального вузла, також можуть стати його сусідами. Для індуктивної рекомендації IG-MC використовує цільового користувача/предмет та їх одноходових сусідів як вузли для побудови підграфа. Конструкція підграфа, що охоплює, зменшує залежність від вихідної структури графа, що підвищує узагальненість моделі для перенесення на інший набір даних. Вибірка – це компроміс між інформацією про вихідний граф та обчислювальною ефективністю. PinSage включає більше випадковості, тоді як IG-MC жертвує більшою інформацією про граф. З точки зору перенесення, метод вибірки IG-MC краще, в іншому випадку, стратегія PinSage може бути кращою. Продуктивність моделі залежить від стратегії вибірки, і більш ефективна стратегія вибірки для побудови околиць заслуговує на подальше вивчення [9].

2.3.2 Агрегація сусідів

Крок агрегування має життєво важливе значення для поширення інформації для структури графа, який вирішує, скільки інформації про сусідів має бути поширеним. Середній пулінг – одна з найпростіших операцій агрегування, що розглядає сусідів однаково:

$$\mathbf{n}_u^{(l)} = \frac{1}{|\mathcal{N}_u|}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)} . \quad (2.11)$$

Середній пулінг простий у реалізації, але може бути неприйнятним, якщо значимість сусідів значно відрізняється. Після традиційної GCN у деяких роботах застосовується «нормалізація ступеня», яка надає ваги вузлам на основі структури графа:

$$\mathbf{n}_u^{(l)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)} . \quad (2.12)$$

Завдяки стратегії вибірки випадкових ходів PinSage приймає нормалізовану кількість відвідувань як важливість сусідів при агрегуванні векторних представлень сусідів. Однак ці функції агрегування визначають важливість сусідів у відповідності до структури графа, але ігнорують відносини між пов'язаними вузлами.

Враховуючи, що взаємодіючі елементи є однаково представницькими для відображення вподобань користувача, дослідники використовують механізм уваги диференціації важливості сусідів. Керуючись здоровим глуздом, що ембедінги елементів, відповідальних інтересам користувача, мають бути передані користувачеві більшою мірою (аналогічно і елементів), NGCF використовує елементний коефіцієнт доповнення характеристик

елементів, які цікавлять користувача, або вподобань користувачів щодо характеристик, якими володіє елемент. Якщо взяти як приклад вузол користувача, то представлення околиці обчислюється як:

$$\mathbf{n}_u^{(l)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \left(\mathbf{W}_1^{(l)} \mathbf{h}_i^{(l)} + \mathbf{W}_2^{(l)} \left(\mathbf{h}_i^{(l)} \odot \mathbf{h}_u^{(l)} \right) \right). \quad (2.13)$$

NIA-GCN стверджує, що існуючі функції агрегування не зберігають реляційну інформацію сусідів, тому пропонує підхід парного агрегування сусідів для явного відображення взаємодій між сусідами. Конкретно, він застосовує поелементне множення між кожними двома сусідами для моделювання відносин користувач-користувач/предмет-предмет.

Незважаючи на ефективність цих методів, вони не приділяють достатньої уваги випадку, коли існує кілька типів взаємодії, таких як перегляд та натискання. Щоб впоратися з багатотипними відносинами між користувачами та предметами, дослідники розробляють ієрархічну стратегію агрегування та спостерігають підвищення ефективності. Наприклад, MBGCN спочатку об'єднує взаємодіючі елементи, які стосуються кожного типу поведінки, відповідно, і далі об'єднує різні типи поведінки.

2.3.3 Оновлення інформації

За наявності агрегованого представлення сусідів та представлення центрального вузла, те, як оновити представлення центрального вузла, має значення для ітеративного поширення інформації. Деякі роботи використовують агреговане представлення сусідів як нове представлення центрального вузла, тобто повністю відкидають вихідну інформацію про користувача або об’єкт, що може упустити внутрішнє

вподобання користувача або внутрішню властивість товару. Інші беруть до уваги як центральний вузол, так і повідомлення його сусідів для оновлення представлень вузлів. У деяких дослідженнях ці два представлення об'єднуються лінійно за допомогою операції підсумовування або середнього підсумовування. Натхненні GraphSage деякі роботи використовують функцію конкатенації з нелінійним перетворенням для інтеграції цих двох представлень наступним чином:

$$\mathbf{h}_u^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot (\mathbf{h}_u^{(l)} \oplus \mathbf{n}_u^{(l)}) + \mathbf{b}^{(l)}\right), \tag{2.14}$$

де σ – функція активації, наприклад, ReLU, LeakyReLU та sigmoid.

У порівнянні з лінійною комбінацією, операція конкатенації з перетворенням ознак дозволяє більш складну взаємодію ознак. У деяких роботах зазначається, що нелінійна активація робить незначний внесок у загальну продуктивність, і вони спрощують операцію оновлення, видаляючи нелінійності, тим самим зберігаючи або навіть покращуючи продуктивність та підвищуючи ефективність обчислень.

2.3.4 Кінцеве представлення вузла

Застосування операцій агрегування та оновлення шару за шаром створює представлення вузлів для кожної глибини GNN. Загальні представлення користувачів та елементів необхідні для кінцевого завдання прогнозування. У деяких роботах як підсумкове представлення використовується вектор вузлів в останньому шарі GNN. Однак представлення, отримані в різних шарах, підкреслюють повідомлення, передані з різних зв'язків. Зокрема, представлення у нижньому шарі більше відображають індивідуальні ознаки, у той час як представлення у верхньому шарі більше відображають ознаки сусідів. Щоб скористатися

перевагами зв'язків, виражених на виході різних шарів, останні дослідження використовуються різні методи інтеграції представлень:

- конкатенація є однією з найпоширеніших стратегій;
- у деяких роботах використовується операція mean-pooling;
- деякі інтегрують представлення всіх шарів за допомогою операції sum-pooling;
- як mean-pooling, так и sum-pooling однаково розглядають представлення з різних шарів. Деякі поєднують представлення всіх шарів за допомогою зваженого об'єднання за шарами.

Зазначимо, що і mean-pooling, і sum-pooling можна розглядати як два окремі випадки зваженого об'єднання за шарами. Поширення інформації здійснюється шляхом накладення кількох шарів GNN, що допомагає вловити далекі залежності та розширити представлення вузлів за допомогою достатньої інформації про сусідів. Однак, надто велика кількість шарів призводить до проблеми надмірного згладжування. У результаті в деяких останніх роботах було запропоновано адаптивний баланс цього співвідношення [10].

2.4 Проблеми застосування GNNs у рекомендаційних системах

Незважаючи на хорошу мотивацію застосування графових нейронних мереж у рекомендаційних системах, існує чотири групи критичних проблем:

- як побудувати відповідні графи для конкретних завдань;
- як розробити механізм поширення та агрегування інформації;
- як оптимізувати модель;
- як забезпечити ефективність навчання та виведення моделі.

2.4.1 Побудова графа

Першим кроком застосування графових нейронних мереж є побудова графів. Це відбувається у два етапи: побудова вхідних даних у вигляді граф-структурованих даних; реорганізація мети рекомендації як завдання на графі. Якщо взяти як приклад завдання стандартної колаборативної фільтрації, то вхідними даними є дані про взаємодію користувача і елемента, а вихідними – передбачення відсутніх взаємодій користувача і елемента. Таким чином, можна побудувати дводольний граф з користувачами/елементами як вузли та взаємодій як ребер. Крім того, завдання CF перетворюється на прогноз зв'язків між користувачем і елементом на графі.

Однак побудувати графи, здатні добре впоратися із цим завданням, досить складно. Він має бути ретельно реалізований з урахуванням наступних аспектів:

- вузли. Однією з основних цілей навчання за допомогою графових нейронних мереж є надання вузлам представлень. Це призводить до того, що визначення вузлів значною мірою визначає масштаб моделей GNN, серед яких більшість параметрів займають ембедінги вузлів шару-0. Ембедінги ребер зазвичай або розглядаються, або обчислюються з урахуванням ембедингів вузлів. З іншого боку, визначення того, чи слід розрізняти різні типи вузлів, також є складним завданням. Наприклад, в завданні колаборативної фільтрації вузли користувачів і вузли елементів можуть моделюватися по-різному або розглядатися як один і той же тип вузлів. Ще одним складним моментом є обробка конкретних вхідних даних, таких як деякі числові характеристики, наприклад ціни товарів, які завжди є безперервними числами. Щоб представити ці характеристики в графі, одним з можливих рішень є їх дискретизація до категоріальних, які можуть бути представлені у вигляді вузлів;

- ребра. Визначення ребер сильно впливає на якість графа при

подальшому поширенні і агрегуванні, і навіть на оптимізацію моделі. У деяких тривіальних задачах вхідні дані рекомендаційної системи можна розглядати як різновид реляційних даних, таких як взаємодія користувач-елемент або соціальні відносини користувач-користувач. У деяких складних завданнях інші відносини можуть бути представлені у вигляді ребер. Наприклад, при рекомендації групи товарів, група складається з декількох товарів. Тоді ребро, що з'єднує групу та елемент, може відображувати відношення приналежності. Хороша конструкція ребер при побудові графа має повністю враховувати щільність графа. Занадто щільний граф означає наявність вузлів із дуже високими ступенями. Це призведе до того, що поширення ембедингів буде здійснюватися за великою кількістю сусідів. Це зробить ембдінги, що розповсюджуються, невиразними і безкорисними. Для обробки дуже густих ребер перспективними рішеннями є вибірка, фільтрація чи обрізка графів. Занадто розріджений граф також, звичайно, призведе до низької корисності розповсюдження ембедингів, оскільки поширення відбуватиметься лише на невеликій частині вузлів [11].

2.4.2 Проектування мережі

Відмінність GNN від традиційних методів навчання графів полягає у шарі поширення. Що ж до поширення, то вибір шляху має вирішальне значення для моделювання подібності високого порядку в рекомендаційних системах. Крім того, поширення може бути параметричним, при якому різним вузлам надаються різні ваги. Наприклад, при поширенні ембедингів елементів на вузол користувача у графі взаємодії користувача та елемента це відображає ефект CF на основі елемента. Ваги відносяться до різної важливості елементів, з якими раніше була взаємодія [12].

При поширенні також існують різні варіанти функцій агрегування,

включаючи mean pooling, LSTM, max, min та ін. Оскільки не існує єдиного варіанту, який може працювати найкращим чином серед усіх завдань рекомендації або різних наборів даних, дуже важливо розробити конкретний та правильний варіант. Крім того, різні варіанти розповсюдження/агрегації значно впливають на ефективність обчислень. Наприклад, mean pooling широко використовується в рекомендаційних моделях на основі GNN, оскільки він може бути обчислений ефективно, особливо для графа, що містить вузли високого ступеня, такі як популярні товари (які можуть пов'язувати велику кількість користувачів). Крім того, шари розповсюдження/агрегації можуть бути складені, щоб допомогти вузлам отримати доступ до сусідів із глибшими ланцюжками. Занадто дрібні шари не дозволяють добре змоделювати структуру графа високого порядку, а надто глибокі роблять ембедінги вузлів надто згладженими. Будь-який із цих двох випадків призведе до низької ефективності рекомендацій.

2.4.3 Оптимізація моделі

Для оптимізації моделей рекомендацій на основі графових нейронних мереж традиційні функції втрат у рекомендаційних системах завжди зводяться до втрат під час навчання графів. Наприклад, logloss при оптимізації можна розглядати як точкову втрату передбачення зв'язків. Аналогічно втрати BPR зазвичай використовуються в задачі передбачення зв'язків на графах. Іншим аспектом є вибірка даних. У рекомендаціях на основі GNN для вибірки позитивних або негативних елементів спосіб вибірки може залежати від структури графа. Наприклад, у соціальних рекомендаціях випадкове блукання графом може генерувати слабкі позитивні елементи (наприклад, елементи, із якими взаємодіяли друзі).

Крім того, іноді рекомендації на основі GNN можуть включати

кілька завдань, наприклад, завдання передбачення зв'язків по різних типах ребер. У такому разі виникає проблема, як збалансувати кожне завдання та зробити так, щоб вони посилювали одне одного.

2.4.4 Обчислювальна ефективність

У реальному світі рекомендаційні системи мають навчатися/інферуватися ефективно. Тому, щоб забезпечити прикладну цінність моделей рекомендацій на основі GNN, необхідно серйозно розглянути ефективність їх обчислень. У порівнянні з традиційними рекомендаційними методами без GNN, такими як NCF або FM, обчислювальні витрати моделей GNN набагато вищі. Особливо для спектральних моделей GNN, таких як GCN, у кожному шарі GCN виконуються складні матричні операції. При багатошаровому сумуванні шарів GCN вартість обчислень ще більше зростає. Тому просторові моделі GNN, такі як PinSage, легше реалізувати у великомасштабних промислових програмах. При використанні вибірки серед сусідів або обрізаної структури графа ефективність може бути збережена доти, доки можливо витримати падіння продуктивності рекомендацій [13].

2.5 Перспективні напрямки досліджень і відкриті питання

Хоча GNN досягла великих успіхів у рекомендаційних системах, у цьому розділі описується кілька перспективних напрямів досліджень.

2.5.1 GNN для гетерогенних графів у рекомендаціях

Гетерогенність – одна із основних характеристик графо-структурованих даних у рекомендаційних системах, тобто багато графів в рекомендаціях містять різні типи вузлів і зв'язків. Наприклад, граф

взаємодії складається з вузлів користувача та вузлів елемента, а граф знань містить багатотипні сутності та зв'язки. Через деякі унікальні характеристики (наприклад, об'єднання великої кількості інформації та багатої семантики) гетерогенних графів, пряме застосування методів для однорідних графів до них може призвести до неоптимальних представлень.

2.5.2 Різноманітне та невизначене представлення

Крім неоднорідності типів даних (наприклад, типів вузлів, таких як користувач і предмет, і типів ребер, таких як різні типи поведінки), користувачі у графі зазвичай мають різноманітні і невизначені інтереси. Подання кожного користувача як одиничного вектора (точки в низькорозмірному векторному просторі), як у попередніх роботах, не дозволяє вловити такі характеристики інтересів користувачів. Тому питання про те, як подати численні та невизначені інтереси користувачів, є напрямком, який варто вивчити.

2.5.3 Масштабованість GNN у рекомендаціях

У сценаріях промислових рекомендацій, де набори даних включають мільярди вузлів та ребер, а кожен вузол містить мільйони ознак, пряме застосування традиційної GNN ускладнене через великий обсяг пам'яті та тривалий час навчання. Для роботи з великомасштабними графами існує два основних напрямки: перше – зменшити розмір графа шляхом вибірки, щоб зробити існуючі GNN можливими до застосування; друге – розробити масштабований та ефективний GNN шляхом зміни архітектури моделі.

2.5.4 Динамічні графи у рекомендаціях

У реальних рекомендаційних системах з часом змінюються не тільки об'єкти, такі як користувачі і товари, а й відносини між ними. Щоб підтримувати актуальність рекомендацій, системи повинні ітеративно оновлюватися з урахуванням нової інформації, що надходить. З погляду графів, постійно оновлювана інформація призводить до появи динамічних графів замість статичних. Статичні графи стабільні, тому їх можна моделювати, у той час як динамічні графи є структурами, що змінюються. Цікавою проблемою для майбутніх досліджень є те, як розробити відповідну структуру GNN у відповідь на динамічні графи на практиці.

2.5.5 Графове змагальне навчання у рекомендаціях

Останні дослідження показують, що GNN легко обдурити невеликими спотвореннями на вході, тобто продуктивність GNN значно знижується, якщо структура графа містить шум. У реальних сценаріях рекомендацій часто буває так, що зв'язки між вузлами не завжди є надійними. Наприклад, користувачі можуть випадково натискати на елементи, і частина соціальних зв'язків не може бути зпіймана. Крім того, зловмисник може вводити до рекомендаційних систем підроблені дані.

2.5.6 Сфера охоплення GNN у рекомендаціях

Область охоплення вузла – це набір вузлів, включаючи сам вузол та його сусідів, досяжних у межах K-кроків, де K – число ітерацій поширення. При підсумовуванні кількох шарів GNN область охоплення вузлів з високим ступенем буде занадто велика і може внести шум, що

може призвести до проблеми надмірного згладжування та подальшого зниження продуктивності. Для вузлів з низьким ступенем, необхідна глибока архітектура GNN, щоб збільшити їхню область охоплення для отримання достатньої інформації про сусідів. Для даних графа рекомендацій ступінь вузлів має розподіл із довгим хвостом, тобто активні користувачі мають багато взаємодій з елементами, у той час як неактивні користувачі мають мало взаємодій, що аналогічно до популярних і не популярних елементів. Тому застосування однакової глибини поширення для всіх вузлів може бути неоптимальним.

3 LIGHTGCN МОДЕЛЬ

3.1 Загальна характеристика моделі

Графова конволюційна мережа (GCN) стала новою передовою розробкою для колаборативної фільтрації. Проте причини її ефективності для рекомендацій недостатньо добре вивчені. У існуючих роботах, що адаптують GCN для рекомендацій, немає ретельного аналізу процесу перетворення GCN, яка спочатку була розроблена для завдань класифікації графів і оснащена безліччю нейромережевих операцій. Однак досвідченим шляхом було виявлено, що дві найбільш поширені конструкції в GCN – перетворення ознак і нелінійна активація – роблять незначний внесок у продуктивність колаборативної фільтрації. Більше того, їхнє включення збільшує складність навчання та погіршує ефективність рекомендацій.

У цій моделі основною метою було спростити конструкцію GCN, щоб зробити її більш лаконічною та придатною для рекомендацій. LightGCN, включає лише найважливіший компонент GCN – агрегування сусідів – для колаборативної фільтрації. Зокрема, LightGCN навчається ембедингам користувача та елемента шляхом лінійного поширення їх за графом взаємодії користувача та елемента та використовує зважену суму ембедингів, отриманих на всіх рівнях, як кінцевий ембединг. Така проста, лінійна та акуратна модель набагато простіше в реалізації та навчанні, демонструє значні покращення порівняно з Neural Graph Collaborative Filtering (NGCF) – сучасною моделлю рекомендації на основі GCN – у точно таких же експериментальних умовах [14].

3.2 Архітектура LightGCN

Основна ідея GCN полягає у навчанні представлень для вузлів шляхом згладжування особливостей графа. І тому виконується ітераційна згортка графа, тобто агрегування характеристик сусідів як нове представлення цільового вузла.

Таке об'єднання сусідів можна представити у вигляді:

$$\mathbf{e}_u^{(k+1)} = \mathrm{AGG}(\mathbf{e}_u^{(k)}, \{\mathbf{e}_i^{(k)} : i \in \mathcal{N}_u\}). \quad (3.1)$$

AGG – це агрегаційна функція – ядро згортки графів, яка враховує представлення k-го шару про цільовий вузол і його сусідні вузли.

Багато робіт визначали AGG, як, наприклад, агрегатор зважених сум GIN, агрегатор LSTM у GraphSAGE, агрегатор білінійної взаємодії BGNN і т. д.

Однак більшість робіт пов'язують перетворення ознак або нелінійну активацію з функцією AGG. Хоча вони добре справляються із завданнями класифікації вузлів або графів, які мають семантичні вхідні ознаки, вони можуть бути тяжкими для колаборативної фільтрації.

На рисунку 3.1 зображена архітектура моделі LightGCN.

У LGC лише нормалізована сума ембедингів сусідів передається на наступний шар, інші операції, такі як самооб'єднання, перетворення ознак і нелінійна активація, видаляються, що значно полегшує GCN.

При об'єднанні шарів підсумовуються ембединги кожного шару щоб одержати остаточні представлення.

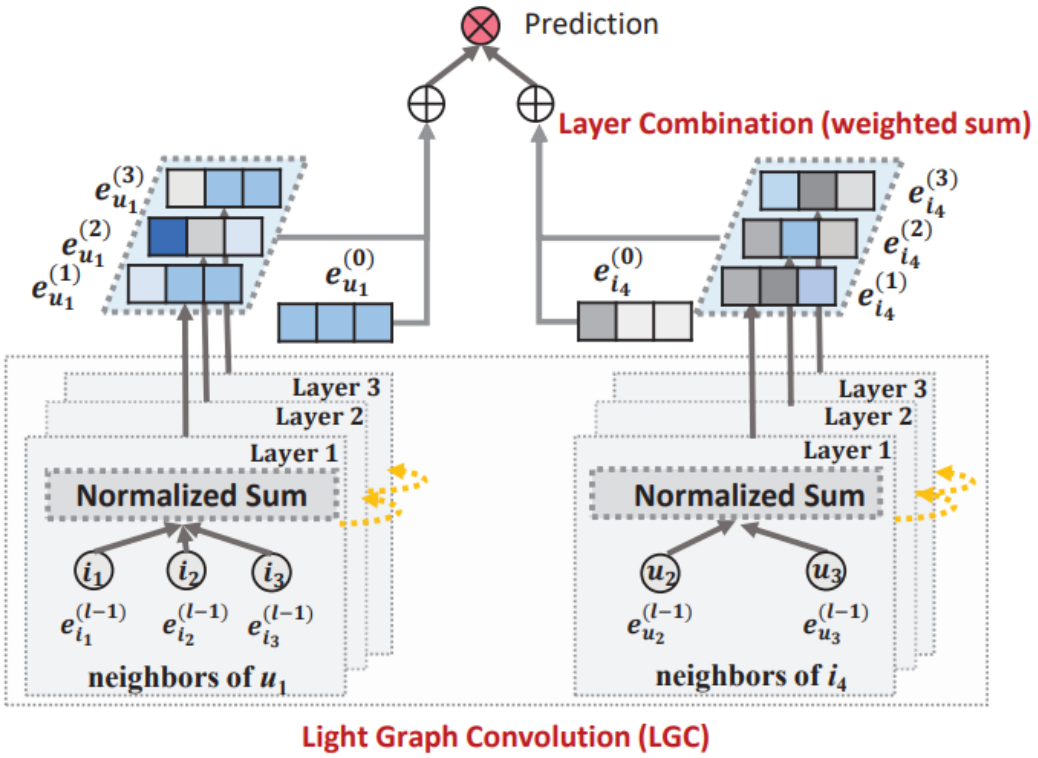


Рисунок 3.1 – Архітектура моделі LightGCN

3.2.1 Light Graph Convolution (LGC)

У LightGCN використовується простий агрегатор зважених сум, і відмовилися від використання перетворення ознак та нелінійної активації. Операція згортки графа (також відома зазвичай поширення) в LightGCN визначається як:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)} \quad . \tag{3.2}$$

Застосування терміну симетричної нормалізації відповідає дизайну стандартного GCN, що дозволяє уникнути збільшення розміру ембедингів під час операцій згортки графа; тут також можуть бути застосовані інші варіанти, наприклад норма L_1, але емпірично

встановлено, що симетрична нормалізація має високу продуктивність.

3.2.2 Об'єднання шарів та прогнозування моделі.

У LightGCN єдиними параметрами моделі, що навчаються, є ембедінги на 0-му шарі, тобто $e^{(0)}_u$ для всіх користувачів та $e^{(0)}_i$ для всіх елементів. Коли вони задані, ембедінги на більш високих шарах можуть бути обчислені за допомогою LGC, визначеного у формулах 3.2. Після K шарів LGC відбувається подальше об'єднання ембедингів , отриманих на кожному шарі, для формування остаточного представлення користувача (елемента):

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}, \tag{3.3}$$

де $\alpha_k \geq 0$ – важливість ембедингу k-го шару у формуванні остаточного ембедингу.

Його можна розглядати як гіперпараметр, який потрібно налаштовувати вручну, або параметр моделі (наприклад, виведення у мережі уваги), який потрібно оптимізувати автоматично. В експериментах виявлено, що установка α_k як 1/(K+1) призводить до високої продуктивності загалом. Тому, щоб уникнути зайвого ускладнення LightGCN та зберегти його простоту, не використовується спеціальний компонент для оптимізації α_k. Причини, за якими виконується об'єднання шарів щоб одержати остаточні представлення, три. Зі збільшенням кількості шарів, ембедінги будуть занадто згладжені. Тому просте використання останнього шару є проблематичним. Ембедінги на різних шарах відображують різну семантику. Наприклад, перший шар забезпечує гладкість для користувачів та предметів, що мають взаємодії, другий шар згладжує

користувачів (предмети), які перетинаються з взаємодіючими предметами (користувачами), а вищі шари фіксують близькість вищого порядку. Таким чином, їхнє об'єднання зробить представлення повнішим. Об'єднання ембедингів на різних шарах за допомогою зваженої суми передає ефект згортки графа із самозв'язками, що є важливим прийомом GCN.

Прогнозування моделі визначається як внутрішній добуток кінцевих представлень користувача та предмета:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i, \tag{3.4}$$

який використовується як ранжируюча оцінка для генерації рекомендацій.

3.3 Можливі зміни в архітектурі LightGCN

Ключова ідея GNNs полягає у тому, що вони використовують структуру графа для оновлення ембедингів вузлів на основі локального оточення навколо кожного вузла. Багаторазово передаючи ці ембединги через граф, можна почати враховувати інформацію як із локальної, так і більш глобальної околиці.

Як і інші моделі нейронних мереж, архітектура GNN складається з шарів, багато з яких можуть містити параметри, що навчаються. На кожному шарі ембединг вузла оновлюється з урахуванням ембедингів його сусідів (конкретний спосіб, яким це робиться, – головне, що відрізняє різні GNN один від одного). Таким чином, фінальний ембединг вузла визначається сусідньою структурою обчислювального графа.

Кожен шар GNN можна розкласти на три основні етапи:

- повідомлення: Для даного центрального вузла кожен сусід

передає центральному вузлу свій поточний ембединг (який спочатку може бути оброблений функцією повідомлення);

- агрегація: Повідомлення від сусідів агрегуються для отримання єдиного ембедингу, наприклад, шляхом підсумовування, усереднення чи взяття максимуму;

- оновлення: На етапі оновлення береться ембединг вузла і на попередньому рівні та поєднує його з агрегацією сусідів на поточному рівні, щоб отримати оновлений ембединг для вузла і на поточному рівні.

Ембединги вузлів після останнього шару K можуть бути використані як ознаки для прогнозування. У випадку з рекомендаційними системами необхідно передбачити існування ребра між двома вузлами. Для цього можна визначити схожість двох вузлів як скалярний добуток їхніх ембедингів:

$$sim(u, v) = x_u^K \cdot x_v^K . \tag{3.5}$$

Для LightGCN оригінальне правило оновлення виглядає як:

$$x_i^{(k+1)} = \sum_{j \in N(i)} \frac{1}{\sqrt{|N(i)|}\sqrt{|N(j)|}} x_j^{(k)} , \tag{3.6}$$

де N(i) – число сусідів вузла i;

N(j)– число сусідів вузла j;

$x_j^{(k)}$ – ембединг вузла, який є сусідом вузла x_i на поточному шарі [15].

З агрегованим представленням сусідів та представленням центрального вузла, як оновити представлення центрального вузла має важливе значення для процесу ітеративного поширення інформації. LightGCN використовує агреговане представлення сусідів як нове представлення центрального вузла, тобто повністю відкидає початкову

інформацію про вузол користувача/об'єкта, що може привести до втрати власних вподобань користувача або власних властивостей об'єкту.

Для уникнення цієї проблеми пропонується зважено об’єднувати ці два представлення, тоді правило оновлення матиме вигляд:

$$x_i^{(k+1)} = \alpha_1 x_i^{(k)} + \alpha_2 \sum_{j \in N(i)} \frac{1}{\sqrt{|N(i)|}\sqrt{|N(j)|}} x_j^{(k)}, \qquad (3.7)$$

де α_1 и α_2 – вагові коефіцієнти для цільового вузла на попередньому шарі та агрегованої суми сусідніх вузлів відповідно.

Саме тому, враховуючи вищезгадане, пропонується вирішити задачу надання рекомендацій взявши за основу LightGCN модель з новим варіантом реалізації update-методу для досягнення більшої точності рекомендацій шляхом збереження власних властивостей користувача/об'єкта.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Обґрунтування вибору мови програмування та середовища розробки

В якості мови програмування була обрана мова Python. Мова програмування Python і її бібліотеки є стандартом індустрії машинного навчання і аналізу великих даних. Вона містить прості і зручні інструменти для аналізу, візуалізації та обробки даних, такі як:

- sklearn – бібліотека, алгоритмів машинного навчання;
- matplotlib – бібліотека для візуалізації даних;
- pandas – бібліотека для обробки і аналізу даних;
- numpy – математична бібліотека з підтримкою багатовимірних масивів.

В якості середовища розробки було обрано Google Colab, це безкоштовний хмарний сервіс на основі Jupyter Notebook. Google Colab надає все необхідне для машинного навчання прямо в браузері, дає безкоштовний доступ до неймовірно швидких графічного процесору і тензорному процесору Google.

4.2 Опис використовуваних бібліотек

В данній роботі переважно використовується бібліотека PyTorch Geometric (PyG), яка є зручним інструментом для написання та навчання GNN.

PyG (PyTorch Geometric) – це бібліотека, створена на основі PyTorch для простого написання та навчання графових нейронних мереж (GNNs) для широкого спектра застосувань, пов'язаних із структурованими даними.

Вона включає різні методи глибокого навчання на графах та інших

нерегулярних структурах, також відомих як геометричне глибоке навчання, з безлічі опублікованих робіт. Крім того, до складу входять прості у використанні міні-пакетні завантажувачі для роботи з безліччю невеликих або одиночних гігантських графів, підтримка кількох GPU, підтримка DataPipe, розподілене навчання на графах через Quiver, велика кількість загальних еталонних наборів даних (на основі простих інструментів для створення власних), менеджер експериментів GraphGym та корисні перетворення, як для навчання на довільних графах, так і на 3D-сітках або хмарах точок.

4.3 Опис вхідних даних

Датасет містить 1 000 000 плейлистів, включаючи назви плейлистів та назви треків, створених користувачами на музичній платформі за кілька років.

Люди створюють плейлисти з різних причин: деякі плейлисти об'єднують музику за категоріями (наприклад, за жанром, виконавцем, роком або містом), за настроєм, темою або нагодою (наприклад, романтичні, сумні, святкові) або для певної мети (наприклад, зосередитися, потренуватися). Деякі плейлисти складаються навіть у тому, щоб знайти роботу мрії чи надіслати повідомлення комусь особливому.

Відібрана з більш ніж 4 мільярдів публічних плейлистів, ця база даних з 1 мільйона плейлистів включає понад 2 мільйони унікальних треків майже 300 000 виконавців і є найбільшою публічною базою даних музичних плейлистів у світі.

Кожен плейлист містить назву плейлиста, список треків (включаючи ID треків та метадані) та інші поля метаданих (час останнього редагування, кількість редагування плейлиста та інші). Всі дані анонімізовані для захисту конфіденційності користувачів. Вибірка

плейлистів проводиться з деякою рандомізацією, вручну фільтрується на предмет якості плейлиста та видалення образливого контенту, додані деякі фіктивні треки.

Кожен плейлист є словником, що містить наступні поля:

- pid – ціле число – id плейлиста – ID даного плейлиста. Це ціле число від 0 до 999999;

- name – рядок – назва списку відтворення;

- description – необов'язковий рядок – якщо є, опис, даний списку відтворення;

- modified_at – seconds – тимчасова мітка (у секундах від епохи), коли цей плейлист був оновлений востаннє;

- num_artists – загальна кількість унікальних виконавців для треків у плейлисті;

- num_albums – кількість унікальних альбомів для треків у плейлисті;

- num_tracks – кількість треків у плейлисті;

- num_followers – кількість передплатників цього списку відтворення на момент створення;

- num_edits – кількість окремих сесій редагування. Треки, додані у двогодинному вікні, вважаються доданими за один сеанс редагування;

- duration_ms – загальна тривалість всіх треків у плейлисті (у мілісекундах);

- collaborative – булеве значення – якщо true, то плейлист є спільним. Декілька користувачів можуть додавати треки в спільний плейлист;

- tracks – масив інформації про кожен трек у плейлисті. Кожен елемент масиву є словником;

- ім'я_треку – назва треку;

- track_uri – URI треку;

- album_name – назва альбому треку;

- album_uri – URI альбому;
- ім'я_артиста – ім'я основного виконавця треку;
- artist_uri– URI основного виконавця треку;
- duration_ms – тривалість треку в мілісекундах;
- pos – позиція треку у списку відтворення (на основі нуля).

4.4 Метрика оцінки точності рекомендацій

Основною метрикою, яка використовувалась в роботі, є recall@k.

Recall at k – це частка релевантних елементів, знайдених у топ-k рекомендаціях.

Наприклад, вирахувавши recall при 10, з'ясувалося, що він становить 40% у системі рекомендацій топ-10. Це означає, що 40% загальної кількості релевантних елементів з'являються в результатах top-k.

Математично recall@k для кожного плейлиста визначається наступним чином:

Recall@k = (кількість рекомендованих пісень @k, які є релевантними) / (загальна кількість релевантних пісень).

4.5 Опис програмної реалізації

4.5.1 Підготовка даних

Датасет містить 1 мільйон плейлистів, більше 2 мільйонів унікальних пісень і понад 66 мільйонів ребер, що пов'язують плейлисти з піснями (рисунок 4.1). Це дуже великий масив.

```
Original graph:
Num nodes: 367684 (30000 playlists, 337684 unique
songs)
Num edges: 1980923 (undirected)
```

Рисунок 4.1 – Початкові характеристики графу

Тому необхідно скоротити набір даних, щоб його можна було навчити у Colab за розумний час. Для цього розрахуємо «K-core» графа. K-core графа G – це найбільший можливий зв'язаний підграф графу G, в якому кожен вузол має ступінь не менше K. Це дасть найбільший підграф, в якому кожен плейлист містить не менше K пісень, і кожна пісня знаходиться не менше ніж в K плейлистах.

Це має дві переваги. По-перше, це зменшить розмір датасету. По-друге, це дозволить виключити рідкісні/непопулярні пісні та невеликі плейлисти. Плейлисти/пісні, що залишилися, будуть відносно інформаційно насиченими, що трохи спростить процес навчання, що ідеально підходить для цілей даної роботи.

Для реального обчислення K-core підграфа було використано бібліотеку SNAP із групи SNAP у Стенфорді під керівництвом професора Юрія Лесковця. Для обчислення K-core графа K=35, наприклад, слід використовувати наступний код (лістинг 4.1):

Лістинг 4.1 – Створення K-core графу

```
import snap
G = snap.TUNGraph().New()
...
K = 35
kcore = G.GetKCore(K)
```

З цього моменту буде використано PyTorch Geometric (PyG), тому граф SNAP буде перетворено на об'єкт PyG Data.

Отриманий набір даних містить 9296 плейлистів, 5696 унікальних

пісень і 659496 (ненаправлених) ребер між ними (1318992 направлених ребра) (рисунок 4.2).

```
K-core graph with K=35:
Num nodes: 14992 (9296 playlists, 5696 unique
songs)
Num edges: 659496 (undirected)
```

Рисунок 4.2 –Характеристики K-core графу

4.5.2 Поділ між training/validation/test наборами

Ключовим компонентом оцінювання будь-якого завдання ML є поділ на train/val/test. Це особливо важливо для задач Graph ML, і ще більш важливо для задач прогнозування ребер/зв'язків, таких як ця.

У графах точки даних сильно взаємопов'язані та залежні. Класифікація зображень, наприклад, набагато простіше; зображення незалежні, тому можемо просто зарахувати кожне зображення до одного з розбиття. З графами все інакше. Якщо вузол 1 і вузол 2 з'єднані ребром, то якщо призначимо вузол 1 в тестовий набір (і більше не будемо на ньому навчати), це вплине на наші прогнози для вузла 2, оскільки вузол 1 використовувався для передачі повідомлень вузлу 2.

Для завдань прогнозування ребер, подібних до цієї, розділяємо по ребрах, а не по вузлах. Основна ідея полягає в тому, щоб приховати деякі ребра під час навчання, передати граф, що залишився, як вхідні дані GNN, а потім прогнозувати по прихованих ребрах.

Таким чином, нам потрібні два типи ребер: ребра для передачі повідомлень, для виконання GNN поширення, та ребра для контролю/оцінки, для обчислення втрат та/або метрик оцінки.

Але також необхідно розділити ці дві групи на train/val/test. Це непросто, і в різних роботах це часто робиться по-різному. Однак

загальний метод полягає в тому, щоб розділити ребра на чотири групи: навчальні ребра повідомлень, навчальні ребра спостереження, ребра валідації та тестові ребра. Тоді:

- під час навчання використовуємо навчальні ребра повідомлень для прогнозування supervision ребер і обчислюємо втрати;

- при валідації використовуємо навчальні ребра повідомлень і supervision ребра для прогнозування валідаційних ребер;

- під час тестування використовуємо навчальні ребра повідомлень, навчальні supervision ребра та ребра валідації для передбачення тестових ребер.

Технічно, використовувалися ті ж навчальні ребра передачі повідомлень, що й навчальні supervision ребра, що також часто робиться.

PyG може виконати цей поділ за нас за допомогою RandomLinkSplit. В роботі використовувався поділ 70%-15%-15% (лістинг 4.2).

Лістинг 4.2 – Поділ датасету

```
    from torch_geometric.transforms import RandomLinkSplit
    transform = RandomLinkSplit(is_undirected=True,
add_negative_train_samples=False,
                                neg_sampling_ratio=0,
num_val=0.15, num_test=0.15)
    train_split, val_split, test_split = transform(data)
```

Встановивши is_undirected=True, гарантовано, що не буде витоку даних про зворотні ребра між сплітами (тобто ребра [2,4] і [4,2] повинні бути в одному спліті, оскільки це одне й те саме ребро, тільки у протилежних напрямках). Також встановлено add_negative_train_samples=False та neg_sampling_ratio=0, щоб відключити негативну вибірку для всіх трьох сплітів.

Частини для передачі повідомлень було збережено в об'єктах PyG Data, а розбиття спостереження/оцінки – в об'єктах PyG Dataset, що

дозволяє завантажувати їх партіями за допомогою DataLoader.

4.5.3 Реалізація LightGCN за допомогою PyTorch Geometric

PyG дозволяє легко визначати користувацькі GNN. Він навіть поставляється з деякими загальними шарами (як-от GCN або GraphSAGE), які можна використовувати готовими. Для LightGCN був написан власний шар.

Було реалізовано два класи Python. Спочатку визначено один шар поширення LightGCN. Цей клас виконуватиме крок поширення LightGCN, описаний раніше (формули 3.6-3.7).

Для цього було розширено базовий клас MessagePassing від PyG. Це надзвичайно корисний клас, який дбає про поширення повідомлень. Потрібно визначити, методи message(), aggregate() і update(), а потім просто викликавши метод propagate(), він виконає поширення відповідно до перших трьох методів.

Ключові методи в шарах PyG MessagePassing:

- message(): визначає, як надсилати повідомлення від сусідніх вузлів j центральному вузлу i. Це може бути просто передача ембедингів вузлів j як є;

- aggregate(): визначає метод агрегування, який використовується для поєднання повідомлень, які отримує кожен центральний вузол після передачі повідомлень;

- update(): визначає правило оновлення, яке зазвичай поєднує представлення центрального вузла на попередньому рівні з агрегованими даними його сусідів. Цей метод можна не використовувати, якщо замість цього інкапсулювати правило оновлення forward();

- propagate(): виконує поширення GNN, внутрішньо викликаючи message(), aggregation() та update(). Кожен із цих трьох методів отримує

аргументи, передані в propagate(). Виходом propagate() є матриця оновлених ембедингів вузлів після процесу розповсюдження;

- forward(): основна рушійна сила. За бажанням можливо попередньо обробити ембединги вузлів, потім запустити поширення GNN, викликавши propagate(), а потім, обробити оновлені ембединги, повернені propagate().

Функція агрегування – 'addition'. Вона проста, тому замість перевизначення функції aggregate() клас MessagePassing дозволяє вказати функцію агрегації в __init__().

Для message() для кожного сусіда j передане нами повідомлення – це ембединг x_j, розділений на нормуючий коефіцієнт. Кожен член у знаменнику коефіцієнта нормалізації – це квадратний корінь зі ступеня вузла i або j (це той самий коефіцієнт нормалізації, який використовується у статті GCN). Ці коефіцієнти нормалізації булі обчислені в forward(), потім передані в propagate(), щоб вони були доступні для використання в message().

Тобто, обчислюється коефіцієнти нормалізації в forward(), передаються у propagate(), вони використовуються для нормалізації кожного повідомлення x_j всередині message(), а потім propagate() внутрішньо обробляє агрегацію. Тепер, виконавши forward(), отримуємо оновлені ембединги після шару LightGCN.

Далі був створений основний клас GNN, який використовує шари LightGCN, які визначені раніше.

Його ключові моменти:

- у конструкторі ініціалізовано об'єкт nn.Embedding(), щоб зберігати отримані ембединги для плейлистів та пісень. Крім того, ініціалізовано список шарів LightGCN, які використовуються під час розповсюдження;

- gnn_propagation() виконує поширення по ребрах передачі повідомлень для обчислення багатомасштабних ембедингів. Для цього

спочатку зберігаються ембединги шару 0 у списку. Потім розповсюджується по кожному шару LightGCN і додаються оновлені ембединги до списку на кожному шарі. Наприкінці береться середнє значення ембедингів на кожному шарі (тобто тих, що були у списку), щоб отримати остаточні багатомасштабні ембединги;

- calc_loss() – це основна функція навчання. Спочатку вона викликає функцію gnn_propagation() для отримання багатомасштабних ембедингів. Потім вона викликає predict_scores(), щоб отримати оцінки для позитивних та негативних ребер. Нарешті, розраховує втрати для оцінок позитивних/негативних ребер, використовуючи втрати Байєсовського персоналізованого ранжирування (BPR), що є загальною функцією втрат для рекомендацій;

- evaluation() – це основна функція тестування/оцінки. Вона знаходить кращі рекомендації для кожного списку відтворення, а потім обчислює recall@k.

У наборі даних Colab є близько 5700 унікальних пісень. Тож для цієї роботи було встановлено k=300 для recall@k. Це здається розумним, тому що це означає, що хочемо, щоб модель передбачала правильні пісні в топ 300 із 5700 пісень, що становить приблизно топ 5%.

4.5.4 Навчання моделі

Після навчання протягом 30 епох, ось результуючі графіки втрат (рисунок 4.4) та валідації recall@k (рисунок 4.5). Модель явно вчиться давати кращі рекомендації; на самому початку recall@k становить близько 6%, що насправді є випадковим передбаченням. До 25 епохи вона становить близько 41% і продовжує зростати (рисунок 4.3).

Навчання протягом більшої кількості епох дає ще кращі результати, але оскільки Colab – це платний сервіс і для безкоштовного користування виділяється лише обмежена кількість ресурсів – було

вирішено зупинитися на такій кількості епох.

Це означає, що в середньому для даного плейлиста 41% релевантних пісень з тестового набору рекомендовано у топ 5% рекомендацій (тобто топ 300 пісень, як встановлено вище). Враховуючи, що рекомендація музики є складним завданням, це досить добрі результати.

```
Epoch 0: train loss=0.6929642933765601, val_recall=0.06648413716606932
Epoch 1: train loss=0.6928791577052674, val_recall=0.08865671004191876
Epoch 2: train loss=0.6927354994547881, val_recall=0.13190728791702275
Epoch 3: train loss=0.692490885200178, val_recall=0.19218143067817617
Epoch 4: train loss=0.6921099358278402, val_recall=0.25416708327755383
Epoch 5: train loss=0.6915452307687245, val_recall=0.3086568590743434
Epoch 6: train loss=0.6907561893817916, val_recall=0.3484754104024423
Epoch 7: train loss=0.6897018222946596, val_recall=0.37763548775922773
Epoch 8: train loss=0.6883381155178747, val_recall=0.39536103595245464
Epoch 9: train loss=0.6866307306947064, val_recall=0.4050363947507457
Epoch 10: train loss=0.6845833129190024, val_recall=0.4105384559863052
Epoch 11: train loss=0.6821252277843786
Epoch 12: train loss=0.679297481291519
Epoch 13: train loss=0.6760832832161908
Epoch 14: train loss=0.672495105748456
Epoch 15: train loss=0.6685025286451961, val_recall=0.4141791106348702
Epoch 16: train loss=0.6642657420186262
Epoch 17: train loss=0.6598207537757883
Epoch 18: train loss=0.654965238592672
Epoch 19: train loss=0.6500349663566224
Epoch 20: train loss=0.6450184308187676, val_recall=0.4139142950511053
Epoch 21: train loss=0.6399533302037312
Epoch 22: train loss=0.6348364130858097
Epoch 23: train loss=0.6298474141706808
Epoch 24: train loss=0.6249562034264826
Epoch 25: train loss=0.6201443292478932, val_recall=0.41573257483289455
Epoch 26: train loss=0.6156106125490435
Epoch 27: train loss=0.6113604429728509
Epoch 28: train loss=0.6071405867772816
Epoch 29: train loss=0.603204778163739

Best validation recall@k: 0.41573257483289455 at epoch 25
Test set recall@k: 0.4199230335217113
```

Рисунок 4.3 – Результати навчання оригінальної LightGCN моделі

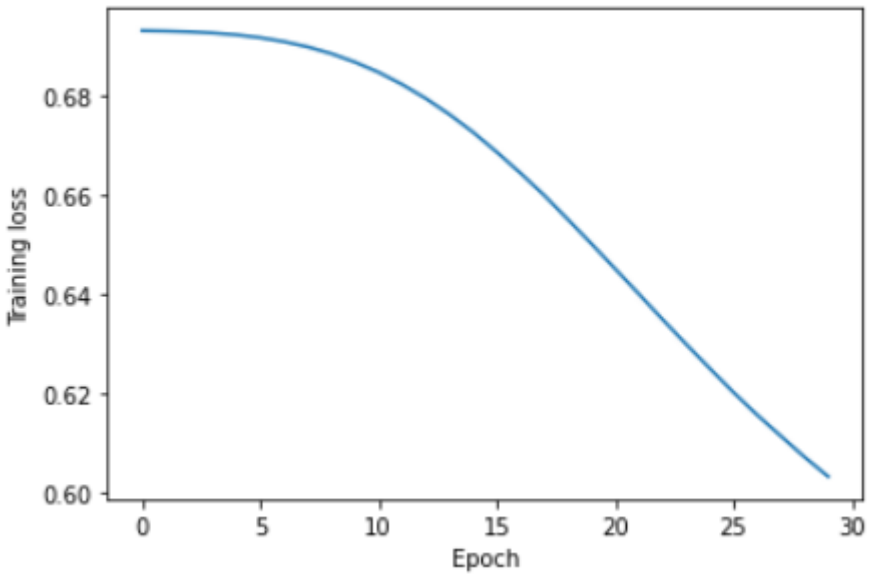


Рисунок 4.4 –Графік втрат оригінальної LightGCN моделі

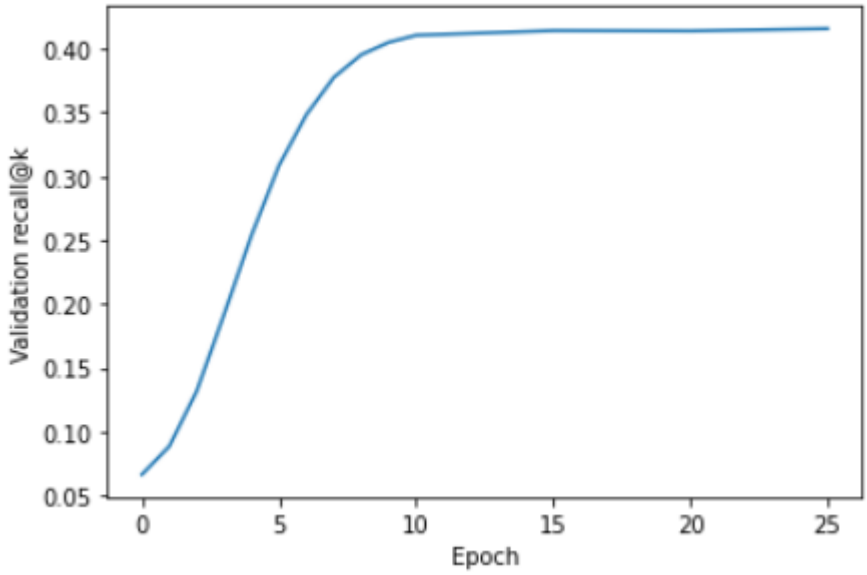


Рисунок 4.5 –Графік recall@k оригінальної LightGCN моделі

Після внесення змін у шар LightGCN, які були запропоновані у розділі 3.3 та навчання протягом 30 епох, результуючі графіки втрат (рисунок 4.7) та валідації recall@k (рисунок 4.8). Можна побачити, що запропонований варіант реалізації за однакову кількість епох демонструє кращі результати – 44% проти 41% (рисунок 4.6). При

подальшому навчанні передбачається що різниця між цими показниками при збільшенні кількості епох буде збільшуватись.

```
Epoch 0: train loss=0.6928500771887652, val_recall=0.05944625937716079
Epoch 1: train loss=0.6927594722715066, val_recall=0.06595194227481972
Epoch 2: train loss=0.6926429891426307, val_recall=0.07824897072781697
Epoch 3: train loss=0.6924619918532712, val_recall=0.10134319925953976
Epoch 4: train loss=0.6922012264080638, val_recall=0.1362206565374027
Epoch 5: train loss=0.6918187377229992, val_recall=0.18051696241756143
Epoch 6: train loss=0.6912816440705241, val_recall=0.22794611845580198
Epoch 7: train loss=0.6905517565652708, val_recall=0.2741573103132974
Epoch 8: train loss=0.689584548546028, val_recall=0.3157219990147159
Epoch 9: train loss=0.6883426852454818, val_recall=0.3478606433242127
Epoch 10: train loss=0.6868022656558124, val_recall=0.37530489658173266
Epoch 11: train loss=0.6848999000364018
Epoch 12: train loss=0.6826428945714994
Epoch 13: train loss=0.6800199458709653
Epoch 14: train loss=0.6770163964328713
Epoch 15: train loss=0.6735800733588451, val_recall=0.42859106780345546
Epoch 16: train loss=0.6698553716262962
Epoch 17: train loss=0.6658228206642991
Epoch 18: train loss=0.6613555010290025
Epoch 19: train loss=0.6566869737382554
Epoch 20: train loss=0.6518215379119849, val_recall=0.4351317705722233
Epoch 21: train loss=0.6467864983396047
Epoch 22: train loss=0.6415945532629487
Epoch 23: train loss=0.6364005097772517
Epoch 24: train loss=0.631211112157082
Epoch 25: train loss=0.6259926592146985, val_recall=0.43779477381211274
Epoch 26: train loss=0.6209654941642565
Epoch 27: train loss=0.6161709532926175
Epoch 28: train loss=0.6113547526849458
Epoch 29: train loss=0.606786795044293

Best validation recall@k: 0.43779477381211274 at epoch 25
Test set recall@k: 0.4431647777687755
```

Рисунок 4.6 – Результати навчання зміненої LightGCN моделі

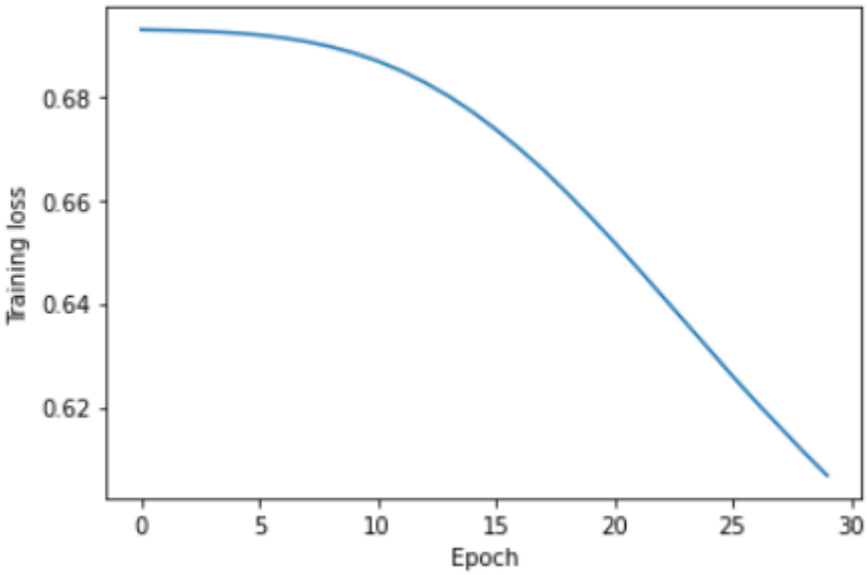


Рисунок 4.7 –Графік втрат зміненої LightGCN моделі

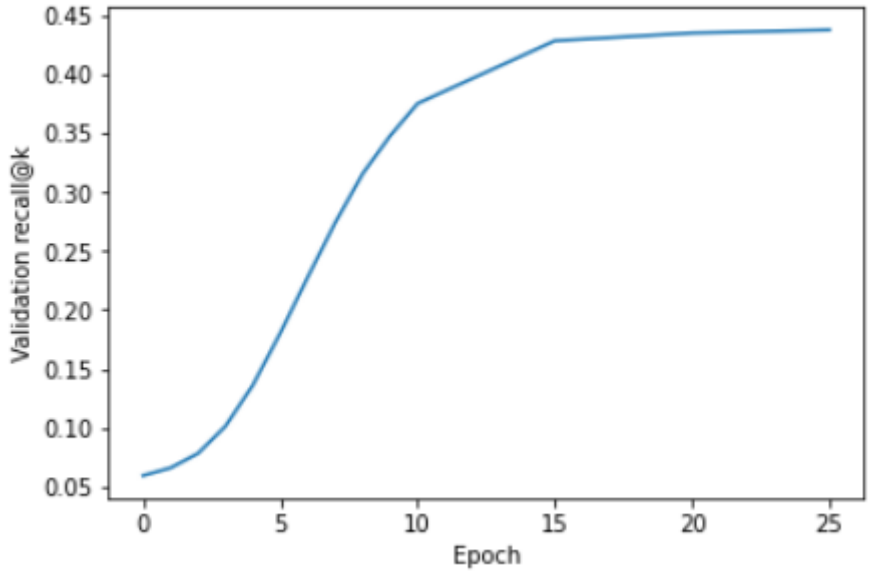


Рисунок 4.8 –Графік recall@k зміненої LightGCN моделі

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи була розроблена рекомендаційна система на основі графової нейронної мережі – оригінальна версія LightGCN моделі та LightGCN модель з власним варіантом реалізації update-методу, проведено аналіз результатів оцінювання точності систем і виявлено, що точність рекомендацій розробленої системи перевищує результати роботи оригінальної моделі (recall@k 44% проти 41%). При подальшому навчанні передбачається що різниця між цими показниками при збільшенні кількості епох буде збільшуватись.

РС створена у хмарному сервісі Google Colab засобами мови Python, на основі даних про плейлисти та пісні які входять до них.

Для досягнення поставленої мети було проведено аналіз існуючих методів створення РС, проаналізувано існуючі проблеми в даній предметній області і визначено шляхи їх вирішення, проведено попередню обробку даних, створено РС на основі LightGCN моделі та РС на основі LightGCN моделі з власною реалізацією update-методу, продено оцінку та аналіз отриманих результатів та перевірено на практиці поведінку системи.

Результати роботи відображені у тезах до Дванадцятої міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» [16].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aggarwal C. C. Recommender Systems. The Textbook. Springer; Springer International Publishing, 2016. 4 p.

2. Sharma L. A., Gera A. C. A Survey of Recommendation System: Research Challenges. International Journal of Engineering Trends and Technology (IJETT). 2013. Vol. 4, No. 9. P. 1989–1992.

3. Negre E. J. Information and Recommender Systems. London; ISTE Ltd, 2015. 44 p.

4. He X., Deng K. LightGCN: Simplifying and powering graph convolution network for recommendation. The 43rd International ACM SIGIR conference on research and development in Information Retrieval. 2020. P. 639–648.

5. Wang X., He X. Neural Graph Collaborative Filtering. The 42nd International ACM SIGIR conference on research and development in Information Retrieval. 2019. P.165–174.

6. Shumpei O., Yukihiro T., Shingo O., Akira T. Embedding-based news recommendation for millions of users. The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017. P.1933–1942.

7. Sun J., Zhang Y., Guo W., Guo H., Tang R., He H., Ma C., Coates M. Neighbor interaction aware graph convolution networks for recommendation. The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020. P.1289–1298.

8. Jannach D.I. Recommender systems: an introduction. New York; Cambridge University Press, 2011. 15 p.

9. Oord A., Dieleman S., Schrauwen B. Deep content-based music recommendation. Neural Information Processing Systems Conference. Neural Information Processing Systems Foundation (NIPS). 2013. Vol. 26. P.121-123.

10. Wang X., He X., Wang M., Feng F., Chua Y. Neural graph

collaborative filtering. The 42nd international ACM SIGIR conference on Research and development in Information Retrieval. 2019. P.165–174.

11. Wang X., Wang Y. Improving content-based and hybrid music recommendation using deep learning. The 22nd ACM international conference on Multimedia. 2014. P.627–636.

12. Adomavicius G., Tuzhilin A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 2017. Vol. 6. P. 734–749.

13. Kabbur S., Ning X., Karypis G. FISM: Factored Item Similarity Models for Top-N Recommender Systems. The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2013. P.659–667.

14. Wu F., Souza A., Zhang T., Fifty C., Yu T., Weinberger K. Simplifying graph convolutional networks.International conference on machine learning. PMLR. 2019. P.6861–6871.

15. Lakshmi S., Dr. Lakshmi T.Adi. Recommendation Systems: Issues and challenges. International Journal of Computer Science and Information Technologies. 2014. Vol. 5(4). P.5771–5772.

16. Количева П.А. Дослідження методів побудови рекомендаційних систем з використанням графових нейронних мереж. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: тез. докл. Дванадцятої міжнародної науково-технічної конференції. 2021. Том 2: секція 5. С. 21.