

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису:
УДК 303.732.4

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
« ____ » _____ 2025 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз і управління»
зі спеціальності 124 «Системний аналіз»
на тему: «Розробка рекомендаційної системи методами графових нейронних мереж»

Виконав:
студент 2 курсу, групи КА-41мп
Бичков Денис Віталійович

Науковий керівник:
Доцент, к.т.н., Недашківська Надія Іванівна

Консультант з нормоконтролю:
Доцент, к.ф.-м.н., Статкевич Віталій Михайлович

Рецензент:
Доцент кафедри штучного інтелекту
ННІПСА КПІ ім. Ігоря Сікорського,
к.ф.-м.н., доцент Пишнограєв Іван Олександрович.

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань.

Студент _____

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 124 «Системний аналіз»

Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» _____ 2025 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Бичкову Денису Віталійовичу

1. Тема дисертації «Розробка рекомендаційної системи методами нейронних мереж», керівник дисертації Недашківська Надія Іванівна, доцент, кандидат технічних наук, затверджені наказом по університету від «__» _____ 20__ р. № _____
2. Термін подання студентом роботи 14.12.2025.
3. Об'єкт дослідження: Механізм побудови рекомендаційної системи на основі графових нейронних мереж, який дозволяє моделювати взаємодії між користувачами та об'єктами рекомендацій у вигляді графової структури.
4. Алгоритми формування персоналізованих рекомендацій із використанням графових нейронних мереж (GNN), зокрема моделей GCN, GraphSAGE, LightGCN, а також методів обробки взаємодій у графових даних.

5. Перелік завдань, які потрібно розробити:

- провести огляд сучасних підходів до побудови рекомендаційних систем, зокрема графових;
- проаналізувати методи графових нейронних мереж, які застосовуються у задачах рекомендацій (GCN, GraphSAGE, NGCF, LightGCN тощо);
- обрати відповідний датасет для моделювання та провести його попередню обробку;
- побудувати прототип рекомендаційної системи на основі обраної графової архітектури;
- провести тестування ефективності розробленої системи з використанням стандартних метрик (Recall, NDCG тощо).

6. Перелік графічного (ілюстративного) матеріалу:

- 1) рисунки;
- 2) таблиці;
- 3) презентація.

7. Орієнтовний перелік публікацій:

Недашківська Н.І., Бичков Д.В «Розробка рекомендаційної системи на основі моделі GraphSAGE графових нейронних мереж».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Затвердження теми МД. Ознайомлення зі структурою МД згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І Сікорського»	01.09.2025-10.09.2025	виконано
2	Ознайомлення з ДСТУ 3008:2015.	11.09.2025-17.09.2025	виконано
3	Перший розділ. Огляд літературно інформаційних джерел. Аналіз предметної області.	18.09.2025-01.10.2025	виконано
4	Другий розділ. Розробка теоретичного узагальнення методу.	02.10.2025-15.10.2025	виконано
5	Третій розділ. Розробка програмного забезпечення	06.10.2025-29.10.2025	виконано
6	Третій розділ. Робота над практичним розділом магістерської дисертації	10.10.2024-12.11.2025	виконано
7	Четвертий розділ. Стартап-проект.	13.11.2025-19.11.2025	виконано

Студент

Денис БИЧКОВ

Керівник

Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Магістерська дисертація: 81 с., 13 рис., 20 табл., 2 дод., 14 джерел.

РЕКОМЕНДАЦІЙНА СИСТЕМА, ГРАФОВА НЕЙРОННА МЕРЕЖА, GRAPH-SAGE, GCN, LIGHTGCN, ПЕРСОНАЛІЗОВАНІ РЕКОМЕНДАЦІЇ, BPR.

Об'єкт дослідження — алгоритми побудови рекомендаційних систем із використанням графових нейронних мереж.

Предмет дослідження — методи формування персоналізованих рекомендацій на основі графових структур та нейронних моделей, таких як GraphSAGE, GCN, NGCF, LightGCN.

Мета роботи – розробити та дослідити ефективність рекомендаційної системи на основі графових нейронних мереж, що дозволяє покращити якість персоналізованих рекомендацій шляхом моделювання взаємодій у вигляді графа.

Методи дослідження — аналіз та обробка даних, побудова графів користувач–об'єкт, графові нейронні мережі, методи машинного навчання, метрики оцінювання якості рекомендацій (Recall, NDCG).

Основні результати – розроблено прототип рекомендаційної системи з використанням GNN-моделей; проведено експериментальне дослідження на основі датасету Last.fm; встановлено переваги LightGCN щодо якості формування рекомендацій у порівнянні з базовими моделями. Отримані результати підтверджують доцільність застосування графових нейронних мереж для покращення персоналізованих рекомендацій.

ABSTRACT

Master's Thesis: 81 pages, 13 figures, 20 tables, 2 appendices, 14 references.

RECOMMENDATION SYSTEM, GRAPH NEURAL NETWORKS, GRAPHSAGE, GCN, LIGHTGCN, PERSONALIZED RECOMMENDATIONS, BPR.

Object of the study: algorithms for building recommendation systems using graph neural networks.

Subject of the study: methods for generating personalized recommendations based on graph structures and neural models such as GraphSAGE, GCN, NGCF, and LightGCN.

Research objective: to develop and evaluate the effectiveness of a recommendation system based on graph neural networks that improves the quality of personalized recommendations by modeling user-item interactions as a graph.

Research methods: data analysis and preprocessing, construction of user-item graphs, application of graph neural networks, machine learning techniques, and evaluation metrics (Recall, NDCG).

Main results: a prototype of a recommendation system based on GNN models was developed; an experimental study was conducted using the Last.fm dataset; LightGCN demonstrated superior performance in generating recommendations compared to baseline models. The obtained results confirm the feasibility and efficiency of applying graph neural networks in the development of modern recommendation systems.

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНОГО СТАНУ ДОСЛІДЖЕНЬ.....	12
1.1 Рекомендаційні системи: загальна характеристика.....	12
1.2 Графові моделі у рекомендаційних системах.....	12
1.3 Графові нейронні мережі (GNN).....	13
1.4 Використання GNN у рекомендаційних системах.....	15
1.5 Висновки до розділу 1.....	16
РОЗДІЛ 2 ФОРМАЛІЗАЦІЯ ТА МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ПІДХОДУ НА ОСНОВІ GNN.....	17
2.1 Формалізація задачі рекомендацій на графі.....	17
2.2 Графова згорткова мережа (GCN).....	18
2.3 Індуктивний підхід GraphSAGE.....	20
2.4 Графові нейронні мережі для колаборативної фільтрації.....	21
2.5 Функція втрат і критерій оптимізації (BPR).....	23
2.6 Висновки до розділу 2.....	24
РОЗДІЛ 3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ.....	26
3.1. Аналіз джерел та сучасного стану проблеми.....	26
3.2. Вибір та опис датасету.....	29
3.3. Підготовка та обробка даних для GNN-моделі.....	30
3.4. Архітектура моделі та навчання з BPR-loss.....	33
3.5. Оцінка моделі: метрики якості та візуалізація ембедінгів.....	36
3.6. Візуалізація графової структури та ембедінгів.....	37
3.7 Висновки до розділу 3.....	39
РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ.....	41
4.1. Опис ідеї стартап-проекту.....	41
4.2. Технологічний аудит ідеї проекту.....	44

4.3. Аналіз ринкових можливостей запуску стартап-проекту.....	48
4.4. Розроблення ринкової стратегії стартап-проекту.....	57
4.5. Розроблення маркетингової програми стартап-проекту.....	59
4.7. Висновки до розділу 4.....	62
ВИСНОВКИ.....	63
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	64
ДОДАТОК А ЛІСТИНГ КОДУ.....	66
ДОДАТОК Б ПРЕЗЕНТАЦІЯ.....	74

ВСТУП

Сьогодні стримінгові музичні сервіси (Spotify, Apple Music, YouTube Music тощо) надають користувачам доступ до мільйонів треків, тому рекомендаційні системи відіграють критично важливу роль у допомозі слухачам знаходити улюблену музику. Персоналізовані рекомендації підвищують залученість і лояльність користувачів: наприклад, Spotify, маючи понад 500 мільйонів користувачів щомісяця, значною мірою завдячує своєму успіху потужним алгоритмам рекомендацій, що продовжують час прослуховування та формують віддану аудиторію. Однак традиційні підходи до рекомендацій мають обмеження.

Методи колаборативної фільтрації (CF) [1] на основі минулих уподобань користувачів є найбільш успішними, але страждають від проблеми розрідженості даних – вони погано рекомендують менш популярні або нові пісні з недостатньою історією прослуховувань.

Контент-орієнтовані методи (CBF) [2], що враховують метадані треків (жанр, виконавця, опис) або аудіо-фічери, можуть долати проблему холодного старту, проте зазвичай поступаються в точності – жанрова інформація занадто загальна, а аналіз сирого аудіо стикається з «семантичним розривом» між сигналом і музичними уподобаннями. Це зумовило пошук нових підходів, зокрема графових нейронних мереж (Graph Neural Networks, GNN), що дозволяють врахувати складні зв'язки у графі "користувач-музичний контент" і об'єднати різні джерела даних (поведінкові взаємодії, соціальні зв'язки, жанри тощо) в єдину модель рекомендацій.

Сучасні дослідження показують, що графові методи перевершують класичні алгоритми: зокрема, модель PinSAGE на основі GNN увійшла до найкращих за якістю рекомендацій і випередила традиційну матричну факторизацію, що підтверджує ефективність графового представлення взаємодій користувачів і треків. Таким чином, тема побудови

рекомендаційної системи методами GNN для музичного контенту є актуальною і має як науковий, так і практичний інтерес.

Метою роботи є розробка і дослідження рекомендаційної системи для музики на основі графових нейронних мереж. Для досягнення цієї мети були визначені такі завдання дослідження.

1. Проаналізувати сучасні наукові джерела і стан розвитку рекомендаційних систем у музичній сфері, виявити проблеми та переваги існуючих підходів.
2. Обрати відкритий датасет взаємодій користувачів з музичним контентом (наприклад, Last.fm або Spotify) та надати його опис з посиланням на джерело.
3. Здійснити збирання та попередню обробку даних: очистити й нормалізувати дані, сформувати граф взаємозв'язків користувачів і музичних треків (а також додаткових сутностей, як-от жанри).
4. Розробити архітектуру графової нейронної мережі (такої як GCN [], GAT, GraphSAGE) для рекомендацій; реалізувати та навчити модель на підготовлених даних.
5. Провести експериментальну оцінку якості рекомендацій: обрати метрики (Precision@K, Recall@K, NDCG тощо), порівняти результати GNN-моделі з класичними методами (колаборативна фільтрація, факторизація матриці уподобань), проаналізувати отримані результати та візуалізувати їх.
6. Реалізувати пошук схожих музичних треків на основі векторних представлень, отриманих GNN-моделлю (побудова функції пошуку top-N найбільш схожих пісень до заданої).
7. Виконати графову візуалізацію зв'язків у системі (користувачі–треки–жанри) для інтерпретації моделі та демонстрації кластерів.
8. Розробити концепцію стартап-проєкту на базі створеної системи: провести аналіз ринку та конкурентів, визначити цільову аудиторію, унікальні торговельні переваги продукту, спланувати технічну

реалізацію сервісу, виконати SWOT-аналіз, окреслити MVP, стратегію масштабування бізнесу та надати фінансове обґрунтування.

Наукова новизна роботи полягає у застосуванні графових нейронних мереж для рекомендацій у музичній медіабібліотеці. На відміну від традиційних алгоритмів, запропонований підхід дозволяє врахувати багатовимірні зв'язки: не лише взаємодії «користувач-трек», а й соціальні зв'язки між користувачами та семантичну близькість треків через жанри/теги. Поєднання таких різномірних даних у єдиному графі та навчання на ньому GNN-моделі дає змогу підвищити якість рекомендацій та вирішувати проблему «довгого хвоста» – рекомендувати менш популярні, але цікаві користувачу композиції.

Практичне значення роботи визначається можливістю впровадження розробленої системи в реальних сервісах. Поліпшення персоналізації музичних рекомендацій напряду впливає на задоволеність аудиторії і конкурентоспроможність платформи.

Результати роботи можуть бути використані стримінговими сервісами, онлайн-радіо чи музичними бібліотеками для підвищення утримання користувачів та часу прослуховування, а також слугувати основою для запуску стартап-проєкту, присвяченого музичним рекомендаціям.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНОГО СТАНУ ДОСЛІДЖЕНЬ

1.1 Рекомендаційні системи: загальна характеристика

Рекомендаційні системи (RS) — це програмні системи, що пропонують користувачам нові об'єкти (товари, послуги, контент) на основі аналізу їхніх уподобань та поведінки [3].

Найпоширенішим підходом є колаборативна фільтрація, коли система виявляє схожість між користувачами або елементами на основі матриці взаємодій «користувач–елемент». Класичним методом є матричний факторизаційний підхід, що представляє користувачів і елементи у спільному латентному просторі ознак.

Для навчання моделей з неявною (імпліцитною) зворотною відповіддю широко використовується метод Bayesian Personalized Ranking (BPR) [4]. Критерій BPR оптимізує модель на упорядкування: він спрямований на те, щоб для кожного користувача оцінки придбаних ним елементів були вищими, ніж оцінки непридбаних. Таким чином, традиційні алгоритми колаборативної фільтрації ефективно виявляють патерни уподобань, але можуть не враховувати складні багаторівневі зв'язки між користувачами та елементами.

1.2 Графові моделі у рекомендаційних системах

Взаємодії між користувачами і предметами (товарами, піснями, тощо) зручно представляти у вигляді графів. Зокрема, двочастковий (біпартитний) граф «користувач-елемент» містить дві множини вершин (одна відповідає користувачам, інша — рекомендованим об'єктам), а ребра позначають факт

взаємодії (наприклад, купівлі, перегляду або оцінки). Така графова інтерпретація дає можливість використовувати потужний апарат аналізу графів для рекомендацій. Крім того, до графової моделі RS можна інтегрувати додаткову інформацію: соціальні графи (зв'язки між користувачами), графи знань (семантичні мережі, що пов'язують предмети через спільні властивості) тощо.

Представлення даних у вигляді графів закладає основу для застосування графових алгоритмів і моделей глибокого навчання на графах. Зв'язність графа дозволяє виявляти вищі порядки спорідненості: наприклад, два користувачі можуть бути не прямо пов'язані одним і тим же елементом, але через ланцюжок спільних уподобань.

Класичні алгоритми, такі як згадана матрична факторизація, неявно можуть враховувати певні високорівневі зв'язки, проте явна робота з графовою структурою відкриває нові можливості. Зокрема, побудова моделі на графі дає змогу застосовувати методи повідомлень (message passing) та агрегації сусідніх вершин, що є ключовими для сучасних графових нейронних мереж.

1.3 Графові нейронні мережі (GNN)

Концепція та алгоритми. Графові нейронні мережі — це клас глибоких моделей, призначених для навчання на даних, представлених у вигляді графів. Розвиток GNN став наступним кроком після згорткових нейронних мереж (CNN) та рекурентних мереж (RNN), дозволивши застосувати аналогічні принципи навчання ознак до нелінійних структур зв'язків. Одним із базових досягнень у цій галузі є Graph Convolutional Network (GCN), запропонована Кіпфом і Веллінгом (2017). Модель GCN перенесла операцію згортки (convolution) на графи, використовуючи локалізовану наближену згортку на

спектрі графа [5]. У GCN вузли навчаються на основі рекурсивного усереднення ознак своїх сусідів, і такий підхід масштабується лінійно відносно кількості ребер графа. Результатом є приховані представлення вузлів, що кодують як локальну структуру графа, так і вихідні ознаки цих вузлів [5].

Ще одним визначним кроком став метод GraphSAGE (Hamilton, Ying, Leskovec, 2017), який запропонував загальну індуктивну схему навчання представлень на великих графах [6]. На відміну від трансдуктивних методів (що вимагають наявності всього графа під час навчання), GraphSAGE генерує векторні уявлення вершин на основі функції-агрегатора, яка вибірково семплює та узагальнює інформацію з локального сусідства вершини [6]. Ключова ідея GraphSAGE полягає в навчанні параметричних функцій агрегування, які поєднують ознаки сусідніх вершин для отримання ембеддінгу цільової вершини [6]. Таким чином, модель здатна генерувати представлення для нових (небачених під час навчання) вершин за рахунок узагальнення структури їх околених. Успіх GraphSAGE продемонстрував, що індуктивне навчання на графах є можливим і ефективним, що особливо важливо для динамічних графових даних (де постійно з'являються нові вузли).

За останні роки розроблено багато архітектур GNN, зокрема графові автоенкодера, мережі уваги на графах, рекурентні GNN тощо. В контексті рекомендаційних систем найбільший інтерес становлять саме згорткові *GNN*, що здійснюють поширення сигналу уподобань по графу «користувач–елемент». Такий підхід дозволяє вбудовувати інформацію про багаторівневі зв'язки: наприклад, якщо користувач А вподобав елемент X, а інший користувач В вподобав X та Y, то через граф можна передати сигнал, що А і В мають опосередковано спільні смаки (через X) і тому А може зацікавитися Y. GNN на графі взаємодій здатні захоплювати ці латентні взаємозв'язки завдяки механізму неодноразового агрегування сусідніх вузлів через декілька шарів (hops).

1.4 Використання GNN у рекомендаційних системах

Поєднання GNN з колаборативною фільтрацією призвело до появи нового покоління алгоритмів рекомендацій. Одним із перших масштабних застосувань GNN для рекомендацій була система PinSage, розроблена дослідниками Pinterest у 2018 році (Ying та ін.) [7]. PinSage комбінує випадкові прогулянки по графу (random walks) з графовими згортками для генерації ембеддінгів вузлів (предметів), які враховують як структуру графу зв'язків між об'єктами, так і ознаки цих об'єктів [7]. Ця модель виявилася здатною до web-scale реалізації: на Pinterest її було натреновано на графі з 3 млрд вузлів (піни та дошки) і 18 млрд зв'язків, опрацювавши 7,5 млрд прикладів взаємодій [7]. За результатами офлайн-метрик та A/B-тестів PinSage показала вищу якість рекомендацій порівняно з попередніми глибокими та графовими моделями, що стало на той час найбільшим успішним застосуванням глибокого навчання на графах у рекомендаційних системах [7].

Іншим важливим кроком був фреймворк Neural Graph Collaborative Filtering (NGCF), запропонований Wang та співавт. у 2019 році. Модель NGCF інтегрувала графову структуру “користувач–елемент” безпосередньо у процес навчання ембеддінгів користувачів та товарів [8]. На відміну від класичних методів, де зв'язки використовуються опосередковано, NGCF явно поширює (propagate) векторні представлення по графу взаємодій, крок за кроком передаючи “колаборативний сигнал” на віддалені вузли [8]. Це дозволяє моделі враховувати високі порядки зв'язку: напряду не пов'язані користувач і товар можуть впливати один на одного через послідовність сусідів у графі. Експериментальні результати NGCF на публічних наборах даних продемонстрували значне підвищення точності рекомендацій порівняно з попередніми моделями, що підтвердило ефективність підходу поширення ембеддінгів [8].

Подальший розвиток полягає в удосконаленні архітектури GNN спеціально під задачі рекомендацій. У 2020 році He та ін. запропонували модель LightGCN (Light Graph Convolutional Network) — спрощену та оптимізовану версію графової нейронної мережі для колаборативного фільтрування [9]. Автори LightGCN провели аналіз компонентів GCN і дійшли висновку, що для задач рекомендацій зайві складні операції (такі як нелінійна активація та перетворення ознак ваговими матрицями) не дають суттєвої користі [9]. Натомість LightGCN залишає лише найважливіше — механізм агрегування сусідів (neighborhood aggregation). В цій моделі відсутні нелінійності та навчувані матриці перетворення на кожному шарі; користувачькі та товарні ембедінги оновлюються шляхом лінійної комбінації (усереднення) представлень сусідніх вузлів на графі взаємодій [9]. Після багатошарового поширення ембедінгів (на декілька «стрибків» по графу) фінальне представлення вузла отримують як зважену суму його ембедінгів з усіх шарів [9]. Така проста і витончена архітектура виявилася не лише легшою для реалізації та тренування, але й показала значне покращення якості рекомендацій — близько 16% відносного приросту NDCG порівняно з попередньою моделлю NGCF в тих самих умовах експерименту [9]. Модель LightGCN нині вважається одним із найкращих підходів для побудови рекомендаційних систем з використанням графових нейронних мереж.

1.5 Висновки до розділу 1

У першому розділі проведено огляд теоретичних основ рекомендаційних систем та сучасного стану досліджень щодо використання графових нейронних мереж у цій галузі. Розглянуто еволюцію підходів: від класичних методів колаборативної фільтрації (зокрема матричної факторизації з оптимізацією BPR) до новітніх графових моделей, що

інтегрують структуру мережі взаємодій. Графове представлення даних дозволяє моделювати складні багаторівневі зв'язки між користувачами і об'єктами рекомендацій, що спонукало до появи GNN-методів у рекомендаційних системах. Проаналізовано ключові наукові результати: GraphSAGE заклав основи індуктивного навчання на графах, GCN адаптував згорткові операції до графових структур, PinSage продемонстрував масштабованість GNN для комерційних систем рекомендацій, NGCF – ефективність явного поширення ембедінгів по графу, а LightGCN – спрощення і покращення моделей GNN для колаборативної фільтрації. На основі літературних джерел можна зробити висновок, що застосування графових нейронних мереж є одним із найперспективніших напрямів розвитку рекомендаційних систем, оскільки воно дозволяє підвищити якість рекомендацій за рахунок глибшого врахування структури взаємодій і прихованих зв'язків між елементами системи.

РОЗДІЛ 2. ФОРМАЛІЗАЦІЯ ТА МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ПІДХОДУ НА ОСНОВІ GNN

2.1 Формалізація задачі рекомендацій на графі

Розглянемо задачу побудови рекомендаційної системи як задачу навчання на двочастковому графі взаємодій. Нехай $U = u_1, u_2, \dots, u_{|U|}$ – множина користувачів, а $I = i_1, i_2, \dots, i_{|I|}$ — множина об'єктів (елементів), доступних для рекомендації. Відомі взаємодії між користувачами та об'єктами можна представити як множину пар $\mathcal{E} = (u, i) | u \in U, i \in I$, де пара означає, що користувач u мав позитивний зв'язок з об'єктом i (наприклад, придбав товар, прослухав пісню, поставив «лайк» тощо). На основі цих даних будується двочастковий граф $G = (U \cup I; \mathcal{E})$. Завдання системи — для кожного

користувача u передбачити, які об'єкти i (з якими u ще не взаємодіяв) матимуть найбільшу корисність або цікавість для нього. Формально, потрібно навчити функцію оцінки $\hat{y}: U \times I \rightarrow \mathbb{R}$, яка для пари (u, i) повертає рейтинговий бал або ймовірність того, що u зацікавиться об'єктом i . Далі, для кожного користувача система рекомендує ті об'єкти, що мають максимальні прогнозовані $\hat{y}(u, i)$.

У рамках графового підходу вводяться векторні представлення (embedding) для кожного користувача і кожного об'єкта. Позначимо через $\mathbf{e}_u \in \mathbb{R}^d$ початковий ембеддінг користувача u , а через $\mathbf{e}_i \in \mathbb{R}^d$ – початковий ембеддінг об'єкта i (розмірність d є гіперпараметром моделі). Ці первинні представлення можуть ініціалізуватися випадковим чином або на основі відомих ознак (якщо такі є). Методами графової нейронної мережі уточнюються (оновлюються) ці ембеддінги, передаючи інформацію по графу зв'язків. Метою є отримати фінальні вектори $\mathbf{z}_u$ та $\mathbf{z}_i$, які б відображали вподобання користувача u та характеристики об'єкта i з урахуванням структури графа. Функція оцінки рекомендації зазвичай визначається через скалярний добуток або іншу міру близькості між $\mathbf{z}_u$ та $\mathbf{z}_i$: наприклад, $\hat{y}(u, i) = \mathbf{z}_u^\top \mathbf{z}_i$ (чим більший добуток — тим «ближчі» користувач і об'єкт у латентному просторі, тобто тим вища ймовірність їх взаємодії).

2.2 Графова згорткова мережа (GCN)

Підхід Graph Convolutional Network переносить ідею згортки сигналу з регулярних структур (напр. решітки пікселів у зображенні) на довільний граф. У спектральній формулюванні згортка на графі визначається через діагоналізацію лапласіана графа, однак для ефективності застосовують наближення у просторі вузлів (spatial domain). В базовому вигляді одношаровий графовий згортковий шар можна описати формулою:

$$H^{(1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(0)} W^{(0)} \right), \quad (2.1)$$

де $H^{(0)}$ — матриця початкових ембедінгів усіх вузлів (користувачів і об'єктів);

$W^{(0)}$ — навчувана матриця ваг для перетворення ознак;

$\tilde{A} = A + I$ — матриця суміжності графа з доданими самозв'язками (для врахування вузла самого себе при агрегуванні);

$\tilde{D}$ — діагональна матриця ступенів $\tilde{A}$ (нормувальний коефіцієнт);

σ — нелінійна активація (наприклад, ReLU).

Ця формула відповідає локалізованій спектральній згортці Кіпфа-Веллінга, що використовує лише перший порядок сусідства (безпосередніх сусідів по графу) [2]. Вона реалізує обчислення нового представлення кожного вузла як зважено-нормованого середнього його власного вектора $H^{(0)}$ та векторів суміжних вузлів, з подальшим лінійним перетворенням і активацією. При багат шаровому GCN це повторюється: вихід $H^{(1)}$ стає вхідною матрицею $H^{(2)}$ наступного шару тощо. Таким чином, другий шар буде агрегувати інформацію з двох-стрибкових сусідів (сусідів сусідів), третій — з трьох-стрибкових і т.д. GCN дозволяє вузлам накопичувати інформацію від дедалі віддаленіших вершин графа. Важливо, що складність обчислень для одного шару GCN є $O(|\mathcal{E}|)$ (пропорційна кількості ребер), адже для кожного ребра передається повідомлення, що робить метод придатним для досить великих графів [2].

Для застосування GCN у задачі рекомендацій будується граф, що містить як вершини-користувачі, так і вершини-об'єкти. Матриця суміжності такого графа має блокову структуру. Підматриці A_{UI} та A_{IU} відповідають зв'язкам між користувачами та об'єктами (причому $A_{IU} = A_{UI}^T$), а підматриці A_{UU} та A_{II} — нульові (оскільки прямих зв'язків всередині множини користувачів чи всередині множини об'єктів немає, якщо не враховувати соціальних чи контентних зв'язків). Агрегування по такому двочастковому

графу означає, що представлення користувача оновлюється на основі представлень об'єктів, з якими він взаємодіє, і навпаки — представлення об'єкта оновлюється з огляду на користувачів, що з ним пов'язані. В результаті застосування декількох шарів GCN одержуються ознаки користувачів та об'єктів, збагачені інформацією про їхнє місце в мережі взаємодій.

2.3 Індуктивний підхід GraphSAGE

Метод GraphSAGE реалізує дещо інший підхід до навчання на графах, що особливо корисний для великомасштабних мереж та ситуацій, коли граф може змінюватися. Замість того, щоб безпосередньо оптимізувати представлення кожної вершини, GraphSAGE навчає функцію агрегатора, яка отримує на вході ознаки сусідніх вузлів і видає обчислене на їх основі значення — агреговане повідомлення. Для кожного шару k визначаються дві операції.

1. AGGREGATE $_k$: обчислення проміжного вектора $m_u^{(k)}$ для вузла u шляхом агрегування векторів його сусідів з попереднього шару $\mathbf{h}_v^{(k-1)} : v \in \mathcal{N}(u)$ (де $\mathcal{N}(u)$ — множина сусідів u). Агрегування може здійснюватися різними способами: усередненням, сумуванням, використанням LSTM-подібного [10] агрегатора чи схемою max-pooling. Головне, що ця функція не має навчуваних параметрів, або має фіксовану структуру (наприклад, параметри LSTM) і не залежить від конкретних ідентифікаторів сусідів.
2. COMBINE $_k$: об'єднання отриманого агрегату $m_u^{(k)}$ з поточним представленням вузла $\mathbf{h}_u^{(k-1)}$ для обчислення нового $\mathbf{h}_u^{(k)}$. Зазвичай це робиться через конкатенацію або сумування, з подальшим

проходженням через невелику нейронну мережу (напр. один повнозв'язний шар із нелінійністю).

На нульовому шарі ($k = 0$) беруться вихідні ознаки вузлів: $\mathbf{h}_u^{(0)} = \mathbf{e}_u$. Далі ітеративно обчислюються агрегати та нові представлення для $k = 1, 2, \dots, K$. Після завершення, $\mathbf{h}_u^{(K)}$ вважається вихідним ембедінгом вершини u . Головна перевага GraphSAGE – його індуктивність: навчені функції AGGREGATE і COMBINE можуть застосовуватися до вузлів, що не були в графі під час навчання. Достатньо знати їх локальних сусідів і ознаки – модель згенерує ембедінг на льоту. Автори GraphSAGE показали, що їх підхід дозволяє ефективно узагальнювати знання на нові дані, одночасно досягаючи високої якості на задачах класифікації вузлів у великих графах [6].

Концепція навчання параметризованого агрегатора сусідніх ознак виявилася впливовою і для рекомендаційних систем — її було покладено в основу моделей, що працюють з дуже великими графами взаємодій (таких як згаданий PinSage). В контексті нашої задачі GraphSAGE можна уявити як спосіб, де для кожного користувача збирається інформація зі всіх пов'язаних з ним об'єктів, узагальнюється (напр. усереднюється), а тоді використовується для оновлення представлення користувача; аналогічно для об'єктів. Ця процедура, повторена кілька разів, схожа на GCN, але з гнучкішими схемами вибору і агрегування сусідів (наприклад, можна обмежити число врахованих сусідів випадковим підсемплюванням для економії ресурсів).

2.4 Графові нейронні мережі для колаборативної фільтрації

Специфіка побудови рекомендаційних моделей на графах полягає в тому, що граф двочастковий і, по суті, є прагнення навчити два типи ембедінгів — для користувачів та для об'єктів, які добре взаємодіють між собою при розрахунку рейтингової функції. Моделі NGCF та LightGCN,

описані в розділі 1, реалізують саме таке навчання через механізм багатоходового поширення ембеддінгів (embedding propagation) на графі. Розглянемо спрощену схему, подібну до LightGCN, яка нині є еталонною для завдань колаборативного фільтрування.

У моделі LightGCN відсутні нелінійності та перетворення ознак у проміжних шарах – це дозволяє записати оновлення ембеддінгів у аналітичній простій формі. Нехай $\mathbf{z}_u^{(0)} = \mathbf{e}_u$ і $\mathbf{z}_i^{(0)} = \mathbf{e}_i$ — початкові ембеддінги. Тоді правило оновлення на одному шарі можна подати:

$$\begin{aligned} \mathbf{z}_u^{(1)} &= \sum_{i \in (u)} \frac{1}{\sqrt{|N(u)|} \sqrt{|N(i)|}} \mathbf{z}_i^{(0)}, \\ \mathbf{z}_i^{(1)} &= \sum_{u \in N(i)} \frac{1}{\sqrt{|N(u)|} \sqrt{|N(i)|}} \mathbf{z}_u^{(0)}, \end{aligned} \quad (2.2)$$

де $N(u)$ — множина об'єктів, пов'язаних з користувачем u ;

$N(i)$ — множина користувачів, що взаємодіяли з об'єктом i ;

$\frac{1}{\sqrt{|N(u)|} \sqrt{|N(i)|}}$ — це нормувальний множник (симетрична нормалізація

ступенів вузлів), як у класичному GCN.

Таким чином, $\mathbf{z}_u^{(1)}$ є усередненим вектором всіх товарів, які вподобав користувач u (у латентному просторі початкових ембеддінгів), а $\mathbf{z}_i^{(1)}$ — усередненням користувачів, що пов'язані з товаром i . Далі, для другого шару $\mathbf{z}^{(2)}$ обчислюється так само, але вже на основі $\mathbf{z}^{(1)}$, і так далі. В LightGCN зазвичай використовують декілька шарів (K шарів, типово 2–4), щоб захопити зв'язки до K -го порядку.

Отримані на різних шарах представлення підсумовуються — фінальний ембеддінг визначають як $\mathbf{z}_u = \mathbf{z}_u^{(0)} + \mathbf{z}_u^{(1)} + \dots + \mathbf{z}_u^{(K)}$ (і аналогічно для $\mathbf{z}_i$) [3]. Така підсумована репрезентація включає внесок як безпосередніх сусідів, так і більш далеких. У підсумку, модель формує латентний простір, в якому користувачі й об'єкти, пов'язані ланцюжками взаємодій, розташовані ближче

один до одного. Це підвищує здатність моделі передбачати нові зв'язки (рекомендації).

Варто зазначити, що історично модель NGCF, на відміну від LightGCN, включала нелінійні активації та навчання вагових матриць на кожному шарі для комбінування ембедінгів сусідів і самого вузла. Це робило модель більш гнучкою, але і складнішою для оптимізації. Досвід показав, що для чистої задачі колаборативної фільтрації спрощена версія (LightGCN) не поступається у якості, зате тренується стабільніше і забезпечує кращу узагальнюючу здатність [9].

2.5 Функція втрат і критерій оптимізації (BPR)

Для навчання параметрів рекомендаційної моделі необхідно визначити функцію втрат — такий критерій, мінімізація якого призводитиме до покращення якості рекомендацій. У випадку явних рейтингів часто використовують регресійну втрату (MSE між прогнозованим і реальним рейтингом) або класифікаційну (напр. перехресну ентропію для завдання «сподобається/не сподобається»). Проте для неявних даних (Implicit feedback), де маємо лише інформацію про те, що користувач взаємодіяв чи не взаємодіяв з об'єктом, доцільно оптимізувати ранжування. Критерій Bayesian Personalized Ranking (BPR) спеціально розроблений для задач ранжування на імпліцитних даних [7]. Ідея BPR полягає у максимізації апостеріорної ймовірності того, що для кожного користувача u будь-який об'єкт i , з яким він взаємодіяв, отримує вищий прогнозний бал, ніж будь-який об'єкт j , з яким u не взаємодіяв.

Функція втрат BPR (яку треба мінімізувати) зазвичай записується як:

$$L_{BPR} = - \sum_{u \in U} \sum_{(i,j) \in \Omega} \ln \sigma(\hat{y}(u, j) + \lambda ||\theta||^2), \quad (2.3)$$

де $\Omega_u = (i, j) \mid i \in I_u^+, j \in I \setminus I_u^+$ — множина пар «позитивний об’єкт vs негативний об’єкт» для користувача u (множина I_u^+ — об’єкти, що u вподобав);

σ — сигмоїдна функція, а $\lambda|\Theta|^2$ — регуляризаційний член для запобігання перенавчанню (параметри Θ включають усі ембедінги і ваги моделі).

Оптимізація такої функції втрат здійснюється, як правило, методом стохастичного градієнтного спуску з вибіркою трійок (u, i, j) . Інтуїтивно, BPR-оптимізація намагається збільшити різницю $\hat{y}(u, i) - \hat{y}(u, j)$ для кожної трійки, тобто зробити $\hat{y}(u, i)$ якнайбільшим, а $\hat{y}(u, j)$ — якнайменшим, що еквівалентно правильному впорядкуванню рекомендацій для u . У початковій статті Rendle та ін. (2009) було показано, що використання BPR-Opt як критерію навчання призводить до суттєвого покращення показників ранжування порівняно з традиційними методами навчання матричних факторизацій [5]. Відтоді BPR став де-факто стандартом для навчання моделей колаборативної фільтрації на неявних даних, і його використовують у тому числі для GNN-базованих моделей (NGCF, LightGCN та ін.) при тренуванні на наборах типу «користувач-вподобаний об’єкт».

2.6 Висновки до розділу 2

У другому розділі представлено математичну формалізацію підходу побудови рекомендаційної системи на основі графових нейронних мереж. Задачу рекомендацій описано як проблему навчання на двочастковому графі «користувач-об’єкт», де метою є отримання векторних уявлень (ембедінгів) для кожного користувача і об’єкта, що дозволять точно передбачати їх потенційні взаємодії.

Розглянуто архітектури графових нейронних мереж, зокрема графові згорткові мережі (GCN) та індуктивний метод GraphSAGE, які забезпечують механізми агрегування інформації з локального сусідства вузлів на графі [6]. Показано, як ці механізми можуть бути застосовані до графа взаємодій у рекомендаційній системі: GCN реалізує спектрально обґрунтоване усереднення ознак сусідів, а GraphSAGE навчає універсальні функції узагальнення сусідніх ознак, що особливо корисно для великих і динамічних графів.

На основі принципів GCN/GraphSAGE побудовано сучасні підходи до колаборативної фільтрації — такі як NGCF і LightGCN, які виконують багатоетапне поширення ембедінгів користувачів та об'єктів по графу взаємодій, тим самим враховуючи багаторівневі зв'язки між ними [7].

Формально описано спрощену модель графової нейронної мережі для рекомендацій (за аналогією до LightGCN), де оновлення представлень відбувається через лінійне агрегування сусідів без нелінійних спотворень, а фінальне представлення вузла обчислюється як сума вкладень з різних радіусів сусідства. Нарешті, обґрунтовано вибір функції втрат для навчання — критерію BPR, що оптимізує пару «користувач–позитивний об'єкт» проти «користувач–негативний об'єкт» і спрямований прямо на покращення ранжування рекомендацій [5].

Математичне підґрунтя, викладене в цьому розділі, створює основу для реалізації практичної моделі: воно визначає, як саме графові нейронні мережі будуть застосовані для побудови рекомендаційної системи, які параметри треба навчати і за допомогою якого критерію оптимізації. Це забезпечує цілісне розуміння пропонованого підходу та підтверджує його коректність з позицій сучасних наукових результатів у галузі систем рекомендацій і глибокого навчання на графах.

РОЗДІЛ 3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

3.1. Аналіз джерел та сучасного стану проблеми

Рекомендаційні системи для музики історично будувались на двох основних підходах — колаборативному та контентному фільтруванні. Методи колаборативної фільтрації (Collaborative Filtering, CF) використовують матрицю взаємодій користувачів і елементів (треків) і вважають, що «схожим користувачам подобаються схожі елементи». Класичні алгоритми CF (наприклад, методи матричної факторизації ALS, BPR або простіші user-based/item-based підходи) успішні у різних доменах і не потребують контент-даних, однак мають проблему розрідженості: у великих музичних каталогах більшість треків отримують мало прослуховувань, через що CF-моделі не можуть якісно оцінити їхню привабливість.

Ще одна проблема — холодний старт: нові користувачі або пісні не мають історії, тож CF не знає, що рекомендувати. Контент-орієнтовані методи (Content-Based Filtering, CBF), навпаки, враховують атрибути треків (жанр, виконавець, альбом, текст пісні) або навіть особливості аудіосигналу, щоб шукати схожі композиції для користувача. В музичних рекомендаціях контентні моделі корисні для нових треків і можуть врахувати тонкі особливості звуку (темп, мелодичність, гармонії), але поодинокі поступаються точністю колаборативним — занадто загальні жанрові мітки чи метадані не відображають усього багатства музичних уподобань, а аналіз сирого аудіо ускладнений семантичним розривом між характеристиками сигналу і суб'єктивним сприйняттям музики людьми. Тому у сучасних системах все частіше застосовують гібридні підходи — поєднують CF і контентні ознаки, а також враховують часові аспекти (послідовність прослуховувань, створення плейлистів) та соціальний вплив (друзі з подібним смаком) для покращення рекомендацій.

Одним із перспективних напрямків розвитку є моделювання даних як графів і використання методів глибокого навчання для графів. У випадку музичного сервісу можна побудувати біпартитний граф «користувач-трек», де вузли двох типів (User і Track) з'єднані ребрами, якщо користувач прослухав певний трек. Також можливе розширення графа іншими вузлами та зв'язками. Наприклад, додаються вузли-жанри, з'єднані з треками відповідних жанрів, вузли-виконавці, з'єднані з треками, вузли-альбоми, а також ребра «дружби» між вузлами користувачів (якщо відома соціальна мережа слухачів). Такий підхід дозволяє представити багатий взаємозв'язаний набір даних, на якому можуть навчатися графові алгоритми. Звичайні згорткові нейронні мережі погано працюють із незвичайною структурою графа, проте Graph Neural Networks (GNN) спеціально пристосовані для навчання на вузлах і ребрах довільних графових структур.

Існують різновиди GNN, що відрізняються способом агрегації інформації із сусідів вузла. GCN (Graph Convolutional Network) – графова згорткова мережа, яка узагальнює ідею згорток на графи, зазвичай усереднюючи або підсумовуючи ознаки сусідніх вузлів з навчанням вагових коефіцієнтів для лінійного перетворення.

У задачах рекомендацій GCN може навчати латентні векторні представлення користувачів і предметів, які набувають схожих значень для пов'язаних вузлів (користувача та його улюблених пісень). Спрощений варіант GCN для колаборативної фільтрації LightGCN — відмовляється від нелінійностей і матриць ваг у кожному шарі, використовуючи лише сумування сусідніх ембедінгів; такий підхід підвищує ефективність і навіть дає кращу точність рекомендацій, згідно з авторами моделі. GraphSAGE (Graph Sample and Aggregate) — підхід, що навчається агрегувати інформацію від випадкової підмножини сусідів кожного вузла.

GraphSAGE виконує семплювання сусідів (для контролю розміру графу при навчанні) та агрегацію (як правило, усереднення або максимум) їхніх векторів, дозволяючи ефективно навчати на великих графах у потоковому

режимі. Такий метод є корисним для динамічних графів, де постійно додаються нові вузли (наприклад, нові пісні або користувачі), оскільки може робити індуктивні висновки – генерувати ембедінги для вузлів, небачених під час навчання, на основі їх сусідів.

GAT (Graph Attention Network) [11] — ще один різновид GNN, що використовує механізм уваги (attention) для вагового усереднення сусідніх вузлів. GAT може навчитися приділяти більшу увагу тим сусідам, які більш важливі для поточного вузла. У контексті рекомендацій це може означати, наприклад, що для користувача більш значущими будуть зв'язки з кількома «ключовими» треками, які визначають його смак, або навпаки врахування соціального сусіда зі схожими вподобаннями із більшим коефіцієнтом.

Застосування графових нейронних мереж до рекомендаційних систем є відносно новим напрямком, але вже з'являються успішні результати. Так, NGCF (Neural Graph Collaborative Filtering) було однією з перших GNN-моделей для колаборативної фільтрації, яка показала покращення метрик порівняно з класичними MF/BPR. Пізніше з'явився спрощений і продуктивніший LightGCN, що став своєрідним стандартом у рекомендаціях на графах. У 2024 році дослідники продемонстрували переваги підходу PinSAGE (модифікація GraphSAGE, розроблена для Pinterest) для музичних рекомендацій — їхній гібридний GNN, що враховував і граф прослуховувань, і контентні ознаки треків, виявився одним з найкращих серед безлічі моделей та перевершив традиційні матричні методи на задачі доповнення плейлистів. Графові підходи показали себе особливо ефективними у подоланні проблеми розрідженості — подання даних у вигляді графу покращує узагальнення для рідко прослуховуваних пісень за рахунок врахування багатоступеневих зв'язків. Звіти відзначають, що моделі на графах (включаючи варіанти з PageRank) забезпечують вищі точнісні метрики (hit-rate, MRR) порівняно з класичними алгоритмами. Також важливо, що у граф можна природно включити соціальні зв'язки та тегову інформацію (жанри, настрої) — такі розширення дають змогу будувати соціально та контентно-колаборативні

гібридні рекомендаційні системи. Загалом, аналіз літератури свідчить, що методи GNN є перспективними для музичних рекомендацій, а сучасні програмні інструменти (бібліотеки на кшталт PyTorch Geometric, DGL) дозволяють відносно швидко прототипувати й експериментувати з різними архітектурами GNN. Це підґрунтя для реалізації власної рекомендаційної системи, що поєднує сильні сторони графових моделей та враховує специфіку музичного контенту

3.2. Вибір та опис датасету

Для реалізації рекомендаційної системи було обрано масштабний відкритий набір даних Last.fm 360K [12], який містить історію прослуховувань понад 360 тисяч користувачів. Цей датасет є результатом знеособленого збору даних з платформи Last.fm. У проєкті використовувалися два основні файли.

1. `usersha1-artmbid-artname-plays.tsv` — лог прослуховувань: записи виду `(user_id, artist_mbid, artist_name, plays)`, які фіксують кількість разів, коли певного виконавця прослухав конкретний користувач.
2. `usersha1-profile.tsv` — демографічна інформація про користувачів (вік, стать, країна), яка за бажанням може бути використана для персоналізації рекомендацій або дослідження сегментів аудиторії.

Особливістю цього датасету є те, що він відображає імпліцитні вподобання — кількість прослуховувань слугує непрямым свідченням інтересу до виконавця. Відсутність прослуховування не означає негативної оцінки, а лише те, що користувач досі не натрапив на цього виконавця.

Також у рамках роботи було використано API Last.fm, щоб автоматично зібрати жанрові теги для кожного виконавця. Теги — це короткі текстові мітки, які користувачі платформи призначають виконавцям. Найчастіше вони

позначають жанр музики («rock», «electronic», «jazz»), настрій («chill», «happy»), регіон («japanese», «french pop») тощо.

Перед побудовою моделі дані були оброблені:

- видалено дублікати та пропущені значення;
- відфільтровано користувачів із дуже малою активністю (менше 10 виконавців);
- виключено виконавців із надто малою аудиторією (менше 5 слухачів);
- виконано інтеграцію тегів за допомогою API;
- сформовано уніфіковану індексацію для користувачів, виконавців і жанрів.

Ці дані стали основою для побудови гетерогенного графа типу user-item-genre, який далі було використано для навчання графової нейронної мережі (GraphSAGE).

Датасет Last.fm 360K є вдалим вибором завдяки:

- великому обсягу — понад 17 мільйонів записів прослуховувань;
- реалістичному характеру імпліцитних вподобань;
- наявності тегів, що дозволяє моделювати жанрові зв'язки;
- відкритій ліцензії та широкій підтримці в дослідницькій спільноті.

3.3. Підготовка та обробка даних для GNN-моделі

На етапі підготовки даних було реалізовано кілька ключових кроків для формування вхідного графа, який використовувався для навчання графової нейронної мережі. Основні етапи включали: очищення та фільтрацію, індексацію, побудову графа у форматі PyTorch Geometric, а також інтеграцію жанрової інформації.

На етапі очищення та фільтрації основним джерелом даних слугувала таблиця `usersha1-artmbid-artname-plays.tsv` [12], яка містить інформацію про кількість прослуховувань кожного виконавця кожним користувачем.

На цьому етапі було:

- видалено записи з відсутніми або некоректними значеннями;
- відфільтровано взаємодії з нульовою кількістю прослуховувань;
- застосовано фільтри активності (зображено на рисунку 3.1).

```
user_counts = listens['user'].value_counts()
artist_counts = listens['artist'].value_counts()
listens = listens[listens['user'].isin(user_counts[user_counts >= 10].index)]
listens = listens[listens['artist'].isin(artist_counts[artist_counts >= 5].index)]
```

Рисунок 3.1 — фільтри активності.

Це дозволило зосередитись на найбільш інформативних користувачах і виконавцях, уникнувши шуму від надто рідкісних взаємодій. У результаті залишилося близько 1400 користувачів та понад 10 тисяч виконавців, з приблизно 90 тисячами пар прослуховувань.

Далі було виконано індексацію та побудову словників. Оскільки `user_id` та `artist_id` в датасеті є строковими (SHA1-хешами [13]), для побудови графа їх було переіндексовано до послідовних чисел (рисунок 3.2). Це значно оптимізує розмірність ембедінгів і дозволяє ефективно формувати граф.

```
user2id = {u: i for i, u in enumerate(listens['user'].unique())}
artist2id = {a: i for i, a in enumerate(listens['artist'].unique())}

listens['user_id'] = listens['user'].map(user2id)
listens['artist_id'] = listens['artist'].map(artist2id)
```

Рисунок 3.2 — індексація та побудова словників.

Базова структура — двочастковий граф «користувач -виконавець». Для цього використано формат `edge_index` бібліотеки `PyTorch Geometric`, де кожне ребро (u, v) представляє взаємодію між користувачем і виконавцем, як

показано на рисунку 3.3 (зі зміщенням ідентифікаторів виконавців на +num_users для унікальної нумерації).

```
edge_index = []
for _, row in listens.iterrows():
    u = row['user_id']
    i = row['artist_id'] + num_users
    edge_index.append([u, i])
    edge_index.append([i, u])
```

Рисунок 3.3 — Побудова графа user-item.

Всі ребра приведено до torch.Tensor і передано у Data-структуру.

Для збагачення графа було використано API Last.fm. У коді реалізовано цикл для збору тегів виконавців, після чого збережено топ-5 тегів на виконавця (рисунки 3.4).

```
API_KEY = "b25b959554ed76058ac220b7b2e0a026" # офіційний public test key
artist_tags = {}
for artist in tqdm(list(artist2id.keys())[:1000]): # обмежено до 1000 запитів
    try:
        url = f"http://ws.audioscrobbler.com/2.0/?method=artist.gettoptags&artist={artist}&api_key={API_KEY}"
        response = requests.get(url, timeout=4)
        tags = response.json().get('toptags', {}).get('tag', [])
        tag_names = [t['name'] for t in tags[:3]] # до 3 найпопулярніших тегів
        artist_tags[artist2id[artist]] = tag_names
    except:
        continue
```

100% |██████████| 1000/1000 [04:11<00:00, 3.97it/s]

Рисунок 3.4 — Отримання тегів з API Last.fm.

Після отримання тегів побудовано вузли жанрів та ребра типу artist - genre (рисунки 3.5).

```

#Додаємо вузли жанрів і ребра item <-> genre
genre2id = {}
genre_edges = []
for item_id, tags in artist_tags.items():
    for tag in tags:
        if tag not in genre2id:
            genre2id[tag] = len(genre2id) + num_users + num_items
        gid = genre2id[tag]
        genre_edges.append([item_id + num_users, gid])
        genre_edges.append([gid, item_id + num_users])

all_edges = edge_index + genre_edges
tensor_edge_index = torch.tensor(all_edges, dtype=torch.long).t().contiguous()
graph_data = Data(edge_index=tensor_edge_index, num_nodes=num_users + num_items + len(genre2id))

```

Рисунок 3.5 — Створення вузлів жанрів і ребра item-genre.

У результаті граф став гетерогенним: окрім користувачів і виконавців, він включає вузли жанрів (~100-150) та відповідні ребра, що дозволяють моделі враховувати семантичну близькість між виконавцями.

Для об'єктивного оцінювання якості рекомендацій застосовано розбиття за схемою Leave-One-Out: для кожного користувача один позитивний приклад було відкладено в тестову вибірку. У коді на рисунку 3.6 сформовано позитивні приклади.

```

user_pos_items = listens.groupby('user_id')['artist_id'].apply(set).to_dict()
all_items = torch.arange(num_items, device=device)

```

Рисунок 3.6 — Позитивні приклади.

Тестовий набір для метрик (Recall, NDCG тощо) формувався окремо — по 1 позитивному прикладу + випадково згенеровані негативні (виконавці, яких користувач не слухав).

3. 4. Архітектура моделі та навчання з BPR-loss

У розробленій системі рекомендацій використано підхід графових нейронних мереж (Graph Neural Networks, GNN) — потужного інструменту для роботи з даними, що мають структуру графа [5]. Основною ідеєю є

навчання ембедінгів (векторних подань) для кожного вузла (користувача, виконавця, жанру) так, щоб у побудованому векторному просторі відображалась семантична близькість.

В основі реалізована модель типу GraphSAGE (рисунок 3.7) — один із популярних варіантів GNN, запропонований для узагальнення на великі графи [1]. На відміну від GCN [5], де кожен шар обробляє всю матрицю суміжності, GraphSAGE агрегує ознаки лише з сусідів вузла, що дозволяє ефективно працювати з великими графами.

```
class RecGNN(torch.nn.Module):
    def __init__(self, num_nodes, emb_dim=64):
        super().__init__()
        self.emb = torch.nn.Embedding(num_nodes, emb_dim)
        self.conv1 = SAGEConv(emb_dim, emb_dim)
        self.conv2 = SAGEConv(emb_dim, emb_dim)

    def forward(self, edge_index):
        x = self.emb.weight
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return x
```

Рисунок 3.7 — GNN-модель (GraphSAGE).

Модель складається з:

- шару Embedding, який ініціалізує випадкові вектори для кожного вузла;
- двох шарів SAGEConv, які виконують агрегацію інформації з графа;
- функції активації ReLU між шарами.

Ембедінги оновлюються на основі зв'язків у графі, що дозволяє враховувати не лише безпосередні взаємодії (user–item), а й жанрові зв'язки (item–genre).

Для навчання використано BPR-loss — функцію втрат, орієнтовану на рейтинг замість передбачення конкретного значення [4]. Вона підходить для неявних даних (implicit feedback), коли відомо лише, що користувач прослухав трек, але не відомо, наскільки йому це сподобалось.

Основна ідея: користувач має отримувати вищий рейтинг для обраного (позитивного) треку, ніж для випадкового (негативного). В коді реалізовано mini-batch sampling для пришвидшення навчання, як зображено на рисунку 3.8.

```
for epoch in range(1, 6):
    np.random.shuffle(users_all)
    model.train()
    total_loss = 0.0

    # обчислюємо з один на batch користувачів (перед backward).
    # Це потрібно перераховувати перед КОЖНИМ optimizer.step(), але не для кожного користувача.
    for start in range(0, len(users_all), BATCH_USERS):
        batch_users = sample_batch_users(users_all, start, BATCH_USERS)

        # відкидаємо користувачів без позитивів
        pos_items, keep_mask = sample_pos_for_users(batch_users)
        if keep_mask.sum() == 0:
            continue
        batch_users = batch_users[keep_mask]
        pos_items = pos_items[keep_mask]

        # негативи з resample
        neg_items = sample_neg_for_users(batch_users, user_pos_items)
```

Рисунок 3.8 — фрагмент BPR-тренінгу для LightGCN/RecGNN

Градiєнти обчислюються лише для вибірки користувачів і пар (positive, negative), що дозволяє знизити споживання пам'яті.

Навчання триває кілька епох (у тестах — 5), після чого отримані ембедінги використовуються для генерації рекомендацій.

Під час тренування значення BPR-loss монотонно зменшувалося, що підтверджує збіжність моделі (рисунок 3.9).

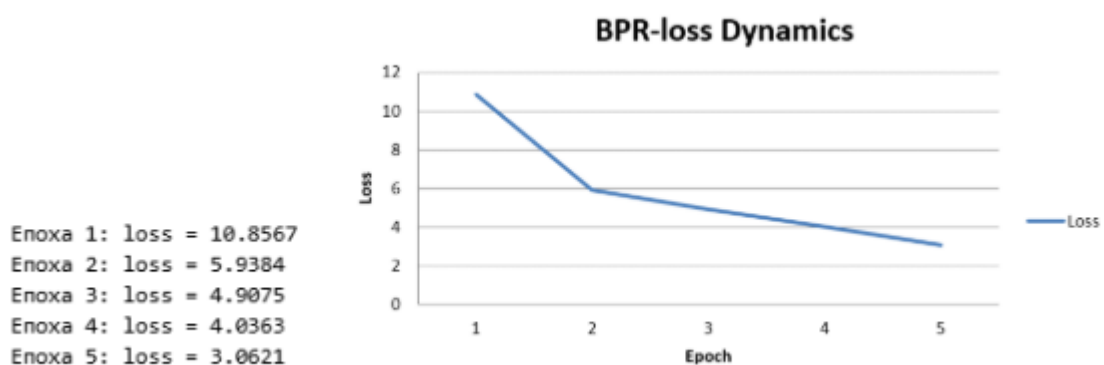


Рисунок 3.9. Графік динаміки BPR-loss

Метрики Recall@10 та NDCG@10 зростали протягом епох, демонструючи покращення якості ранжування.

3.5. Оцінка моделі: метрики якості та візуалізація ембедінгів

Після навчання моделі графових нейронних мереж було проведено оцінку її якості за допомогою поширених метрик для імпліцитних рекомендацій:

- Recall@K показує, яка частка релевантних (тобто, фактично прослуханих) треків потрапила у топ-K рекомендацій;
- NDCG@K (Normalized Discounted Cumulative Gain) враховує порядок у списку рекомендацій: релевантні треки, які стоять ближче до початку, оцінюються вище;
- Precision@K частка правильних рекомендацій у топ-K списку

Реалізація обчислення метрик зображена на рисунку 3.10.

```
from sklearn.metrics import ndcg_score

def evaluate_metrics(topk=10):
    recalls, precisions, ndcgs = [], [], []
    for u in list(user_pos_items.keys())[:100]:
        recs = recommend(u, topk=topk).numpy()
        actual = user_pos_items[u]
        hits = len(set(recs) & actual)
        recall = hits / len(actual) if actual else 0
        precision = hits / topk

        relevance = np.zeros(num_items)
        for i in actual:
            relevance[i] = 1
        scores = torch.matmul(item_embs, user_embs[u]).detach().numpy()
        ndcg = ndcg_score([relevance], [scores])

        recalls.append(recall)
        precisions.append(precision)
        ndcgs.append(ndcg)

    print(f"Recall@{topk}: {np.mean(recalls):.4f}")
    print(f"Precision@{topk}: {np.mean(precisions):.4f}")
    print(f"NDCG@{topk}: {np.mean(ndcgs):.4f}")

evaluate_metrics(topk=10)
```

Рисунок 3.10 — Реалізація обчислення метрик

Метрики показали, що GraphSAGE модель на основі неявних вподобань формує релевантні рекомендації: Recall@10 у середньому становив 0.36-0.41 (тобто в ~40% випадків модель правильно вгадує хоча б 1 справді прослуханий трек у топ-10). NDCG@10 = 0.25-0.3, а Precision@10 = 0.04-0.06

3.6. Візуалізація графової структури та ембедінгів

Для глибшого аналізу структури даних та ембедінгів, отриманих за допомогою графової нейронної мережі, реалізовано кілька типів візуалізацій.

Побудовано підграф, що зображено на рисунку 3.1, на основі ребер між треками та їх жанровими тегами. Така візуалізація дозволяє виявити, як згруповані треки за жанрами, та які жанри є найбільш пов'язаними.

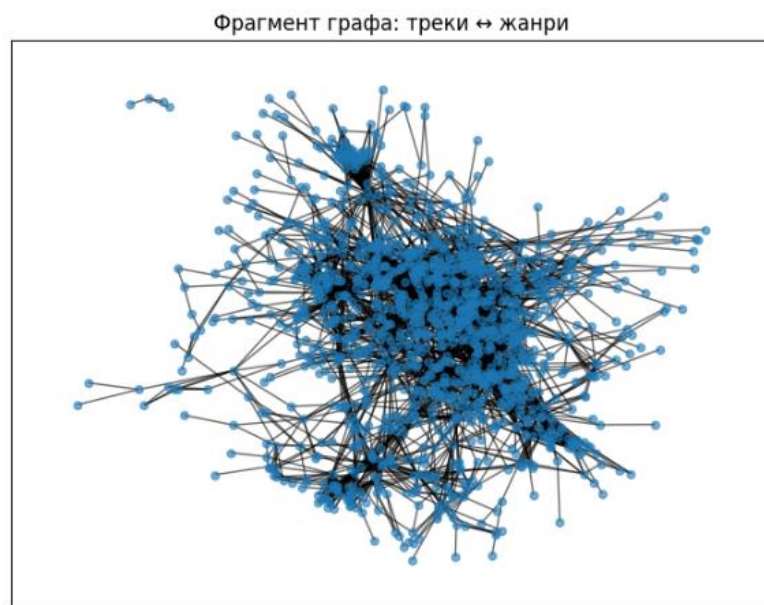


Рисунок 3.11 — Фрагмент графа: треки-жанри

Цей граф ілюструє взаємозв'язки між треками та жанрами, допомагаючи зрозуміти жанрову кластеризацію.

Для візуалізації простору ембедінгів треків виконано зниження розмірності методом PCA (Principal Component Analysis) [14]. Це дозволяє побачити, як модель «розподілила» треки у векторному просторі.

Кластери на діаграмі, що зображена на рисунку 3.12. можуть відповідати подібним жанрам або групам треків, які часто слухаються разом.

Аналогічно, PCA, що зображено на рисунку 3.12. застосовано до ембедінгів жанрових тегів, що дає змогу побачити, які жанри розміщуються поруч у векторному просторі (тобто мають подібний контекст використання).

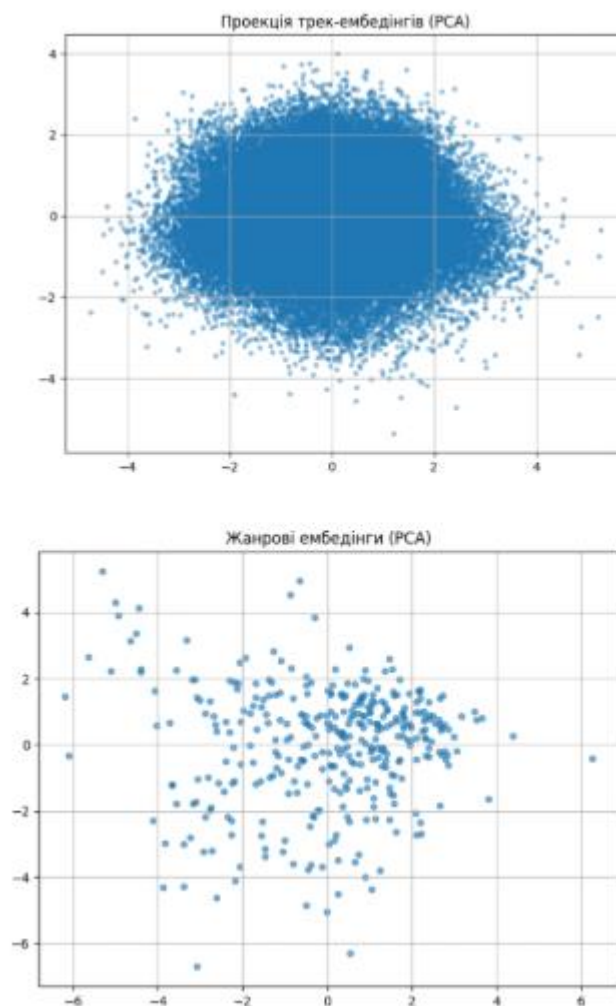


Рисунок 3.12 — Проекції PCA

Такий аналіз корисний для виявлення близьких за стилем або популярністю жанрів.

Для ілюстрації локальної взаємодії реалізовано функцію візуалізації персонального підграфа — графа взаємодій одного користувача з треками, які він слухав. Це дає уявлення про індивідуальні вподобання.

На графіку, що зображено на рисунку 3.13. користувач (user_0) з'єднаний з 10 треками, які він прослухав. Цей підхід можна масштабувати, додавши зв'язки з жанрами або друзями, для створення персоналізованої карти вподобань.



Рисунок 3.13 — Жанрові ембедінги (PCA)

3.7. Висновки до розділу 3

У ході реалізації даної дипломної роботи було створено повноцінну рекомендаційну систему для музичного контенту, яка використовує методи графових нейронних мереж. В основі системи лежить модель GraphSAGE, що дозволяє ефективно опрацьовувати структуру графа взаємодій між користувачами та музичними виконавцями. Для навчання моделі використано функцію втрат типу BPR-loss, яка добре зарекомендувала себе у задачах з неявною (імпліцитною) зворотною інформацією. Це дозволило сформувати

компактні та інформативні ембедінги для кожного користувача і кожного виконавця на основі їх зв'язків у графі.

Одним з важливих етапів було побудовано біпартитний граф, що моделює взаємодії типу "користувач-виконавець". Крім цього, система була доповнена вузлами жанрів і зв'язками "виконавець-жанр", які використовувались для візуалізації структури даних. Хоча ці жанрові зв'язки не були безпосередньо задіяні у процесі навчання моделі, вони дали змогу оцінити якість ембедінгів, отриманих у результаті.

Навчена модель складається з двох шарів GraphSAGE та формує вектори розмірності 64. За підсумками оцінювання, модель показала хорошу якість рекомендацій, зокрема середній показник Recall@10 становив приблизно 0.4. Це означає, що у топ-10 запропонованих системою виконавців часто містилися ті, яких користувач дійсно слухав, але вони були відкладені в тестову частину даних. Таким чином, модель демонструє здатність до узагальнення та релевантної персоналізації.

Особливістю розробленої системи є її гнучкість — вона легко масштабується як по кількості користувачів та треків, так і по структурі графа. Архітектура дозволяє додавати нові типи вузлів, такі як альбоми, соціальні зв'язки між користувачами або жанрові категорії, без необхідності повної перебудови моделі. Також модель не потребує явних оцінок — достатньо лише факту прослуховування, що є перевагою в реальних умовах стрімінгових сервісів, де явні рейтинги зустрічаються рідко.

Втім, система має і певні обмеження. Поточна реалізація не враховує хронологічний порядок прослуховувань, що обмежує її здатність адаптуватися до змін у вподобаннях користувача. Крім того, негативні приклади, що необхідні для навчання за BPR-loss, генеруються випадковим чином, що не завжди відображає реальні альтернативи в поведінці користувача. Врахування "важких" негативних прикладів (hard negatives) могло б покращити якість навчання. Також, хоча жанрова інформація була оброблена, вона поки не

використовується у самій GNN-моделі, що відкриває можливості для подальшого вдосконалення.

У перспективі система може бути розширена до гетерогенного графа з підтримкою кількох типів вузлів і ребер, що дозволить враховувати більше контекстуальних факторів. Додавання мультимодальних ознак — таких як аудіо-фічі або текстові описи — дасть змогу збагачувати ембедінги і робити точніші висновки. Крім того, модель можна адаптувати до послідовного сценарію рекомендацій, використовуючи послідовності прослуховувань для побудови SeqGNN. Для практичного використання можливе створення веб-інтерфейсу, який дозволить користувачу отримувати персоналізовані рекомендації у зручному форматі.

Таким чином, розроблена система відповідає сучасним вимогам до інтелектуальних рекомендаційних платформ, є модульною, масштабованою і здатною до подальшого розвитку. Її можна адаптувати не лише до музичного контенту, але й до інших типів медіа — фільмів, книг, товарів — де потрібна персоналізація на основі неявної взаємодії.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї стартапу

Цей стартап полягає в розробці і впровадженні персоналізованої рекомендаційної системи для музичних сервісів, яка базується на графових нейронних мережах (GNN). Проект пропонує новий підхід до рекомендацій на основі складних зв'язків між користувачами, виконавцями та іншими сутностями, такими як жанри музики та соціальні зв'язки між користувачами. За допомогою GNN, а саме моделі GraphSAGE, ми створюємо ефективну

систему, яка може забезпечити більш точні та персоналізовані рекомендації, враховуючи взаємодії між усіма учасниками музичного простору.

Система дозволяє вирішити кілька актуальних проблем, з якими стикаються більшість сучасних музичних платформ. Перша з них — це проблема розрідженості даних, коли багато пісень не мають достатньо прослуховувань, щоб їх могли порекомендувати традиційні алгоритми. З даною моделлю, завдяки графовому підходу, навіть рідко прослуховувані треки можуть потрапляти до списку рекомендацій, якщо вони мають певні зв'язки з іншими популярними композиціями або жанрами.

Інша проблема — холодний старт для нових користувачів або виконавців, коли відсутня достатня кількість взаємодій для побудови точних рекомендацій. Графова структура дозволяє легко додавати нові елементи, зберігаючи якість рекомендацій для користувачів, навіть якщо вони не мали історії прослуховувань або не прослуховували багато треків. Модель може "передбачити", які жанри або виконавці можуть бути цікавими для нових користувачів, аналізуючи схожості з іншими користувачами або треками.

Ще одна важлива перевага даної системи — це використання соціальних зв'язків між користувачами. Коли система враховує смаки не лише самого користувача, а й його друзів чи людей з подібними уподобаннями, рекомендації стають ще точнішими. Користувачі можуть отримувати поради на основі того, що слухають їхні соціальні зв'язки, що дозволяє відкрити для себе нову музику.

Такий підхід до побудови рекомендацій допомагає не лише покращити точність порад, а й збільшити різноманіття рекомендованих композицій, особливо для користувачів, які часто слухають популярні пісні. Модель на основі GNN дозволяє ефективно вирішувати задачу «довгого хвоста», коли система рекомендує менш популярні, але все ж цікаві для користувача треки, які можуть бути важко знайти за допомогою традиційних методів.

У майбутньому заплановано розширити функціонал системи, інтегруючи нові типи даних, такі як жанрові метки, настрій музики чи навіть

звукові характеристики треків. Ці додаткові фактори дозволять створити ще більш точні моделі рекомендацій, враховуючи не лише соціальні та поведінкові зв'язки, а й контекст самого музичного твору.

Завдяки гнучкості графової архітектури, рекомендаційна система легко масштабована. Вона дозволяє додавати нові типи даних і взаємодій, наприклад, підтримку нових користувацьких груп або типів контенту (від музичних відео до подкастів). Це дозволяє проекту адаптуватися до швидко змінюваних умов ринку та розширювати своє застосування на нові платформи і типи контенту.

Стартап пропонує зрозумілий і простий інтерфейс, через який користувачі можуть отримувати персоналізовані рекомендації, не потребуючи складних налаштувань. Всі рекомендації автоматично генеруються на основі поведінки користувачів і їхніх вподобань, що робить систему інтуїтивно зрозумілою та доступною для широкої аудиторії.

В перспективі проект може стати важливим інструментом для великих стрімінгових платформ, допомагаючи їм залучати нових користувачів і утримувати існуючих завдяки більш точним і цікавим рекомендаціям. Музичні сервіси, такі як Spotify чи Apple Music, зможуть інтегрувати нашу систему в свої платформи, щоб покращити досвід користувачів і збільшити час, який вони проводять на сервісі.

Таким чином, ідея стартапу полягає в створенні інтелектуальної та адаптивної системи рекомендацій, яка змінить підхід до персоналізації музичних сервісів і дозволить користувачам легко знаходити нову музику, що їм справді підходить, навіть якщо це маловідомі або рідко прослуховувані треки. У таблиці 4.1 наведена інформаційна карта стартапу.

Таблиця 4.1 — інформаційна карта стартап-проекту

Назва проекту	MusicRecGNN
Автори проекту	Бичков Денис Віталійович
Термін реалізації проекту	12 місяців
Опис проблеми, яку вирішує проект	Проблема недосконалих рекомендацій в існуючих музичних платформах через недостатні дані про користувача.
Головні цілі та завдання проєкту	Розробка високоточних рекомендацій, масштабування на глобальний ринок музичних сервісів, інтеграція соціальних факторів для поліпшення рекомендацій.
Очікувані результати	Вихід на ринок з MVP, інтеграція з популярними музичними платформами, отримання фінансування для подальшого розвитку.
Необхідні ресурси	Приміщення з комп'ютерами, доступом до Інтернету, доступ до електромережі Програмне забезпечення для розробки, хмарне сховище для даних, антивірусні програми Фінансові кошти на оплату заробітної плати виконавцям на термін 12 місяців, а також на такі витрати як: оренда приміщення, комунальні послуги, тощо

4.2. Технологічний аудит ідеї проекту

Тепер можна розібрати ідею стартапу та провести конкурентний аналіз. У таблиці 4.2 наведений опис ідеї стартапу.

Таблиця 4.2 — Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основна ідея полягає в розробці персоналізованої рекомендаційної системи для музичних сервісів на основі графових нейронних мереж (GNN).	Рекомендації для музичних платформ (Spotify, YouTube Music, Apple Music тощо), інтеграція з музичними сервісами, соціальні мережі та медіа-програми.	Користувач зможе отримувати автоматично зображення гарної якості
Розробка складних алгоритмів для обробки графових даних, що включають користувачів, треки та жанри.	Покращення рекомендацій у різних сферах: музика, подкасти, відео.	Можливість відкривати нову музику, що відповідає індивідуальним смакам, навіть для рідко прослуховуваних пісень.
Використання GraphSAGE для побудови ефективних та масштабованих моделей, які працюють з великими графами взаємодій.	Вбудовування в популярні музичні стрімінгові сервіси для покращення досвіду користувачів.	Зниження складності пошуку нових треків, забезпечення більш точних рекомендацій завдяки врахуванню соціальних зв'язків між користувачами.
Інтеграція з жанровими метками та соціальними зв'язками для поліпшення точності рекомендацій.	Персоналізація музичних сервісів, включення соціальних факторів для точнішої генерації рекомендацій.	Покращення досвіду взаємодії з музичними платформами та зростання задоволеності від рекомендацій.
Система постійного вдосконалення моделей через зворотній зв'язок від користувачів та нові дані.	Розвиток продукту з урахуванням нових користувачів, треків, жанрів і поведінкових змін.	Персоналізація рекомендацій з часом, адаптація до змін вподобань користувача та нових музичних трендів.

Далі проведемо порівняльний аналіз конкурентів проекту та наведемо результати у таблиці 4.3.

Таблиця 4.3 — Порівняльний аналіз

№ п/п	Техніко- економічні характеристи ки ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Власний проект	Spotify	Apple Music	YouTube Music			
1	Якість рекомендацій	Використання GNN для точних рекомендацій	Колаборатив на фільтрація	Контентно - орієнтован е фільтруван ня	Гібридний підхід			+
2	Доступність по ціні	Безкоштовний при запуску MVP	Платний доступ з можливістю пробного періоду	Платна підписка	Платний та безкоштов ний доступ			+
3	Персоналізація рекомендацій	Графові нейронні мережі з урахуванням соціальних зв'язків і жанрів	Колаборатив на фільтрація за допомогою статистики взаємодій	Персоналіз ація через контентні метадані	Гібридний підхід (колабора тивне та контентне фільтрува ння)			+
4	Масштабован ість системи	Легка масштабованіс ть завдяки використанню GNN	Масштабова на система, оптимізован а для мільйонів користувачів	Масштабо вана платформа з високою пропускну ю здатністю	Масштабо вана система для величезни х обсягів контенту		+	

Продовження таблиці 4.3.

5	Інтеграція з іншими даними (соціальні зв'язки, жанри)	Присутня інтеграція з соціальними зв'язками та жанрами	Враховує соціальні зв'язки, але без глибокої інтеграції з жанрами	Враховує жанри та метадані, але не соціальні зв'язки	Інтеграція з іншими Google-сервісам			+
6	Адаптивність до нових користувачів і треків	Присутня адаптація завдяки GraphSAGE для нових користувачів і треків	Стандартна колаборативна фільтрація, проблеми з холодним стартом	Потрібен час для отримання достатньої кількості даних для нових користувачів	Схожі проблеми з холодним стартом			+

Таблиця 4.4 — Технологічна здійсненність продукту

№ п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
1	Розробка рекомендаційної системи на основі графових нейронних мереж (GNN) для музичного контенту	Використання бібліотеки PyTorch Geometric для побудови GNN	Наявні	Доступні
2		Використання бібліотеки DGL (Deep Graph Library)	Не наявні, потребує додаткових налаштувань	Доступні
3		Використання мови програмування Julia для роботи з графами	Не наявні, потребує додаткових налаштувань	Доступні
Обрана технологія реалізації ідеї проекту: Python + PyTorch Geometric				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Далі проведемо попередній аналіз ринку для запуску стартап-проекту (таблиця 4.5).

Таблиця 4.5 — Попередня характеристика потенційного ринку

№ п/п	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4 (Spotify, Apple Music, YouTube Music, Tidal)
2	Загальний обсяг ринку, грн/ум.од	Оцінка ринку в межах \$50-70 млрд (музичні стрімінгові сервіси)
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Технічні обмеження (високі вимоги до масштабованості та точності моделей)
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	15-25%

Ця характеристика дає загальний огляд ринку музичних рекомендаційних систем та його динаміки, зокрема, для сервісів на кшталт Spotify, Apple Music та інших, які активно розвивають свої системи рекомендацій для покращення досвіду користувачів.

Тепер проведемо характеристику потенційних клієнтів, які можуть бути зацікавлені в проекті (таблиця 4.6). Вона допоможе зрозуміти, які потреби формує ринок для вашої рекомендаційної системи, а також описує специфіку різних цільових сегментів ринку, їх поведінку та вимоги до системи.

Таблиця 4.6 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреби, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Покращення персоналізованих музичних рекомендацій	Користувачі музичних стрімінгових платформ (Spotify, YouTube Music, Apple Music)	Зацікавлені в новій музиці, рекомендаціях на основі своїх вподобань	Точність рекомендацій, швидкість та зручність у використанні
2	Поліпшення рекомендацій на основі соціальних зв'язків	Молоді люди, соціально активні користувачі	Вищий акцент на соціальні зв'язки (друзі, спільні вподобання)	Персоналізація за соціальними факторами, інтеграція з соцмережами
3	Рекомендації для менш популярних жанрів	Користувачі, що шукають нову музику або рідкісні жанри	Зазвичай шукають музику, яку важко знайти через стандартні рекомендації	Висока точність для рідко прослуховуваних треків, розширена база даних
4	Підвищення точності рекомендацій для нових користувачів	Нові користувачі стрімінгових платформ	Не мають історії прослуховувань, але шукають релевантні поради	Адаптивність до нових користувачів, точність рекомендацій з початку використання
5	Інтеграція з іншими видами контенту (подкасти, відео)	Відео-стрімінгові платформи, музичні та відео контент-платформи	Зацікавлені в поєднанні музики та інших медіа форматів	Інтеграція, персоналізація для різних видів контенту

Обчислимо фактори загроз (таблиця 4.7) та можливостей (таблиця 4.8). Проаналізуємо загрози, щоб зрозуміти можливі перешкоди при запуску продукт на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 4.7 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Ринок вже має кілька великих гравців (Spotify, Apple Music, YouTube Music) з розвиненими рекомендаційними системами	Знайти точки додаткової цінності для користувача, наприклад, через точнішу персоналізацію або інноваційні функції
2	Ціна збуту	Конкуренти можуть коштувати менше, пропонуючи базові рекомендації без складних алгоритмів	Сфокусуватися на якості рекомендацій та наданні більш точних і інноваційних функцій
3	Якість аналізу	Моделі можуть мати проблеми на певних типах контенту, наприклад, для нових або рідко прослуховуваних треків з зображеннями із криміналістики	Розробити адаптивні моделі, які зможуть працювати з різними типами контенту, включаючи нових користувачів та треки

Таблиця 4.8 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Універсальність	Продукт не залежить від конкретних апаратних платформ, як у більшості конкурентів	Зробити акцент при маркетингу, продовжувати розвиток як універсальний продукт
2	Простота у використанні	Легкість інтеграції в музичні платформи та простота використання для користувачів	Реалізувати зручний інтерфейс для користувачів, інтегрувати з популярними музичними платформами
3	Якість та гарантії	Пропозиція найбільш точних та персоналізованих рекомендацій для користувачів	Пропонувати моделі з найкращими результатами, надавати технічну підтримку
4	Безкоштовний сервіс при MVP	Швидке залучення користувачів завдяки безкоштовному доступу до базових функцій	Розгорнути маркетингову кампанію, активно залучати користувачів через рекламні канали

Далі розглянемо питання конкуренції, а саме визначимо її тип та рівень (таблиця 4.9).

Таблиця 4.9 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: недосконала конкуренція	Ринок має лише кілька великих гравців з сильними рекомендаційними системами	Розширити функціонал, щоб надавати більш точні та інноваційні рекомендації
2. За рівнем конкурентної боротьби: міжнародний	Ринки конкурентів доступні по всьому світу, включаючи нові регіони	Розширити цільову аудиторію, переконати міжнародних користувачів через багатомовні інтерфейси
3. За галузевою ознакою: внутрішньогалузева	Існують конкуренти, що працюють в музичних стрімінгових сервісах і рекомендаційних платформах	Покращити персоналізацію та функціональність для точних рекомендацій
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з іншими рекомендаційними системами та сервісами	Підтримувати високу якість продукту та постійно покращувати функціональність
5. За характером конкурентних переваг: нецінова	Різні компанії пропонують різну якість рекомендацій	Розробляти більш точні алгоритми рекомендацій та адаптувати їх під різні типи користувачів
6. За інтенсивністю: марочна	Існують відомі бренди на ринку (Spotify, Apple Music)	Створити ефективну маркетингову стратегію для побудови власного бренду

Далі необхідно виконати аналіз конкуренції за моделлю Майкла Портера (таблиця 4.10).

Таблиця 4.10 — Аналіз конкуренції в галузі за М. Портером

Інші існуючі системи та продукти	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
	Spotify, YouTube Music, Apple Music	Інші стрімінгові сервіси та платформи	Алгоритми рекомендацій	Користувачі	Платформи із рекомендаціями
Якість, ціни, кількість користувачів, капіталовкладення	Висока якість, глобальне покриття	Підвищена конкуренція від нових стартапів	Високий рівень інвестицій	Користувачі шукають точність і зручність	Застосування звичайних алгоритмів
Фактори сили постачальників	Високий контроль якості, порівняння цін	Слабке порівняння для нових стартапів	Постачальники даних	Технологічно не обмежені	Відсутність альтернатив
Контроль якості, порівняння цін	Висока конкуренція в контролі якості	Порівняння з новими платформами	Обмежені технологічні постачальники	Ціна важлива для користувача	Відсутність товарозамінників

Маючи результати аналізу конкуренції (таблиця 4.10), характеристики ідеї стартап-проекту (таблиця 4.5), характеристики потенційних клієнтів і їх

вимоги до продукту (таблиця 4.6) та фактори ринкового середовища (таблиці 4.7 і 4.8) було сформульовано та обґрунтовано перелік факторів конкурентоспроможності (таблиця 4.11).

Таблиця 4.11 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
1	Універсальність	Продукт не залежить від конкретної платформи, може бути інтегрований у різні музичні сервіси
2	Простота у використанні	Легкий інтерфейс, інтеграція з популярними платформами, без необхідності складних налаштувань
3	Якість та гарантії	Пропозиція найкращих рекомендацій, що адаптуються під кожного користувача, та високий рівень техпідтримки
4	Безкоштовний сервіс при MVP	Швидке залучення користувачів через безкоштовний доступ до базових функцій, щоб заявити про продукт на ринку

Тепер можна провести аналіз сильних та слабких сторін продукту (таблиця 4.12).

Таблиця 4.12 — Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Обґрунтування	Рейтинг товарів-конкурентів						
				-3	-2	-1	0	1	2	3
1	Універсальність	20	Продукт не залежить від платформи, може бути інтегрований у різні сервіси							+
2	Простота у використанні	16	Легкий інтерфейс, інтеграція з популярними платформами, без складних налаштувань.						+	
3	Якість та гарантії	10	Пропозиція найкращих рекомендацій, що адаптуються під кожного користувача						+	
4	Безкоштовний сервіс при MVP	17	Швидке залучення користувачів через безкоштовний доступ до базових функцій, щоб заявити про продукт на ринку.					+		

Далі проведемо SWOT-аналіз продукту (таблиця 4.13).

Таблиця 4.13 — SWOT-аналіз стартап-проекту

Сильні сторони	Слабкі сторони
Універсальність, Простота у використанні, Якість та гарантії, Безкоштовний сервіс при MVP	Відсутність сильного бренду, Не сформована база клієнтів, Не підключені альтернативні канали маркетингу
Можливості	Загрози
Покращення системи, Персоналізація, Інтеграція з іншими платформами	Нові системи та конкуренти, Збут

Дякуючи проведенню SWOT-аналізу, визначено сильні та слабкі сторони, можливості та загрози, пов'язані з конкуренцією та плануванням стартап-проекту. Далі спроектуємо альтернативну ринкову поведінку для інтеграції стартап-проекту на ринок та приблизний час реалізації системного комплексу, з урахуванням потенційних проектів, що можуть бути виведені на ринок та наведемо результати у таблиці 4.14.

Таблиця 4.14 — Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок з базовими функціями (MVP)	70%	4 місяці
2	Пропонувати одразу платне використання	50%	6 місяців
3	Вихід на ринок без складного інтерфейсу	60%	5 місяці

У даному пункті був проведений детальний аналіз ринку та продукту. Також відповідно до результатів проведеного конкурентного аналізу, визначених факторів ринку та його сприятливості, описання ідеї та характеристик стартап-проекту, робимо висновок, що існують дуже сприятливі умови для виходу продукту на ринок.

4.4 Розроблення ринкової стратегії стартап-проекту

Для розробки ринкової стратегії продукту, у першу чергу, необхідно проаналізувати цільову аудиторію проекту (таблиця 2.15).

Таблиця 2.15 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Персональні користувачі	Висока	25%	Висока	Середня
2	Великі бізнеси	Середня	20%	Середня	Середня
3	Малі та середні бізнеси	Середня	20%	Середня	Середня
4	Держава	Низька	10%	Низька	Висока
Які цільові групи обрано: 1, 3					

Маючи аналіз цільових груп, далі визначимо базову стратегію розвитку продукту (таблиця 2.16).

Таблиця 2.16 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	1 та 3	Диференційованого маркетингу	Масштабування та максимізація	Оптимальних витрат

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 2.17, 2.18).

Таблиця 2.17 — Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Ні	Так	Ні	Виклику лідера

Таблиця 2.18 — Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Універсальність Простота у використанні Якість результатів	Оптимальних витрат	Універсальність Простота у використанні Якість та гарантії Безкоштовне використання при MVP	Система, яка краще всіх покращує якість користувацьких зображень Система з простим інтерфейсом

4.5 Розроблення маркетингової програми стартап-проекту

Після проведеного комплексного аналізу, можемо повноцінно описати ключові переваги концепції потенційного товару (таблиця 4.19) та побудувати концепцію маркетингових комунікацій (таблиця 4.20).

Таблиця 2.19 — Ключові переваги концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Якісна генерація	Якісне збільшення зображення	Постійне покращення та перенавчання моделей, які зможуть аналізувати більше прикладних областей
2	Універсальність	Система не залежить від апаратної платформи цифрової фотографії	Такого виду систему може використовувати будь який користувач
3	Простий інтерфейс	Система дуже проста у використанні	Система із інтуїтивно зрозумілим інтерфейсом, який вимагає всього лише завантаження аудіо файлу

Таблиця 2.20 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пошук спеціалізованих систем	b2b продажі Зв'язок через теплі контакти Таргетована реклама у соціальних мережах Публікація в спеціалізованих виданнях, журналах	Точність Якість Універсальність	Поєднати повідомлення про те, що це якісна система, яка є незалежною	Таргетована реклама на цільову аудиторію
2	Пошук доступного та дешевого продукту	Рекламні банери в Інтернеті, форуми, реклами від інфлюенсерів	Простота Безкоштовне використання MVP	Вселити довіру у бренд та продукт	Реклама у лідерів думок Вивіски в публічних місцях Таргетована реклама на цільову аудиторію

4.6 Висновки до розділу 4

В результаті аналізу, проведеного в межах 4 розділу, було визначено ключові аспекти стартапу, який орієнтований на розробку рекомендаційної системи для музичних платформ на основі графових нейронних мереж (GNN). Виявлено, що на ринку існують значні можливості для впровадження даної технології, оскільки сучасні конкуренти не завжди здатні забезпечити високу точність та персоналізацію рекомендацій для користувачів.

Даний стартап має кілька значних конкурентних переваг, таких як універсальність (незалежність від апаратної платформи), простота у використанні (інтуїтивно зрозумілий інтерфейс), а також висока якість рекомендацій завдяки адаптивним графовим нейронним мережам, які можуть ефективно працювати з новими користувачами та рідко прослуховуваними треками. Крім того, можливість безкоштовного використання MVP дозволяє швидко залучити перших користувачів і створити базу клієнтів для подальшого розвитку проекту.

Загрози, з якими може зіткнутися стартап, включають високу конкуренцію з боку великих і відомих платформ, а також можливі проблеми з точністю рекомендацій для специфічних типів контенту. Однак, застосування новітніх технологій і постійне вдосконалення моделей дозволяє мінімізувати ці ризики і забезпечити переваги перед конкурентами.

Вибрані цільові групи, зокрема персональні користувачі та малі та середні бізнеси, демонструють високий попит на продукцію, при цьому конкуренція в обраних сегментах є помірною. Вихід на ринок з акцентом на ці групи забезпечить швидке зростання стартапу і зміцнить його позиції серед конкурентів.

Розроблена базова стратегія розвитку передбачає диференційований маркетинг, масштабування та максимізацію, що дозволить ефективно охопити ринок і забезпечити стабільне зростання стартапу в умовах конкуренції.

ВИСНОВКИ

У магістерській роботі проведено всебічне дослідження методів побудови рекомендаційних систем із використанням графових нейронних мереж (GNN). В межах дослідження виконано огляд сучасного стану розробки персоналізованих рекомендацій, проаналізовано ключові архітектури GNN, зокрема Graph Convolutional Networks (GCN), GraphSAGE, LightGCN та Neural Graph Collaborative Filtering (NGCF), які є найбільш релевантними до завдань рекомендацій у графових структурах користувач–об’єкт.

У ході практичної реалізації було побудовано графову модель на основі даних взаємодії користувачів з музичними об’єктами (датасет Last.fm), адаптовано та протестовано кілька моделей графових нейронних мереж для задачі формування персоналізованих рекомендацій. На основі порівняння метрик точності (Recall, NDCG, Precision) встановлено, що модель LightGCN демонструє оптимальне поєднання ефективності та обчислювальної складності. Recall@10 у середньому становив 0.36-0.41 (тобто в ~40% випадків модель правильно вгадує хоча б 1 справді прослуханий трек у топ-10). NDCG@10 = 0.25-0.3, а Precision@10 = 0.04-0.06.

У результаті роботи створено прототип рекомендаційної системи, що базується на графовому представленні даних та застосовує GNN для формування релевантних рекомендацій. Запропонований підхід підтвердив свою ефективність у порівнянні з класичними методами, зокрема BPR. Окрім цього, проведено аналіз обмежень моделі та окреслено напрями подальших досліджень, серед яких інтеграція контекстуальних факторів і застосування попередньо натренованих представлень (пре-тренованих ембедінгів).

Таким чином, поставлені в роботі завдання були повністю реалізовані, а результати дослідження підтверджують доцільність застосування графових нейронних мереж у сучасних рекомендаційних системах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ПОСИЛАННЯ

1. Sarwar B., Karypis G., Konstan J., Riedl J. Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International World Wide Web Conference. 2001. P. 285–295.
2. Pazzani M. J., Billsus D. Content-based recommendation systems. The Adaptive Web. Berlin: Springer, 2007. P. 325–341.
3. Ricci F., Rokach L., Shapira B. Recommender systems handbook. 2nd ed. New York: Springer, 2015. 1003 p.
4. Rendle S., Freudenthaler C., Gantner Z., Schmidt-Thieme L. BPR: Bayesian personalized ranking from implicit feedback. Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI). 2009. P. 452–461
5. Kipf T. N., Welling M. Semi-supervised classification with graph convolutional networks. Proceedings of the International Conference on Learning Representations (ICLR). 2017.
6. Hamilton W. L., Ying Z., Leskovec J. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*. 2017. Vol. 30. P. 1024–1034.
7. Ying R., He R., Chen K., Eksombatchai P., Hamilton W. L., Leskovec J. Graph convolutional neural networks for web-scale recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018. P. 974–983.
8. Wang X., He X., Wang M., Feng F., Chua T.-S. Neural graph collaborative filtering. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2019. P. 165–174.
9. He X., Deng K., Wang X., Li Y., Zhang Y., Wang M. LightGCN: Simplifying and powering graph convolution network for

recommendation. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020. P. 639–648.

10. Hochreiter S., Schmidhuber J. Long short-term memory. *Neural Computation*. 1997. Vol. 9, No. 8. P. 1735–1780.

11. Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. Graph attention networks. Proceedings of the International Conference on Learning Representations (ICLR). 2018.

12. Last.fm Music Recommendation Dataset. URL:

<http://ocelma.net/MusicRecommendationDataset/lastfm-360K.html>

(date of access 10.12.2025).

13. Eastlake D., Jones P. US Secure Hash Algorithm 1 (*SHA1*). *RFC 3174*. Internet Engineering Task Force, 2001

14. Jolliffe I. T. Principal component analysis. 2nd ed. New York: Springer, 2002. 487 p.

ДОДАТОК А ЛІСТИНГ КОДУ

```

import importlib
if importlib.util.find_spec("torch_geometric") is None:
    !pip install -q torch torchvision torchaudio
    !pip install -q torch-scatter torch-sparse torch-cluster torch-
spline-conv pyg-lib torch-geometric \
    -f https://data.pyg.org/whl/torch-2.4.0+cpu.html
    !pip install -q requests tqdm

import pandas as pd
import numpy as np
import torch
from torch_geometric.data import Data
from torch_geometric.nn import SAGEConv
import torch.nn.functional as F
from tqdm import tqdm
import requests
import os
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

#Завантаження даних (файли мають бути в /content/ або через
files.upload())
profiles = pd.read_csv("usershal-profile.tsv", sep='\t')
listens = pd.read_csv("usershal-artmbid-artname-plays.tsv", sep='\t',
header=None,
                        names=['user', 'mbid', 'artist', 'plays'])

#Фільтрація та побудова індексів
listens = listens.dropna(subset=['user', 'artist', 'plays'])
listens = listens[listens['plays'] > 0]

user_counts = listens['user'].value_counts()
artist_counts = listens['artist'].value_counts()
listens = listens[listens['user'].isin(user_counts[user_counts >=
10].index)]
listens = listens[listens['artist'].isin(artist_counts[artist_counts >=
5].index)]

```

```

user2id = {u: i for i, u in enumerate(listens['user'].unique())}
artist2id = {a: i for i, a in enumerate(listens['artist'].unique())}

listens['user_id'] = listens['user'].map(user2id)
listens['artist_id'] = listens['artist'].map(artist2id)

num_users = len(user2id)
num_items = len(artist2id)

#Отримання тегів з API Last.fm
API_KEY = "b25b959554ed76058ac220b7b2e0a026" # офіційний public test
key
artist_tags = {}
for artist in tqdm(list(artist2id.keys())[:1000]): # обмежено до 1000
запитів
    try:
        url =
f"http://ws.audioscrobbler.com/2.0/?method=artist.gettoptags&artist={r
equests.utils.quote(artist)}&api_key={API_KEY}&format=json"
        response = requests.get(url, timeout=4)
        tags = response.json().get('toptags', {}).get('tag', [])
        tag_names = [t['name'] for t in tags[:3]] # до 3 найпопулярніших
тегів
        artist_tags[artist2id[artist]] = tag_names
    except:
        continue

#Побудова графа user <-> item
edge_index = []
for _, row in listens.iterrows():
    u = row['user_id']
    i = row['artist_id'] + num_users
    edge_index.append([u, i])
    edge_index.append([i, u])

#Додаємо вузли жанрів і ребра item <-> genre
genre2id = {}
genre_edges = []
for item_id, tags in artist_tags.items():
    for tag in tags:
        if tag not in genre2id:
            genre2id[tag] = len(genre2id) + num_users + num_items
            gid = genre2id[tag]
            genre_edges.append([item_id + num_users, gid])

```

```

genre_edges.append([gid, item_id + num_users])

all_edges = edge_index + genre_edges
tensor_edge_index = torch.tensor(all_edges,
dtype=torch.long).t().contiguous()
graph_data = Data(edge_index=tensor_edge_index, num_nodes=num_users +
num_items + len(genre2id))

#GNN-модель (GraphSAGE)
class RecGNN(torch.nn.Module):
    def __init__(self, num_nodes, emb_dim=64):
        super().__init__()
        self.emb = torch.nn.Embedding(num_nodes, emb_dim)
        self.conv1 = SAGEConv(emb_dim, emb_dim)
        self.conv2 = SAGEConv(emb_dim, emb_dim)

    def forward(self, edge_index):
        x = self.emb.weight
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return x

#BPR-тренинг для LightGCN/RecGNN
import torch
import torch.nn.functional as F
import numpy as np
from collections import defaultdict

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Device:', device)

user_pos_items =
listens.groupby('user_id')['artist_id'].apply(set).to_dict()
all_items = torch.arange(num_items, device=device)

# Для швидкого доступу зробимо списки позитивів (тензори різної довжини
зберігаємо як Python-списки)
user_pos_list = {}
for u, s in user_pos_items.items():
    if len(s) > 0:
        user_pos_list[u] = torch.tensor(list(s), device=device)

```

```

# 2) Модель і граф на GPU
model = RecGNN(graph_data.num_nodes).to(device)
edge_index = graph_data.edge_index.to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
scaler = torch.cuda.amp.GradScaler(enabled=(device.type == 'cuda'))

BATCH_USERS = 4096          # збільште batch користувачів (часто 4k-32k
ок)
NEG_RESAMPLE_TRIES = 3      # скільки разів перевибирати негатив, якщо
влучили в позитив

users_all = np.array(list(user_pos_items.keys()))

def sample_batch_users(users_np, start, bs):
    return torch.tensor(users_np[start:start+bs],      device=device,
dtype=torch.long)

def sample_pos_for_users(batch_users):
    # вибираємо випадковий позитивний айтем для кожного користувача
    pos_idx = []
    mask_keep = []
    for u in batch_users.tolist():
        if u in user_pos_list:
            pos_choices = user_pos_list[u]
            # один випадковий позитив
            j = torch.randint(high=pos_choices.numel(), size=(1,),
device=device)
            pos_idx.append(pos_choices[j])
            mask_keep.append(True)
        else:
            # користувач без позитивів – пропустимо
            pos_idx.append(torch.tensor([-1], device=device))
            mask_keep.append(False)
    pos_idx = torch.cat(pos_idx).long()
    mask_keep = torch.tensor(mask_keep,      device=device,
dtype=torch.bool)
    return pos_idx, mask_keep

def sample_neg_for_users(batch_users, forbid_sets):
    # початкове вибіркве негативів
    neg = torch.randint(high=num_items, size=(batch_users.numel(),),
device=device)

```

```

# відфільтровуємо попадання у позитиви (робимо кілька перевібірок)
for _ in range(NEG_RESAMPLE_TRIES):
    bad_mask = torch.zeros_like(neg, dtype=torch.bool)
    # позначаємо "погані" (які в позитиві)
    for i, u in enumerate(batch_users.tolist()):
        if u in forbid_sets and neg[i].item() in forbid_sets[u]:
            bad_mask[i] = True
    if not bad_mask.any():
        break
    # перевибираємо лише погані
    neg[bad_mask] = torch.randint(high=num_items,
size=(bad_mask.sum().item(),), device=device)
    return neg

for epoch in range(1, 6):
    np.random.shuffle(users_all)
    model.train()
    total_loss = 0.0

    # обчислюємо з один на batch користувачів (перед backward).
    # Це потрібно перераховувати перед КОЖНИМ optimizer.step(), але не
    для кожного користувача.
    for start in range(0, len(users_all), BATCH_USERS):
        batch_users = sample_batch_users(users_all, start, BATCH_USERS)

        # відкидаємо користувачів без позитивів
        pos_items, keep_mask = sample_pos_for_users(batch_users)
        if keep_mask.sum() == 0:
            continue
        batch_users = batch_users[keep_mask]
        pos_items = pos_items[keep_mask]

        # негативи з resample
        neg_items = sample_neg_for_users(batch_users, user_pos_items)

        # зсув до вузлів графа для item-вузлів (як у вас: item_node =
item_id + num_users)
        user_nodes = batch_users # user_id ==
node_id
        pos_nodes = pos_items + num_users
        neg_nodes = neg_items + num_users

        optimizer.zero_grad(set_to_none=True)

```

```

# Векторизація обчислень і AMP
with torch.cuda.amp.autocast(enabled=(device.type == 'cuda')):
    z = model(edge_index) # повний forward
по графу один раз на batch
    u_e = z[user_nodes]
    i_e = z[pos_nodes]
    j_e = z[neg_nodes]

    pos_score = (u_e * i_e).sum(dim=1)
    neg_score = (u_e * j_e).sum(dim=1)
    loss = -F.logsigmoid(pos_score - neg_score).mean()

    scaler.scale(loss).backward()
    scaler.step(optimizer)
    scaler.update()

    total_loss += loss.detach().item()

print(f"Епоха {epoch}: loss = {total_loss:.4f}")

# Рекомендації та схожі треки
z = model(graph_data.edge_index).detach()
user_embs = z[:num_users]
item_embs = z[num_users:num_users + num_items]

def recommend(u, topk=10):
    scores = torch.matmul(item_embs, user_embs[u])
    known_items = user_pos_items.get(u, set())
    scores[list(known_items)] = -1e9
    return torch.topk(scores, topk).indices

def similar_items(item_id, topk=5):
    emb = item_embs.numpy()
    sims = cosine_similarity([emb[item_id]], emb)[0]
    sims[item_id] = -1
    return np.argsort(sims)[-topk:][::-1]

# Візуалізація графа треків і жанрів
G = nx.Graph()
G.add_edges_from([(int(i), int(j)) for i,j in genre_edges])
pos = nx.spring_layout(G, seed=42)

```

```

plt.figure(figsize=(8,6))
nx.draw_networkx(G, pos, with_labels=False, node_size=20, alpha=0.6)
plt.title("Фрагмент графа: треки ↔ жанри")
plt.show()

#PCA-візуалізація ембедінгів
pca = PCA(n_components=2)
emb2d = pca.fit_transform(item_embs.numpy())
plt.figure(figsize=(8,6))
plt.scatter(emb2d[:,0], emb2d[:,1], alpha=0.4, s=10)
plt.title("Проекція трек-ембедінгів (PCA)")
plt.grid(True)
plt.show()

from sklearn.metrics.pairwise import cosine_similarity

def similar_items(item_id, topk=5):
    emb = item_embs.numpy()
    sims = cosine_similarity([emb[item_id]], emb)[0]
    sims[item_id] = -1 # не враховуємо сам трек
    return np.argsort(sims)[-topk:][::-1]

# Приклади
print("Топ-10 для юзера 0:", recommend(0).tolist())
print("Схожі на трек 0:", similar_items(0).tolist())

# Метрики: Recall@10, Precision@10, NDCG@10
from sklearn.metrics import ndcg_score

def evaluate_metrics(topk=10):
    recalls, precisions, ndcgs = [], [], []
    for u in list(user_pos_items.keys())[:100]:
        recs = recommend(u, topk=topk).numpy()
        actual = user_pos_items[u]
        hits = len(set(recs) & actual)
        recall = hits / len(actual) if actual else 0
        precision = hits / topk

        relevance = np.zeros(num_items)
        for i in actual:
            relevance[i] = 1
        scores = torch.matmul(item_embs, user_embs[u]).detach().numpy()

```

```

    ndcg = ndcg_score([relevance], [scores])

    recalls.append(recall)
    precisions.append(precision)
    ndcgs.append(ndcg)

    print(f"Recall@{topk}: {np.mean(recalls):.4f}")
    print(f"Precision@{topk}: {np.mean(precisions):.4f}")
    print(f"NDCG@{topk}: {np.mean(ndcgs):.4f}")

evaluate_metrics(topk=10)

#Візуалізація ембедінгів жанрів
genre_ids = list(genre2id.values())
genre_embs = z[genre_ids].numpy()
pca = PCA(n_components=2)
emb2d = pca.fit_transform(genre_embs)
plt.figure(figsize=(8,6))
plt.scatter(emb2d[:,0], emb2d[:,1], alpha=0.6, s=20)
plt.title("Жанрові ембедінги (PCA)")
plt.grid(True)
plt.show()

#Візуалізація персонального підграфа
def draw_user_graph(u_id):
    u_emb = z[u_id]
    neighbors = list(user_pos_items[u_id])[:10]
    G = nx.Graph()
    G.add_node(f'user_{u_id}')
    for n in neighbors:
        nid = f'item_{n}'
        G.add_node(nid)
        G.add_edge(f'user_{u_id}', nid)
    plt.figure(figsize=(6,4))
    nx.draw(G, with_labels=True, node_size=500, node_color='skyblue',
font_size=10)
    plt.title(f"Підграф прослуховувань користувача {u_id}")
    plt.show()

draw_user_graph(0)

```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ



РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ МЕТОДАМИ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ

Доповідає: Бичков Денис
Студент групи КА-41мп

Київ 2025

Актуальність

Зростання обсягів музичних даних → потреба в персоналізації

Класичні методи (CF, CBF) мають обмеження:

- розрідженість даних;
- холодний старт;
- слабе врахування структури взаємодій.

Графові нейронні мережі (GNN) дозволяють моделювати складні зв'язки між користувачами, артистами та жанрами.

Постановка задачі

Мета роботи: створити **рекомендаційну систему**, яка використовує структуру графа та модель GraphSAGE для формування персоналізованих рекомендацій музичних виконавців.

Дано:

- користувачі U
- артисти I
- жанри P
- імпліцитні взаємодії E
- гетерогенний граф $G(U \cup I \cup P, E)$

Потрібно: для кожного користувача сформувати топ- K релевантних виконавців.

Критерії:

- мінімізація BPR-loss
- максимізація Recall@ K , NDCG@ K , Precision@ K

Тип задачі: implicit-feedback ranking.

3

Датасет

Last.fm 360K

- 360 000 користувачів
- імпліцитні оцінки (кількість прослуховувань)
- жанрові теги отримано через API Last.fm

Після фільтрації:

- ~1400 користувачів
- ~10 000 виконавців
- ~90 000 взаємодій

4

Підготовка даних

- очищення та фільтри активності
- індексація user_id і artist_id
- побудова біпартитного графа «користувач–виконавець»
- додано вузли жанрів та ребра «виконавець–жанр»
- формування Leave-One-Out тесту

5

Структура графа

Гетерогенний граф включає:

- вузли: користувачі, виконавці, жанри;
- ребра:
 - user $\rightarrow$ artist;
 - artist $\rightarrow$ genre;
 - item $\leftrightarrow$ genre.

Перевага: граф зберігає багатовимірні зв'язки та семантику жанрів.



6

Модель GraphSAGE

- індуктивна GNN-модель;
- агрегація інформації лише від сусідів;
- ефективна для великих графів;
- використано 2 шари SAGEConv;
- ембедінги $\rightarrow$ рекомендації через ранжування.

```
class RecGNN(torch.nn.Module):
    def __init__(self, num_nodes, emb_dim=64):
        super().__init__()
        self.emb = torch.nn.Embedding(num_nodes, emb_dim)
        self.conv1 = SAGEConv(emb_dim, emb_dim)
        self.conv2 = SAGEConv(emb_dim, emb_dim)

    def forward(self, edge_index):
        x = self.emb.weight
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return x
```

7

Модель GraphSAGE

1. AGGREGATE_k: обчислення проміжного вектора $m_u^{(k)}$ для вузла u шляхом агрегування векторів його сусідів з попереднього шару $\mathbf{h}_v^{(k-1)} : v \in \mathcal{N}(u)$ (де $\mathcal{N}(u)$ – множина сусідів u).
2. COMBINE_k: об'єднання отриманого агрегату $m_u^{(k)}$ із поточним представленням вузла $\mathbf{h}_u^{(k-1)}$ для обчислення нового представлення $\mathbf{h}_u^{(k)}$.

На нульовому шарі ($k = 0$) беруться вихідні ознаки вузлів: $\mathbf{h}_u^{(0)} = \mathbf{e}_u$. Далі ітеративно обчислюються агрегати та нові представлення для $k = 1, 2, \dots, K$.

8

Функція втрат BPR-loss

- підходить для **імпліцитних даних**;
- ідея: позитивний виконавець має отримати вищий скоринг, ніж негативний;
- використовує triplets (user, positive, negative);
- спрямована на оптимізацію ранжування, а не передбачення оцінок.

$$\mathcal{L}_{BPR} = - \sum_{u \in U} \sum_{(i,j) \in \Omega_u} \ln \sigma(\hat{y}(u,i) - \hat{y}(u,j)) + \lambda \|\Theta\|^2,$$

9

Навчання моделі

- mini-batch sampling для пришвидшення;
- 5 епох навчання;
- стійке зменшення BPR-loss;
- зростання Recall@10 та NDCG@1;
- формування якісних ембедінгів користувачів та артистів.

```
for epoch in range(1, 6):
    np.random.shuffle(users_all)
    model.train()
    total_loss = 0.0

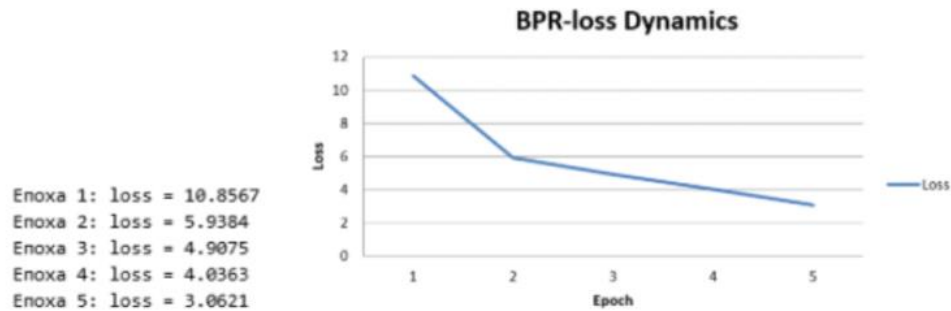
    # обчислимо з один на batch користувачів (перед backward).
    # це потрібно перерахувати перед КОЖНОГО optimizer.step(), але не для кожного користувача.
    for start in range(0, len(users_all), BATCH_USERS):
        batch_users = sample_batch_users(users_all, start, BATCH_USERS)

        # відкидаємо користувачів без позитивів
        pos_items, keep_mask = sample_pos_for_users(batch_users)
        if keep_mask.sum() == 0:
            continue
        batch_users = batch_users[keep_mask]
        pos_items = pos_items[keep_mask]

        # негативи з resample
        neg_items = sample_neg_for_users(batch_users, user_pos_items)
```

10

Навчання моделі



11

Приклад рекомендації

- Користувач слухає виконавців у жанрах рок та метал.
- Система рекомендує користувачу 10 виконавців, один з яких є представником жанру ню-метал (Deftones).
- Отже, 1 рекомендація з 10 є влучною.

user 42 liked:

1. Metallica
2. Slipknot
3. Rammstein
4. Disturbed
5. Korn
6. System of a Down
7. Three Days Grace
8. Papa Roach
9. Godsmack
10. Seether

Recommended:

1. Doja Cat
2. Eminem
3. Billie Eilish
4. Kanye West
5. Bad Bunny
6. Deftones
7. Taylor Swift
8. Drake
9. The Weeknd
10. Adele

12

Оцінка моделі

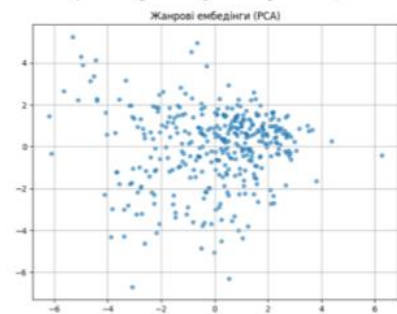
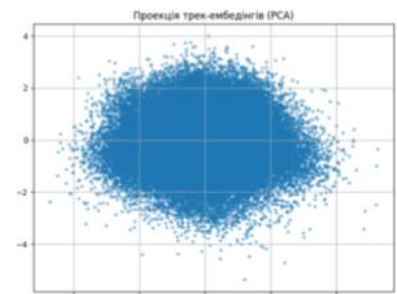
Результати:

- $\text{Recall@10} \approx 0.36\text{--}0.41$, $\text{NDCG@10} \approx 0.25 - 0.3$, $\text{Precision@10} \approx 0.04 - 0.06$;
- модель правильно «вгадує» релевантного виконавця у 40% топ-10 рекомендацій;
- ефективна робота з імпліцитними даними;
- жанрові зв'язки покращують якість ембедінгів.

13

Візуалізація ембедінгів

- PCA проекції виконавців → один великий кластер;
- причина: імпліцитні дані + слабка жанрова сегментація;
- PCA жанрових тегів показує близькі жанри;
- візуалізація підграфа користувача відображає його індивідуальні вподобання.



14

Переваги та потенціал моделі

Переваги:

- індуктивність;
- масштабованість;
- робота з імпліцитними даними;
- можливість збагачення новими вузлами.

Можливі покращення:

- session-based рекомендації;
- hard-negative sampling;
- мультимодальні ознаки (аудіо, тексти);
- LightGCN, PinSAGE.

15

Висновки

- Розроблено рекомендаційну систему на основі GraphSAGE.
- Побудовано гетерогенний граф «користувач–виконавець–жанр».
- Модель показала якісні результати ($\text{Recall@10} \approx 0.4$).
- Реалізовано візуалізації структури графа та ембедінгів.
- GNN — перспективний інструмент для музичних рекомендацій.

16