

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження та застосування генеративних моделей
в рекомендаційних системах
(тема)

Виконав:
здобувач другого року навчання,
групи СШМ-24-2

Євгеній ФАРАФОНОВ
(власне ім'я, прізвище)

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

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

Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)

Керівник проф. Наталія РЯБОВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук
Кафедра _____ Штучного інтелекту
Рівень вищої освіти _____ другий (магістерський)
Спеціальність _____ 122 Комп'ютерні науки
(код і повна назва)
Тип програми _____ освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма _____ Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____» _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Фарафонову Євгенію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та застосування генеративних моделей в рекомендаційних системах

затверджена наказом університету від 6 квітня 2026 р. № 269Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 22 травня 2026 р.

3. Вихідні дані до роботи локальний набір даних Amazon Appliances, науково-технічні публікації, Python documentation

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі

2) Постановка задачі

3) Теоретичне дослідження

4) Експериментальне дослідження

КАЛЕНДАРНИЙ ПЛАН

[illegible]

Дата видачі завдання 6 квітня 20 26 р.

Здобувач

(підпис)

Керівник роботи

(підпис)

проф. Наталія РЯБОВА

(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 72 с., 3 рис., 14 табл., 1 дод., 20 джерел.

ГЕНЕРАТИВНІ МОДЕЛІ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, СИНТЕТИЧНІ ДАНІ, AMAZON REVIEWS, PYTHON, TF-IDF.

Об'єкт дослідження – процес побудови та оцінювання рекомендаційних систем на основі історії взаємодій користувачів із товарами.

Предмет дослідження – методи використання великих мовних моделей для формування синтетичних сигналів у рекомендаційних системах.

Мета роботи – дослідити, чи можуть LLM-згенеровані дані покращити якість рекомендацій, та визначити найбільш ефективний спосіб їх застосування.

У роботі використано набір даних Amazon Appliances. Виконано очищення даних, об'єднання взаємодій із метаданими та k-core фільтрацію.

Досліджено два способи використання LLM: генерацію synthetic user-item interactions та побудову item-item semantic similarity. Встановлено, що пряме генерування user-item взаємодій не дає стабільного покращення, оскільки такі дані не відображають реальну поведінку користувачів.

Кращі результати отримано при використанні LLM як джерела item-item семантичної подібності та сигналу для recent-history reranking. Приріст NDCG@20 становив 4,34 % для Item-CF, 4,89 % для GRU4Rec, 2,25 % для LightGCN і 9,64 % для BPR-MF. Зроблено висновок, що LLM доцільніше використовувати для семантичного доповнення рекомендаційних моделей, а не для генерації штучної поведінки користувачів.

ABSTRACT

Master's thesis contains: 72 pp., 14 fig., 18 tabl., 2 app., 20 references.

AMAZON REVIEWS, BPR-MF, COLLABORATIVE FILTERING, GENERATIVE MODELS, GRU4REC, ITEM-CF, ITEM-ITEM SIMILARITY, LLM, LIGHTGCN, NDCG, PYTHON, RECOMMENDER SYSTEMS, SYNTHETIC DATA, TF-IDF.

The object of research is the process of building and evaluating recommender systems based on the history of user interactions with items.

The subject of research is methods for using large language models to generate synthetic signals in recommender systems.

The aim of the work is to investigate whether LLM-generated data can improve recommendation quality and to determine the most effective way to apply such data.

The Amazon Appliances dataset was used in the work. Data cleaning, merging of interactions with metadata, and k-core filtering were performed.

Two ways of using LLMs were studied: generation of synthetic user-item interactions and construction of item-item semantic similarity. It was found that direct generation of user-item interactions does not provide stable improvement, because such data do not reflect real user behavior.

The best results were obtained when the LLM was used as a source of item-item semantic similarity and as a signal for recent-history reranking. The NDCG@20 improvement was 4.34% for Item-CF, 4.89% for GRU4Rec, 2.25% for LightGCN, and 9.64% for BPR-MF. It was concluded that LLMs are more appropriate for semantic augmentation of recommender models than for generating artificial user behavior.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	8
Вступ.....	10
1 Аналіз предметної галузі	14
1.1 Аналіз проблемної галузі рекомендаційних систем.....	14
1.2 Основні підходи до побудови рекомендаційних систем	15
1.3 Застосування генеративних моделей у рекомендаційних системах ..	17
1.4 Аналіз обмежень та формулювання наряду дослідження.....	19
2 Постановка задачі.....	22
2.1 Формулювання проблеми дослідження.....	23
2.2 Мета, об'єкт і предмет дослідження	24
2.3 Основні завдання роботи.....	24
2.4 Вхідні та вихідні дані.....	25
2.5 Вимоги до експериментального дослідження.....	26
3 Теоретичне дослідження	28
3.1 Математична постановка задачі рекомендації.....	28
3.2 Метрики оцінювання рекомендаційних систем.....	29
3.3 Базові моделі рекомендацій	31
3.3.1 Most Popular	32
3.3.2 Item-based Collaborative Filtering	32
3.3.3 Matrix Factorization та BPR-MF	33
3.3.4 LightGCN та послідовні моделі	34
3.4 Генерація синтетичних user-item взаємодій за допомогою LLM.....	34
3.5 Генерація синтетичних item-item similarity за допомогою LLM.....	36
3.6 Гібридна модель	38
3.7 Архітектура програмного забезпечення	39
4 Експериментальне дослідження	48
4.1 Набір даних Amazon Appliances	48
4.2 Попередня обробка даних	50

4.3 Методика експериментального оцінювання	50
4.4 Реалізація рекомендаційних моделей	51
4.5 Експеримент із прямою генерацією synthetic user-item interactions ..	52
4.6 Retrieval-based та collaborative-aware synthetic interactions	53
4.7 LLM-generated item-item similarity	54
4.8 Hybrid Item-CF + LLM item similarity.....	56
4.9 Вплив розміру user subset	58
4.10 LLM recent-history reranking.....	59
4.11 Порівняльний аналіз результатів.....	63
Висновки	65
Перелік джерел посилання	69
Додаток А Відомість кваліфікаційної роботи	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс для взаємодії між програмними системами;

BPR – Bayesian Personalized Ranking – баєсівське персоналізоване ранжування;

BPR-MF – Bayesian Personalized Ranking Matrix Factorization – матрична факторизація з навчанням за критерієм попарного ранжування;

CF – Collaborative Filtering – колаборативна фільтрація;

EASE – Embarrassingly Shallow Autoencoders – лінійна item-based модель рекомендацій для розріджених даних;

GRU – Gated Recurrent Unit – керований рекурентний блок;

GRU4Rec – Gated Recurrent Unit for Recommendation – рекомендаційна модель на основі GRU;

IDF – Inverse Document Frequency – обернена документна частота;

Item-CF – Item-based Collaborative Filtering – колаборативна фільтрація на основі подібності між об'єктами;

Item-item similarity – подібність між об'єктами рекомендації;

K-core filtering – ітеративна фільтрація користувачів та об'єктів за мінімальною кількістю взаємодій;

LightGCN – Light Graph Convolutional Network – спрощена графова згорткова мережа;

LLM – Large Language Model – велика мовна модель;

MF – Matrix Factorization – матрична факторизація;

NDCG@K – Normalized Discounted Cumulative Gain at K – нормалізований дисконтований кумулятивний виграш на перших K позиціях;

NLP – Natural Language Processing – обробка природної мови;

OpenRouter – сервіс API-доступу до великих мовних моделей;

Precision@K – точність на перших K позиціях рекомендаційного списку;

Recall@K – повнота на перших K позиціях рекомендаційного списку;

Recommender system – рекомендаційна система;

SLIM – Sparse Linear Methods – розріджені лінійні методи для top-N рекомендацій;

TF-IDF – Term Frequency-Inverse Document Frequency – частота терміна з урахуванням оберненої документної частоти.

ВСТУП

У сучасних інформаційних системах користувачі щоденно взаємодіють із великою кількістю цифрового контенту, товарів, сервісів та інформаційних об'єктів. У таких умовах виникає потреба в автоматизованих методах персоналізації, які дозволяють скоротити інформаційне перевантаження та надати користувачу найбільш релевантні рекомендації. Рекомендаційні системи стали невід'ємною частиною електронної комерції, стримінгових платформ, соціальних мереж, онлайн-освіти та інших цифрових сервісів.

Основою більшості рекомендаційних систем є аналіз історії взаємодій користувачів з об'єктами. Такими взаємодіями можуть бути покупки, перегляди, оцінки, відгуки або кліки. На їх основі система намагається виявити приховані закономірності у поведінці користувачів і сформувати список об'єктів, які з найбільшою ймовірністю будуть для них корисними. Класичні підходи, зокрема колаборативна фільтрація, матрична факторизація та item-based методи, довели свою ефективність у багатьох прикладних задачах [1].

Водночас рекомендаційні системи мають низку суттєвих обмежень. Однією з основних проблем є розрідженість даних: у реальних системах користувач взаємодіє лише з невеликою частиною доступних об'єктів. Унаслідок цього user-item матриця містить дуже мало заповнених значень, що ускладнює точне моделювання інтересів користувачів. Також актуальними залишаються проблеми холодного старту, надмірної орієнтації на популярні об'єкти та недостатнього використання семантичної інформації про товари або контент.

Останніми роками значний розвиток отримали генеративні моделі, зокрема великі мовні моделі. Вони здатні аналізувати природномовні описи, узагальнювати текстову інформацію, знаходити семантичні зв'язки між об'єктами та генерувати нові дані. Це відкриває перспективи їх

застосування у рекомендаційних системах, особливо в задачах, де доступні текстові метадані товарів, відгуки користувачів або описи об'єктів [15].

Одним із можливих напрямів використання генеративних моделей є створення синтетичних даних для покращення рекомендаційної якості. На перший погляд, великі мовні моделі можуть бути використані для генерації додаткових user-item взаємодій, тобто штучного розширення навчальної вибірки. Проте така ідея потребує ретельної експериментальної перевірки, оскільки поведінка користувачів у рекомендаційних системах залежить не лише від семантичної відповідності товару інтересам користувача, а й від багатьох зовнішніх факторів: доступності товару, ціни, часу, контексту покупки, експозиції об'єкта та індивідуальних обмежень.

Тому актуальність теми дослідження полягає в необхідності визначити, який саме тип синтетичного сигналу, створеного генеративною моделлю, є корисним для рекомендаційних систем. Важливо не лише перевірити можливість генерації додаткових даних, а й оцінити, чи не спотворюють такі дані реальну структуру користувацьких взаємодій. Особливої уваги заслуговує альтернативний підхід, за якого генеративна модель використовується не для створення штучної поведінки користувачів, а для побудови семантичних зв'язків між об'єктами.

Метою роботи є дослідження та застосування генеративних моделей у рекомендаційних системах, а також експериментальне визначення доцільності використання LLM-згенерованих синтетичних сигналів для покращення якості рекомендацій.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної галузі рекомендаційних систем та визначити основні проблеми, пов'язані з розрідженістю даних і недостатнім використанням семантичної інформації;
- розглянути основні класичні та сучасні підходи до побудови рекомендаційних систем, зокрема Most Popular, Item-based Collaborative Filtering, матричну факторизацію, BPR-MF, EASE та графові моделі;

- дослідити можливі способи застосування генеративних мовних моделей у рекомендаційних системах;
- підготувати набір даних Amazon Appliances для проведення експериментів, включаючи очищення даних, об'єднання взаємодій з метаданими товарів та k-core фільтрацію;
- реалізувати базові рекомендаційні моделі для подальшого порівняння результатів;
- розробити підходи до генерації синтетичних user-item взаємодій за допомогою великої мовної моделі;
- розробити підхід до генерації item-item семантичної подібності на основі текстових метаданих товарів;
- реалізувати гібридну рекомендаційну модель, яка поєднує колаборативну item-item подібність із LLM-згенерованою семантичною подібністю;
- провести експериментальне оцінювання моделей за стандартними метриками top-K recommendation;
- проаналізувати отримані результати та визначити, який тип LLM-згенерованого сигналу є найбільш доцільним для застосування в рекомендаційних системах.

Об'єктом дослідження є процес побудови та оцінювання рекомендаційних систем на основі взаємодій користувачів з об'єктами.

Предметом дослідження є методи застосування генеративних мовних моделей для створення синтетичних сигналів у рекомендаційних системах, зокрема синтетичних user-item взаємодій та item-item семантичної подібності.

Методи дослідження базуються на використанні алгоритмів колаборативної фільтрації, item-based recommendation, матричної факторизації, TF-IDF пошуку, генерації текстових відповідей за допомогою великих мовних моделей та експериментального оцінювання рекомендацій за метриками Precision@K, Recall@K, HitRate@K і NDCG@K. Програмна

реалізація виконана мовою Python з використанням бібліотек pandas, NumPy, SciPy, scikit-learn та інструментів для взаємодії з LLM через API.

Наукова новизна роботи полягає в експериментальному порівнянні двох різних способів використання генеративних моделей у рекомендаційних системах: генерації синтетичних user-item взаємодій та генерації item-item семантичної подібності. На відміну від підходів, які розглядають LLM переважно як безпосередній генератор рекомендацій або пояснень, у цій роботі досліджується їх застосування як джерела допоміжного синтетичного сигналу для класичних рекомендаційних алгоритмів.

Практичне значення отриманих результатів полягає у розробці експериментального програмного пайплайну для підготовки даних, генерації LLM-сигналів та оцінювання рекомендаційних моделей. Запропонований підхід може бути використаний у системах електронної комерції та інших предметних галузях, де доступні текстові метадані об'єктів, але спостерігається нестача колаборативних взаємодій. Крім того, результати роботи дозволяють обґрунтовано визначити, у яких випадках генеративні моделі можуть бути корисними для рекомендаційних систем, а в яких їх застосування може призводити до спотворення реальної поведінкової структури даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз проблемної галузі рекомендаційних систем

Рекомендаційні системи є класом інформаційних систем, призначених для автоматичного добору релевантних об'єктів для користувача на основі його попередньої поведінки, характеристик об'єктів або поведінки інших користувачів. Такі системи широко застосовуються в електронній комерції, стримінгових сервісах, соціальних мережах, цифрових бібліотеках, новинних платформах та інших інформаційних середовищах, де користувач стикається з великою кількістю потенційно доступних об'єктів [1].

Основною задачею рекомендаційної системи є формування для кожного користувача списку об'єктів, які мають найбільшу ймовірність бути для нього корисними або цікавими. У типовій постановці користувачі утворюють множину U , об'єкти – множину I , а історія взаємодій задається матрицею R , де кожен елемент відображає взаємодію користувача u з об'єктом i . Такою взаємодією може бути оцінка, покупка, перегляд, клік, додавання в кошик або текстовий відгук. У задачах top-K recommendation модель повинна не просто передбачити числову оцінку, а сформувати ранжований список із K найбільш релевантних об'єктів.

Ключовою проблемою рекомендаційних систем є розрідженість user-item матриці. У реальних системах кількість можливих пар «користувач–об'єкт» є дуже великою, але фактичні взаємодії становлять лише малу частину від усіх можливих комбінацій. Наприклад, користувач інтернет-магазину взаємодіє лише з кількома товарами з великого каталогу. Унаслідок цього модель має робити висновки про вподобання користувача на основі неповної та нерівномірної інформації.

Розрідженість даних безпосередньо пов'язана з проблемою холодного старту. Для нових користувачів система має недостатньо інформації про їхні інтереси, а для нових або малопопулярних товарів – недостатньо взаємодій,

щоб оцінити їхню релевантність. Це призводить до зміщення рекомендацій у бік популярних об'єктів, тобто до popularity bias. У такій ситуації система часто рекомендує товари, які вже мають велику кількість взаємодій, і погано просуває об'єкти з довгого хвоста каталогу.

Класичні колаборативні методи використовують лише історію взаємодій. Їхньою перевагою є здатність виявляти неочевидні поведінкові закономірності, проте вони погано працюють там, де взаємодій мало. Контентні методи, навпаки, використовують ознаки самих об'єктів: назви, категорії, описи, характеристики, бренди або тексти відгуків. Вони можуть бути корисними в умовах холодного старту, але не завжди точно відображають реальні поведінкові вподобання користувачів.

Саме на перетині цих двох підходів виникає актуальна дослідницька проблема: як поєднати колаборативну інформацію про поведінку користувачів із семантичною інформацією про об'єкти. Особливо важливим це стає для наборів даних електронної комерції, де товари мають багаті текстові метадані, але історія взаємодій є неповною. У таких умовах генеративні мовні моделі можуть бути використані як інструмент для побудови додаткового семантичного сигналу.

1.2 Основні підходи до побудови рекомендаційних систем

Найпростішим підходом до рекомендацій є модель Most Popular. Вона рекомендує всім користувачам однаковий список найпопулярніших об'єктів. Популярність може визначатися кількістю переглядів, покупок, оцінок або середнім рейтингом. Такий підхід не є персоналізованим, однак часто використовується як базова лінія порівняння. Якщо складніша модель не перевищує Most Popular за якістю, її практична цінність є сумнівною.

Одним із найпоширеніших класичних підходів є колаборативна фільтрація. Вона базується на припущенні, що користувачі з подібною історією взаємодій можуть мати подібні майбутні вподобання, а об'єкти, з

якими взаємодіють схожі групи користувачів, також є подібними. У user-based collaborative filtering рекомендації будуються на основі схожості користувачів, тоді як в item-based collaborative filtering основою є схожість між об'єктами. Item-based collaborative filtering було формалізовано в роботі Sarwar та співавт., де показано, що item-based підхід може бути ефективним і масштабованим для практичних рекомендаційних систем [2].

Item-CF є особливо релевантним для цієї роботи, оскільки він безпосередньо оперує item-item similarity. У класичному варіанті подібність між товарами обчислюється на основі спільних користувачів, які взаємодіяли з цими товарами. Наприклад, якщо значна кількість користувачів взаємодіяла з двома товарами, ці товари вважаються близькими у колаборативному просторі. Перевагою такого підходу є простота інтерпретації та відносна стабільність item-item зв'язків порівняно з user-user зв'язками. Недоліком є залежність від наявності достатньої кількості спільних взаємодій.

Іншим важливим класом методів є матрична факторизація. Вона представляє користувачів і об'єкти у вигляді латентних векторів меншої розмірності. Ідея полягає в тому, що оцінка або ймовірність взаємодії між користувачем і товаром може бути апроксимована скалярним добутком їхніх латентних представлень. Методи матричної факторизації стали одним із ключових напрямів розвитку рекомендаційних систем після успіху в Netflix Prize [3].

Для задач top-N recommendation важливо не тільки передбачати числові рейтинги, а й правильно ранжувати релевантні об'єкти вище за нерелевантні. Саме для цього було запропоновано Bayesian Personalized Ranking. BPR оптимізує попарне ранжування: для кожного користувача модель навчається ставити відомий позитивний об'єкт вище за невідомий або негативний [4]. Це робить BPR-MF більш придатною моделлю для top-K рекомендацій, ніж класична матрична факторизація, оптимізована під передбачення рейтингу.

До сильних item-based моделей належить EASE. Ця модель будує лінійне відображення між об'єктами і фактично навчає матрицю item-item ваг для реконструкції user-item матриці. Незважаючи на просту архітектуру, EASE часто показує високу якість на розріджених рекомендаційних наборах даних [6]. Для цієї роботи EASE є важливою як сильніша item-based baseline-модель, з якою доцільно порівнювати Item-CF та його LLM-розширення.

Сучасним напрямом є графові рекомендаційні моделі. User-item взаємодії природно можна подати у вигляді двочасткового графа, де вузлами є користувачі та товари, а ребрами – взаємодії. LightGCN спрощує графові згорткові мережі для рекомендацій, залишаючи лише поширення embeddings по графу без зайвих нелінійних перетворень [8]. Такий підхід є корисним для моделювання багатокрокових зв'язків між користувачами та товарами, але потребує ретельнішого налаштування і більших обчислювальних ресурсів.

Окремо можна виділити послідовні рекомендаційні моделі, наприклад GRU4Rec. Вони моделюють порядок взаємодій користувача і намагаються передбачити наступний об'єкт у послідовності [9]. Такі моделі добре підходять для session-based recommendation, але їхня ефективність залежить від довжини історії користувачів. Якщо в наборі даних багато користувачів з двома-трьома взаємодіями, послідовні моделі можуть не мати достатньо інформації для повноцінного навчання.

Таким чином, рекомендаційні системи охоплюють широкий спектр підходів: від простих popularity-based моделей до графових і нейронних архітектур. Проте для даної роботи центральним є клас item-based моделей, оскільки саме вони дозволяють природно інтегрувати додатковий item-item семантичний сигнал, отриманий за допомогою генеративної мовної моделі.

1.3 Застосування генеративних моделей у рекомендаційних системах

Великі мовні моделі є генеративними моделями, здатними аналізувати, інтерпретувати та створювати текст природною мовою. Їх

розвиток істотно вплинув на задачі обробки тексту, пошуку, діалогових систем і персоналізації. У контексті рекомендаційних систем LLM можуть використовуватись не лише для прямої генерації рекомендацій, а й як допоміжний інструмент для роботи з текстовими ознаками, відгуками, описами товарів та профілями користувачів [15].

Існує кілька основних способів використання LLM у рекомендаційних системах. Перший підхід – застосування LLM як безпосереднього recommender agent, який отримує опис користувача та набір кандидатів і генерує рекомендацію у текстовій формі. Такий підхід є гнучким, але складним для масштабування і стабільного оцінювання в класичних top-K метриках. Попередні дослідження можливостей ChatGPT як рекомендаційної системи показують, що LLM можуть демонструвати корисну семантичну обізнаність, але їхня якість залежить від формату prompt, доступного контексту та способу подання кандидатів [17].

Другий підхід – використання LLM для генерації пояснень рекомендацій. У цьому випадку основна рекомендаційна модель може залишатися класичною, а LLM застосовується для формування зрозумілого користувачу текстового пояснення: чому саме цей товар або об'єкт був рекомендований [20].

Третій підхід – застосування LLM для обробки текстових метаданих. У багатьох доменах об'єкти мають назви, категорії, описи, технічні характеристики, відгуки та інші текстові поля. LLM може узагальнювати ці дані, виділяти ключові ознаки товарів, знаходити семантичні зв'язки між об'єктами або формувати додаткові representations. Це особливо корисно для товарів з малою кількістю взаємодій, де колаборативний сигнал є слабким.

Четвертий підхід – генерація синтетичних даних. У рекомендаційних системах синтетичними можуть бути профілі користувачів, текстові відгуки, user-item взаємодії або item-item зв'язки. Ідея полягає в тому, що LLM може заповнити прогалини в даних, створюючи додаткову

інформацію, яка потім використовується класичною рекомендаційною моделлю. Проте цей напрям потребує особливої обережності, оскільки синтетичні дані можуть бути семантично правдоподібними, але статистично або поведінково некоректними.

Окремий інтерес становлять підходи, у яких LLM комбінується з графовими або колаборативними структурами. Наприклад, LLMRec використовує великі мовні моделі разом із графовим доповненням для рекомендацій [18]. Це демонструє загальну тенденцію: LLM не обов'язково має замінювати рекомендаційну модель, а може виступати додатковим джерелом інформації, яке посилює вже наявні collaborative або graph-based сигнали.

1.4 Аналіз обмежень та формулювання напрямку дослідження

Аналіз предметної галузі показує, що класичні рекомендаційні системи мають сильну залежність від якості та повноти даних про взаємодії користувачів з об'єктами. Якщо таких взаємодій достатньо, колаборативні моделі можуть ефективно виявляти поведінкові закономірності. Проте в умовах розріджених даних, коротких історій користувачів або довгого хвоста товарів їхня якість знижується. Саме тому актуальним є пошук додаткових джерел сигналу.

Текстові метадані товарів є одним із таких джерел. У наборах даних електронної комерції товари часто мають назву, категорію, опис, характеристики, бренд, середній рейтинг та текстові відгуки. Такі дані містять інформацію, яка не завжди проявляється у user-item матриці. Наприклад, два товари можуть бути функціонально близькими, але не мати достатньо спільних користувачів через низьку популярність одного з них. Колаборативна модель у такому випадку не виявить сильного зв'язку, тоді як семантична модель може визначити близькість за описом і характеристиками.

Водночас просте додавання синтетичних user-item interactions є методично ризикованим. Якщо рекомендаційна модель навчається на штучних взаємодіях, вона починає сприймати їх як реальні прояви поведінки користувачів. У випадку помилки це змінює структуру co-occurrence, впливає на item neighborhoods і може погіршити якість рекомендацій. Проблема полягає не лише в окремих неправильних записах, а в тому, що synthetic interactions можуть систематично зміщувати модель у бік популярних або семантично очевидних товарів.

У цьому контексті важливо відрізнити семантичну правдоподібність від поведінкової достовірності. LLM може правильно визначити, що певний аксесуар підходить до певного побутового приладу, або що два товари належать до близької функціональної категорії. Але з цього не випливає, що конкретний користувач справді купив би цей товар або поставив би йому високий рейтинг. Для рекомендаційної системи така різниця є принциповою.

Тому напрям дослідження цієї роботи полягає в експериментальному порівнянні двох стратегій використання генеративних моделей. Перша стратегія – генерація синтетичних user-item взаємодій. Вона перевіряє гіпотезу, що LLM може безпосередньо доповнювати навчальну матрицю взаємодій. Друга стратегія – генерація item-item семантичної подібності. Вона перевіряє гіпотезу, що LLM краще використовувати для побудови допоміжного семантичного графа товарів без зміни реальних історій користувачів.

Для реалізації такого дослідження доцільно обрати домен, де наявні як реальні взаємодії користувачів, так і достатньо багаті текстові метадані. Таким доменом є Amazon Reviews, який широко використовується в дослідженнях рекомендаційних систем та містить оцінки, відгуки й описи товарів [14]. У межах цієї роботи використовується локальний Amazon Appliances dataset, що дозволяє дослідити задачу в предметній області побутової техніки та аксесуарів.

Сформульований напрям дослідження дозволяє не лише перевірити, чи можуть LLM покращити рекомендаційні системи, а й уточнити, в якій саме ролі вони є найбільш корисними. Це важливо з практичної точки зору, оскільки безпосереднє використання LLM як генератора поведінки може бути дорогим, нестабільним і важко контрольованим. Натомість використання LLM для побудови item-item semantic augmentation може бути простішим для інтеграції в існуючі recommendation pipelines.

2 ПОСТАНОВКА ЗАДАЧІ

Після аналізу предметної галузі було встановлено, що рекомендаційні системи значною мірою залежать від повноти та якості даних про взаємодії користувачів з об'єктами. У практичних системах такі дані зазвичай є розрідженими: більшість користувачів взаємодіє лише з малою частиною доступного каталогу, а значна кількість товарів має недостатню кількість оцінок або покупок. Це обмежує ефективність класичних колаборативних методів, зокрема Item-based Collaborative Filtering та матричної факторизації.

Одним із перспективних напрямів подолання цієї проблеми є використання додаткової інформації про об'єкти. У домені електронної комерції такою інформацією можуть бути назви товарів, категорії, текстові описи, характеристики, бренди, середні рейтинги та відгуки користувачів. Ці дані містять семантичну інформацію, яка не завжди відображається у матриці user-item взаємодій. Наприклад, два товари можуть бути функціонально схожими або взаємодоповнюваними, але не мати достатньо спільних користувачів для того, щоб класична колаборативна модель виявила між ними сильний зв'язок.

У зв'язку з розвитком великих мовних моделей виникає можливість використати їх для створення додаткового синтетичного сигналу в рекомендаційній системі. Однак ключова проблема полягає в тому, що не будь-який LLM-згенерований сигнал є однаково корисним. Синтетичні user-item взаємодії можуть виглядати правдоподібно з погляду текстового опису користувача або товару, але при цьому не відповідати реальній поведінці користувачів. Рекомендаційні моделі навчаються не лише на семантичній близькості, а й на статистичних закономірностях поведінки, тому штучне додавання взаємодій може спотворити колаборативну структуру даних.

2.1 Формулювання проблеми дослідження

У класичній рекомендаційній задачі задано множину користувачів U , множину об'єктів I та множину спостережуваних взаємодій R . Кожна взаємодія може містити ідентифікатор користувача, ідентифікатор товару, оцінку, час взаємодії та додаткову інформацію, наприклад текст відгуку. Метою рекомендаційної системи є побудова для кожного користувача ранжованого списку об'єктів, які з найбільшою ймовірністю будуть для нього релевантними.

У цій роботі розглядається задача *top-K recommendation*. Для кожного користувача необхідно сформувати список із K товарів, де релевантний тестовий товар повинен займати якомога вищу позицію. Така постановка відповідає практичним сценаріям, у яких користувачу показується обмежена кількість рекомендацій, наприклад 5, 10 або 20 товарів.

Основна дослідницька проблема формулюється так: необхідно визначити, чи можуть LLM-згенеровані синтетичні сигнали покращити якість рекомендаційної системи, і який саме тип такого сигналу є найбільш корисним для задачі *top-K recommendation*.

У межах роботи перевіряються два підходи:

- генерація *synthetic user-item interactions*, тобто створення штучних записів про можливі взаємодії користувачів із товарами;
- генерація *LLM-based item-item similarity*, тобто створення семантичних зв'язків між товарами на основі їхніх текстових метаданих.

Перший підхід безпосередньо змінює навчальну матрицю взаємодій, додаючи до неї синтетичні записи. Другий підхід не змінює реальні історії користувачів, а лише доповнює *item-item similarity* додатковою семантичною інформацією. Саме ця різниця є принциповою для подальшого дослідження.

2.2 Мета, об'єкт і предмет дослідження

Метою роботи є дослідження та застосування генеративних моделей у рекомендаційних системах, а також експериментальне визначення доцільності використання LLM-згенерованих синтетичних сигналів для покращення якості рекомендацій.

Об'єктом дослідження є процес побудови та оцінювання рекомендаційних систем на основі взаємодій користувачів з об'єктами.

Предметом дослідження є методи застосування генеративних мовних моделей для створення синтетичних сигналів у рекомендаційних системах, зокрема синтетичних user-item взаємодій та item-item семантичної подібності.

У межах роботи досліджується не автономна рекомендаційна здатність LLM, а їх використання як допоміжного компонента для класичних рекомендаційних алгоритмів. Такий підхід дозволяє оцінити, чи може генеративна модель покращити вже існуючу рекомендаційну модель без повної заміни рекомендаційного пайплайну.

2.3 Основні завдання роботи

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної галузі рекомендаційних систем та визначити ключові проблеми, пов'язані з розрідженістю user-item матриці;
- розглянути класичні та сучасні підходи до рекомендацій, зокрема Most Popular, Item-based Collaborative Filtering, Matrix Factorization, BPR-MF, EASE та графові моделі;
- дослідити можливі способи застосування великих мовних моделей у рекомендаційних системах;
- підготувати локальний набір даних Amazon Appliances, що містить реальні user-item взаємодії, рейтинги, текстові відгуки та метадані товарів;

- реалізувати попередню обробку даних, включно з очищенням записів, нормалізацією полів, об'єднанням взаємодій із метаданими та k-core фільтрацією;
- реалізувати базові рекомендаційні моделі для подальшого порівняння результатів;
- розробити метод прямої генерації synthetic user-item interactions за допомогою LLM;
- дослідити покращені підходи до генерації synthetic interactions, зокрема retrieval-based та collaborative-aware generation;
- розробити метод генерації LLM-based item-item similarity на основі текстових метаданих товарів;
- реалізувати гібридну модель Item-CF, що поєднує колаборативну item-item similarity з LLM-згенерованою семантичною подібністю;
- провести експериментальне оцінювання моделей за метриками Precision@K, Recall@K, HitRate@K та NDCG@K;
- порівняти ефективність різних типів LLM-сигналу та визначити, який із них є найбільш доцільним для застосування у рекомендаційних системах.

2.4 Вхідні та вихідні дані

Вхідними даними для роботи є локальний Amazon Appliances dataset. Він складається з двох основних файлів: Appliances.jsonl файл із відгуками та взаємодіями користувачів, meta_Appliances.jsonl файл із метаданими товарів.

Файл взаємодій містить інформацію про користувача, товар, рейтинг, заголовок відгуку, текст відгуку, часову мітку, кількість helpful votes та ознаку verified purchase. Ці дані використовуються для побудови user-item матриці, формування train/test split та оцінювання рекомендаційних моделей.

Файл метаданих містить інформацію про товари: назву, основну категорію, список категорій, характеристики, опис, магазин або бренд, середній рейтинг, кількість рейтингів і ціну. Ці поля використовуються для формування текстового представлення товару, на основі якого виконується TF-IDF retrieval та подальша генерація LLM-based item-item similarity.

Після попередньої обробки формуються такі основні вихідні файли:

- interactions.parquet та interactions.csv – очищені user-item взаємодії;
- items.parquet та items.csv – очищені метадані товарів;
- interactions_enriched.parquet та interactions_enriched.csv – об'єднана таблиця взаємодій і товарних метаданих;
- stats_report.csv – короткий статистичний звіт щодо набору даних;
- rating_distribution.csv – розподіл оцінок;
- top_items.csv – список товарів з найбільшою кількістю взаємодій.

Окремо в роботі формуються вихідні файли LLM-генерації. Для synthetic user-item interactions зберігаються синтетичні взаємодії, змішані навчальні набори та звіти про помилки генерації. Для item-item similarity зберігаються прямі та симетризовані item-item пари з оцінкою LLM similarity, а також debug-файли з кандидатами та summary-звіти.

2.5 Вимоги до експериментального дослідження

Експериментальне дослідження має забезпечити коректне та відтворюване порівняння рекомендаційних моделей. Для цього в роботі використовується однакова схема train/test split, однакові метрики оцінювання та фіксовані параметри negative sampling.

Розбиття даних виконується за схемою leave-one-out. Для кожного користувача остання взаємодія за часовою міткою використовується як тестова, а всі попередні взаємодії як навчальні. Такий підхід є поширеним у задачах top-K recommendation, оскільки імітує практичний сценарій передбачення наступної взаємодії користувача.

Для оцінювання застосовуються такі метрики:

- Precision@K – частка релевантних товарів серед перших K рекомендацій;
- Recall@K – частка знайдених релевантних товарів серед усіх релевантних товарів користувача;
- HitRate@K – показник того, чи потрапив тестовий товар у перші K рекомендацій;
- NDCG@K – метрика ранжування, яка враховує позицію релевантного товару у списку рекомендацій.

У роботі використовуються значення $K=5,10,20$. Такий вибір дозволяє оцінити якість моделі як для коротких списків рекомендацій, де важлива точність верхніх позицій, так і для довших списків, де важливим є ширше покриття релевантних товарів.

Оскільки повне ранжування по всьому каталогу товарів може бути обчислювально дорогим, використовується negative sampling. Для кожного користувача тестовий товар оцінюється разом із фіксованою кількістю негативних кандидатів, з якими користувач раніше не взаємодіяв. У роботі використовується $\text{NEGATIVE_SAMPLE_SIZE} = 100$.

Експериментальна частина повинна дати відповіді на такі дослідницькі питання:

- чи покращує пряме додавання LLM-generated synthetic user-item interactions якість рекомендацій;
- чи покращує LLM-generated item-item similarity якість Item-CF;
- як впливає кількість LLM source items на якість рекомендацій;
- яке значення llm_weight є оптимальним для гібридної моделі;
- чи зберігається ефект LLM item similarity при різних розмірах user subset.

3 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

3.1 Математична постановка задачі рекомендації

Рекомендаційна система працює з множиною користувачів (U), множиною об'єктів (I) та множиною спостережуваних взаємодій (R). У межах цієї роботи об'єктами є товари з категорії Amazon Appliances, а взаємодіями є оцінки та відгуки користувачів. Кожна взаємодія містить ідентифікатор користувача, ідентифікатор товару, рейтинг, текст відгуку та часову мітку.

Історію взаємодій можна подати у вигляді матриці:

$$R \in R^{|U| \times |I|}. \quad (3.1)$$

де рядки відповідають користувачам, стовпці – товарам, а елементи $R^{|U| \times |I|}$ – описує взаємодію користувача u з товаром i .

У випадку explicit feedback значенням може бути рейтинг від 1 до 5. У випадку implicit feedback достатньо самого факту взаємодії, тобто $R^{|U| \times |I|} = 1$, якщо користувач взаємодіяв із товаром, і значення відсутнє, якщо взаємодії не було.

У реальних рекомендаційних задачах матриця R є сильно розрідженою. Це означає, що кількість відомих взаємодій значно менша за кількість усіх можливих пар «користувач-товар». Саме ця властивість ускладнює побудову якісних рекомендацій, оскільки система має прогнозувати інтерес користувача до товарів, з якими він раніше не взаємодіяв.

У цій роботі розглядається задача top-K recommendation. Для кожного користувача u необхідно сформулювати впорядкований список товарів:

$$L_u^K = \{i_1, i_2, \dots, i_K\}. \quad (3.2)$$

де K – довжина списку рекомендацій.

Метою є розташувати релевантний товар якомога вище у цьому списку. На відміну від задачі передбачення рейтингу, top-K recommendation фокусується саме на якості ранжування, тобто на тому, які товари будуть показані користувачу першими.

У роботі використовується leave-one-out схема оцінювання. Для кожного користувача остання взаємодія за часом відноситься до тестової вибірки, а всі попередні взаємодії використовуються для навчання. Така схема імітує практичний сценарій, у якому система має передбачити наступний потенційно релевантний товар на основі попередньої історії користувача.

3.2 Метрики оцінювання рекомендаційних систем

Для оцінювання рекомендаційних систем у роботі використовуються метрики top-K recommendation. Вони дозволяють оцінити не лише факт наявності релевантного товару серед рекомендацій, а й позицію цього товару у списку [11].

Основними метриками є Precision@K, Recall@K, HitRate@K та NDCG@K.

Precision@K показує частку релевантних товарів серед перших (K) рекомендацій:

$$\text{Precision@K} = \frac{|Rel_u \cap L_u^K|}{K}. \quad (3.3)$$

де Rel_u – множина релевантних товарів для користувача u ;

L – список із K рекомендованих товарів.

Recall@K показує, яку частку релевантних товарів система змогла знайти серед перших (K) рекомендацій:

$$\text{Recall@K} = \frac{|Rel_u \cap L_u^K|}{|Rel_u|}. \quad (3.4)$$

У leave-one-out оцінюванні для кожного користувача є один тестовий товар. У такому випадку Recall@K набуває значення 1, якщо тестовий товар потрапив у top-K, і 0, якщо не потрапив. Середнє значення по всіх користувачах показує частку користувачів, для яких система змогла знайти правильний товар.

HitRate@K також показує факт потрапляння тестового товару в topK:

$$\text{HitRate@K} = \begin{cases} 1 & \text{якщо тестовий товар потрапив у } L_u^K \\ 0 & \text{інакше} \end{cases} \quad (3.5)$$

У leave-one-out схемі Recall@K і HitRate@K фактично збігаються, оскільки для кожного користувача є лише один релевантний тестовий товар. Проте обидві метрики зберігаються у роботі для повноти звітності та порівнянності з іншими дослідженнями.

NDCG@K враховує позицію релевантного товару у списку рекомендацій. Якщо релевантний товар знаходиться на першій позиції, значення метрики є максимальним. Якщо товар знаходиться нижче, його внесок зменшується логарифмічно:

$$\text{DCG@K} = \frac{\sum_{j=1}^K rel_j}{\log_2(j+1)}. \quad (3.6)$$

де rel_j дорівнює 1, якщо товар на позиції j є релевантним, і 0 в іншому випадку.

Нормалізована версія метрики має вигляд:

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}. \quad (3.7)$$

Оскільки в leave-one-out оцінюванні релевантний товар один, NDCG@K дорівнює:

$$\text{NDCG@K} = \begin{cases} \frac{1}{\log_2(\text{rank}+1)} & \text{якщо } \text{rank} \leq K \\ 0 & \text{інакше} \end{cases} \quad (3.8)$$

де rank – позиція тестового товару у списку рекомендацій, починаючи з 1.

У роботі використовуються значення ($K = 5, 10, 20$). Значення ($K=5$) дозволяє оцінити якість найвищих позицій списку, де помилка є найбільш критичною. Значення ($K=20$) показує ширше покриття релевантних товарів.

У таблиці 3.1 наведено основні метрики, які використовуються для оцінювання якості рекомендаційних моделей.

Таблиця 3.1 – Метрики оцінювання рекомендаційних систем

Метрика	Призначення	Особливість
Precision@K	Оцінює частку релевантних товарів серед top-K	Чутлива до довжини списку рекомендацій
Recall@K	Оцінює частку знайдених релевантних товарів	У leave-one-out збігається з HitRate@K
HitRate@K	Показує факт потрапляння тестового товару в top-K	Інтерпретується як частка успішних рекомендацій
NDCG@K	Оцінює якість ранжування з урахуванням позиції	Найважливіша для аналізу верхніх позицій списку

3.3 Базові моделі рекомендацій

У роботі розглядається кілька базових моделей, які використовуються для порівняння з підходами, що застосовують LLM-згенерований сигнал. Вибір моделей охоплює як прості popularity-based підходи, так і сильніші колаборативні методи.

3.3.1 Most Popular

Most Popular є найпростішою baseline-моделлю. Вона формує один і той самий список рекомендацій для всіх користувачів, ранжуючи товари за кількістю взаємодій у навчальній вибірці. Формально популярність товару i можна визначити як:

$$pop(i) = \sum_{u \in U} r_{ui}. \quad (3.9)$$

Перевагою цього підходу є простота та стабільність. Недоліком є повна відсутність персоналізації. Модель не враховує індивідуальну історію користувача і схильна рекомендувати лише найбільш популярні товари.

Most Popular використовується як нижня межа якості. Якщо складніша модель не перевищує цей підхід, її використання не має практичного сенсу.

3.3.2 Item-based Collaborative Filtering

Item-based Collaborative Filtering базується на обчисленні подібності між товарами. Якщо користувач раніше взаємодіяв з певними товарами, система рекомендує йому інші товари, подібні до них. Цей підхід був формалізований у роботі Sarwar та співавт. [2].

Для кожної пари товарів (i) та (j) обчислюється подібність на основі векторів взаємодій користувачів. Один із найпоширеніших варіантів – косинусна подібність:

$$\text{sim}(i, j) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \|\vec{j}\|}. \quad (3.10)$$

де $\vec{i}$ та $\vec{j}$ – вектори взаємодій користувачів із товарами i та j .

Для користувача (u), який має історію взаємодій (H_u), оцінка кандидата (i) може бути обчислена як сума подібностей до товарів з історії користувача:

$$score(u, i) = \sum_{j \in H_u} sim(i, j). \quad (3.11)$$

Item-CF є центральною baseline-моделлю в цій роботі, оскільки запропонований LLM-підхід безпосередньо модифікує item-item similarity. Перевагою Item-CF є інтерпретованість: рекомендацію можна пояснити через товари, з якими користувач вже взаємодівав. Недоліком є те, що подібність між товарами визначається лише на основі спільних користувачів, тому для малопопулярних товарів вона може бути ненадійною.

3.3.3 Matrix Factorization та BPR-MF

Matrix Factorization представляє користувачів і товари у вигляді латентних векторів. Ідея полягає в тому, що взаємодія між користувачем і товаром може бути апроксимована скалярним добутком їхніх латентних представлень.

Для top-K recommendation важливо оптимізувати не стільки точність передбачення рейтингу, скільки правильне ранжування товарів. Для цього використовується Bayesian Personalized Ranking. BPR навчає модель так, щоб для користувача позитивний товар мав вищий score, ніж негативний товар.

BPR-MF є важливою baseline-моделлю, оскільки вона оптимізує саме ранжування. На відміну від Item-CF, вона працює у латентному просторі та не використовує явну item-item similarity. Тому порівняння з BPR-MF дозволяє оцінити, чи є запропонований item-similarity підхід конкурентним порівняно з ranking-oriented моделлю.

3.3.4 LightGCN та послідовні моделі

LightGCN представляє user-item взаємодії як двочастковий граф і поширює embeddings користувачів та товарів по цьому графу. Модель була запропонована як спрощення графових згорткових мереж для задач рекомендації [8]. На відміну від складніших GCN-архітектур, LightGCN відмовляється від нелінійних трансформацій і зберігає лише механізм агрегування сусідніх вузлів.

LightGCN дозволяє враховувати не лише прямі взаємодії користувача з товарами, а й багатокрокові зв'язки через інших користувачів і товари. Це робить модель сильною сучасною baseline-моделлю. Водночас вона потребує більше обчислювальних ресурсів і ретельнішого налаштування.

Послідовні моделі, наприклад GRU4Rec, моделюють порядок взаємодій користувача. GRU4Rec використовує рекурентну нейронну мережу для передбачення наступного об'єкта у сесії або послідовності взаємодій [9]. Такі моделі особливо ефективні там, де порядок дій користувача має суттєве значення. Проте для наборів даних із короткими історіями користувачів їхня перевага може бути обмеженою.

У цій роботі LightGCN та GRU4Rec розглядаються як потенційні сильніші baseline-моделі.

3.4 Генерація синтетичних user-item взаємодій за допомогою LLM

Перший досліджуваний спосіб застосування LLM полягає у генерації синтетичних user-item interactions. Ідея цього підходу полягає в тому, щоб на основі історії користувача та списку товарів-кандидатів згенерувати додаткові взаємодії, які потім додаються до навчальної вибірки.

Для кожного користувача формується профіль на основі його попередніх взаємодій. Історія користувача поділяється на три групи:

- liked items – товари з рейтингом 4 або 5;

- neutral items – товари з рейтингом 3;
- disliked items – товари з рейтингом 1 або 2.

До prompt додаються назви товарів, рейтинги та за можливості короткі фрагменти текстів відгуків. Після цього формується candidate pool із товарів, з якими користувач раніше не взаємодіяв. LLM отримує інструкцію вибрати один або кілька товарів-кандидатів і спрогнозувати можливий рейтинг.

Після генерації виконується валідація:

- товар повинен належати до candidate pool;
- товар не повинен уже бути в історії користувача;
- рейтинг має бути в допустимому діапазоні;
- дублікати видаляються.

Початковий підхід використовував переважно популярні товари як кандидатів. Така стратегія виявилася проблемною, оскільки LLM схильна вибирати очевидні та популярні товари, що призводить до popularity collapse. Для зменшення цього ефекту було розроблено retrieval-based generation. У ньому candidate pool формується з трьох джерел:

- TF-IDF semantic candidates;
- popular candidates;
- random long-tail candidates.

TF-IDF retrieval дозволяє знайти товари, текстово схожі на попередні товари користувача. Додавання long-tail кандидатів підвищує різноманітність і зменшує концентрацію генерації на кількох популярних товарах.

Подальше покращення – collaborative-aware generation. У цьому підході для кожного користувача знаходяться схожі користувачі за перетином train items. Кандидатами стають товари, з якими взаємодіяли схожі користувачі, а також семантичні кандидати з TF-IDF retrieval. LLM отримує можливість повернути 0–2 взаємодії або порожній список, якщо сильного збігу немає.

Доцільно додати схему трьох підходів до генерації синтетичних взаємодій.

Незважаючи на логічність цього підходу, він має фундаментальне обмеження. User-item interaction є поведінковим фактом, а не лише семантичним зв'язком. Користувач може бути зацікавлений у товарі, але не купити його через ціну, доступність, бренд, потребу в конкретний момент часу або відсутність експозиції. LLM не має доступу до цих факторів, тому може створювати правдоподібні, але поведінково недостовірні взаємодії.

3.5 Генерація синтетичних item-item similarity за допомогою LLM

Другий підхід полягає у використанні LLM для побудови item-item semantic similarity. На відміну від synthetic interactions, цей підхід не створює штучних записів про поведінку користувачів. Він формує додаткові зв'язки між товарами на основі їхнього змісту, функції та характеристик.

Для кожного товару формується текстове представлення:

$$text_i = title_i + category_i + features_i + description_i + store_i. \quad (3.12)$$

У практичній реалізації використовуються такі поля метаданих:

- title;
- main_category;
- categories;
- features;
- description;
- store;
- average_rating;
- rating_number;
- price.

Спочатку для кожного source item виконується TF-IDF retrieval. Для цього всі товари перетворюються у TF-IDF вектори, а для source item обчислюється косинусна подібність з іншими товарами. Найближчі кандидати передаються до LLM. TF-IDF у цьому випадку використовується не як фінальний метод рекомендації, а як швидкий фільтр для звуження простору кандидатів. Метод TF-IDF є класичним способом статистичного зважування текстових ознак і реалізований, зокрема, у бібліотеці scikit-learn [19].

LLM отримує source item і список кандидатів. Завдання моделі – вибрати товари, які є:

- семантично схожими;
- взаємозамінними;
- функціонально комплементарними;
- близькими за призначенням або контекстом використання.

Після отримання відповіді виконується валідація:

- similar_item_id має належати до candidate pool;
- source item не може бути подібним до самого себе;
- similarity_score має бути числом від 0 до 1;
- дублікати видаляються;
- пари симетризуються.

Симетризація означає, що для кожної пари (i, j) створюється також пара (j, i). Це необхідно, оскільки Item-CF використовує item-item similarity як двонаправлений зв'язок.

У межах поточної реалізації LLM item-similarity generation формує кілька технічних файлів. Вони використовуються для аналізу якості генерації, відтворення експериментів і подальшого оцінювання Hybrid Item-CF.

У таблиці 3.2 наведено приклад LLM-згенерованих item-item зв'язків та їх інтерпретацію.

Таблиця 3.2 – Приклад LLM-згенерованих item-item similarity pairs

Source item	Similar item	LLM similarity	Інтерпретація
Keurig-related product	Disposable filters for Keurig brewers	0.92	Функціонально близькі товари для кавоварок
Appliance replacement part	Compatible replacement/accessory item	0.85–0.90	Товари належать до одного ремонтного або аксесуарного контексту

3.6 Гібридна модель

Основна запропонована модель поєднує колаборативну item-item similarity з LLM-generated item-item semantic similarity. Ідея полягає в тому, щоб не змінювати user-item матрицю, а лише скоригувати item-item similarity там, де LLM надала додатковий семантичний зв'язок.

Фінальна подібність визначається як:

$$\text{sim}_{final}(i, j) = (1 - \alpha) \cdot \text{sim}_{cf}(i, j) + \alpha \cdot \text{sim}_{llm}(i, j). \quad (3.13)$$

де α – вага LLM-сигналу;

$\text{sim}_{cf}(i, j)$ – колаборативна подібність між товарами i та j ;

$\text{sim}_{llm}(i, j)$ – LLM-згенерована семантична подібність між цими товарами.

У програмній реалізації ця формула застосовується лише для тих пар товарів, для яких існує LLM similarity pair. Якщо LLM-пара відсутня, використовується звичайна колаборативна подібність.

Такий підхід має важливу перевагу: реальні user-item взаємодії не змінюються. На відміну від synthetic user-item interactions, LLM similarity не додає штучних поведінкових фактів. Вона лише коригує item neighborhoods, тобто допомагає моделі краще оцінити близькість між товарами там, де колаборативний сигнал є слабким або неповним.

Параметр α , який у коді позначається як `llm_weight`, контролює силу впливу LLM-сигналу. Якщо $\alpha = 0$, модель збігається зі звичайним Item-CF. Якщо α занадто велике, модель може надмірно покладатися на семантичний сигнал і втрачати колаборативну точність. Тому оптимальне значення α визначається експериментально.

3.7 Архітектура програмного забезпечення

Програмна реалізація роботи побудована як послідовний експериментальний *pipeline*, який охоплює всі етапи дослідження: підготовку даних, формування тимчасових підмножин, генерацію LLM-сигналів, навчання рекомендаційних моделей, оцінювання результатів і подальший аналіз. Така архітектура дозволяє окремо запускати кожен етап, зберігати проміжні результати у файлах і повторно використовувати вже згенеровані дані без повторного звернення до LLM.

Загальна структура програмного забезпечення складається з таких основних модулів:

- модуль попередньої обробки даних;
- модуль формування тимчасового набору даних;
- модуль генерації *synthetic user-item interactions*;
- модуль генерації LLM-based item-item similarity;
- модуль оцінювання рекомендаційних моделей;
- модуль аналізу експериментальних результатів.

Загальна архітектура програмного забезпечення та послідовність виконання подана на рисунку 3.1.

Програмна реалізація виконана мовою Python. Вибір Python зумовлений наявністю широкого набору бібліотек для обробки даних, машинного навчання, побудови рекомендаційних моделей, роботи з розрідженими матрицями та інтеграції із зовнішніми API. Python також зручний для експериментального дослідження, оскільки дозволяє швидко

змінювати параметри, запускати окремі етапи pipeline і аналізувати проміжні результати.

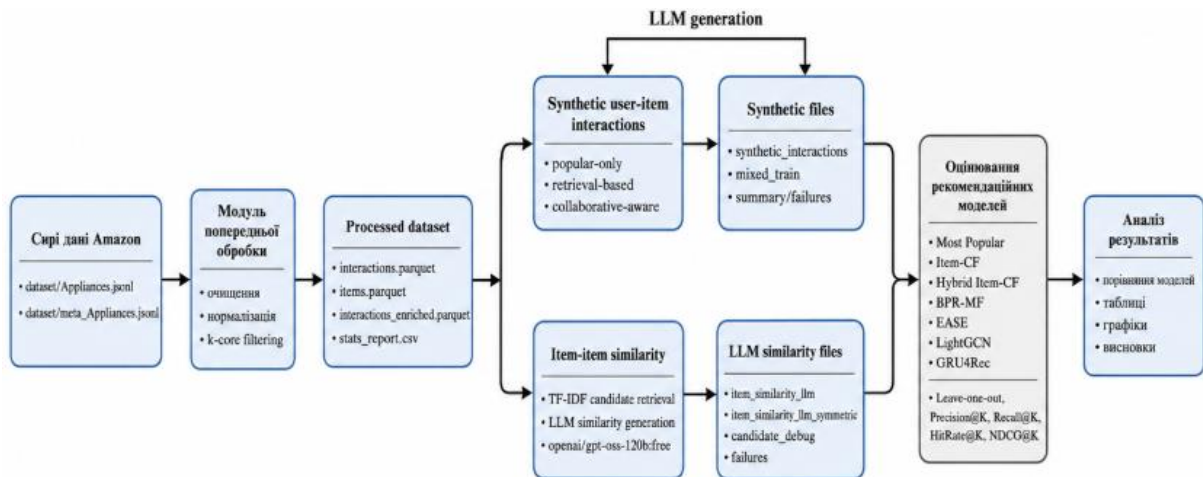


Рисунок 3.1 – Загальна архітектура програмного забезпечення (рисунок створено самостійно)

У роботі використано набір Python-бібліотек, які забезпечують обробку даних, побудову рекомендаційних моделей, роботу з текстом та інтеграцію з LLM.

Бібліотека `pandas` використовується для зчитування JSONL, CSV і Parquet-файлів, очищення таблиць, групування даних, об'єднання взаємодій з метаданими та формування статистичних звітів. Вона є основною бібліотекою для табличної обробки даних у проєкті.

Бібліотека `numpy` використовується для числових операцій, роботи з масивами, обчислення метрик, `negative sampling` і формування векторних представлень.

Бібліотека `scipy.sparse` використовується для ефективного зберігання user-item матриці та item-item similarity matrix. Це важливо, оскільки рекомендаційні дані є розрідженими, і зберігання таких матриць у dense-форматі було б неефективним.

Бібліотека `scikit-learn` використовується для побудови TF-IDF представлень товарів через `TfidfVectorizer`, обчислення `cosine similarity` та нормалізації матриць. Саме TF-IDF використовується для попереднього відбору кандидатів перед передачею їх до LLM. `Scikit-learn` є стандартною бібліотекою машинного навчання в Python і широко використовується для експериментальних ML-досліджень [19].

Бібліотека `openai` використовується як клієнт для звернення до `OpenRouter-compatible API`. Хоча використовується OpenAI-сумісний клієнт, фактичний доступ до моделі здійснюється через `OpenRouter endpoint`.

Також у роботі використовуються стандартні бібліотеки Python: `json`, `os`, `time`, `pathlib`, `typing`, `re`. Вони потрібні для роботи з файлами, JSON-відповідями, регулярними виразами, затримками між API-викликами та типізацією.

Для генерації синтетичних сигналів у роботі використовується велика мовна модель: `openai/gpt-oss-120b`.

Доступ до моделі здійснюється через `OpenRouter-compatible API`.

Модель використовується у двох основних задачах:

- генерація `synthetic user-item interactions`;
- генерація `item-item semantic similarity`.

Вибір моделі `openai/gpt-oss-120b` зумовлений кількома причинами. По-перше, модель має достатню мовну потужність для аналізу описів товарів, категорій, характеристик і текстових відгуків. Це важливо, оскільки задача `item-item similarity` потребує не простого збігу ключових слів, а розуміння функціональної або семантичної близькості товарів. По-друге, модель доступна через `OpenRouter-compatible API`, що дозволяє інтегрувати її у локальний Python pipeline без необхідності розгортання великої моделі на власному обладнанні. По-третє, використання free-моделі знижує вартість експериментів, що важливо для дослідження з великою кількістю API-викликів.

У фінальній версії pipeline для LLM-запитів використовуються такі параметри:

```
MODEL_NAME = "openai/gpt-oss-120b:free"
TEMPERATURE = 0.1
MAX_TOKENS = 500
SLEEP_BETWEEN_CALLS = 6
MAX_RETRIES = 5
```

Низьке значення `TEMPERATURE = 0.1` використовується для зменшення випадковості відповідей. Це важливо, оскільки LLM повинна повертати не креативний текст, а структурований JSON із валідними `item_id` та `similarity_score`. Параметр `MAX_TOKENS = 500` обмежує довжину відповіді, оскільки очікуваний результат є коротким JSON-масивом. Параметри `SLEEP_BETWEEN_CALLS` і `MAX_RETRIES` використовуються для стабільної роботи з API та обробки тимчасових помилок або `rate limit`.

Модуль попередньої обробки даних відповідає за перетворення сирих JSONL-файлів у структуровані таблиці, придатні для подальших експериментів. Він читає два початкові файли:

- `dataset/Appliances.jsonl`;
- `dataset/meta_Appliances.jsonl`.

У модулі реалізовано функції очищення тексту, спискових полів і числових значень. Текстові поля нормалізуються: зайві пробіли видаляються, списки перетворюються в рядки, словники перетворюються в текстові описи виду `key: value`. Це потрібно, оскільки метадані Amazon можуть містити поля різних типів: рядки, списки, вкладені списки або словники.

Для `reviews`-файлу формується таблиця `interactions`, яка містить:

- `user_id`;
- `item_id`;
- `asin`;
- `rating`;
- `review_title`;

- review_text;
- timestamp;
- datetime;
- helpful_vote;
- verified_purchase.

Для metadata-файлу формується таблиця items, яка містить:

- item_id;
- title;
- main_category;
- categories;
- features;
- description;
- store;
- average_rating;
- rating_number;
- price.

Після очищення застосовується k-core filtering з параметрами:

```
MIN_USER_INTERACTIONS = 2
MIN_ITEM_INTERACTIONS = 2
```

Це гарантує, що кожен користувач і кожен товар у фінальному наборі мають мінімальну кількість взаємодій, необхідну для коректного leave-one-out оцінювання.

У таблиці 3.3 наведено вихідні файли, які формуються модулем попередньої обробки даних.

Таблиця 3.3 – Вихідні файли модуля попередньої обробки

Файл	Призначення
interactions.parquet	Очищені user-item взаємодії
interactions.csv	CSV-копія очищених взаємодій
items.parquet	Очищені метадані товарів

Продовження таблиці 3.3

Файл	Призначення
items.csv	CSV-копія метаданих
interactions_enriched.parquet	Об'єднання взаємодій з метаданими
interactions_enriched.csv	CSV-копія enriched dataset
stats_report.csv	Загальна статистика processed dataset
rating_distribution.csv	Розподіл рейтингів
top_items.csv	Найпопулярніші товари

Модуль генерації synthetic user-item interactions був реалізований для перевірки початкової гіпотези про те, що LLM може доповнити навчальну вибірку штучними взаємодіями. Він використовує train history користувача, формує профіль інтересів і передає його до LLM разом зі списком товарів-кандидатів.

Профіль користувача містить поділ історії на три групи:

- liked items: рейтинг 4 або 5;
- neutral items: рейтинг 3;
- disliked items: рейтинг 1 або 2.

До prompt також можуть додаватися фрагменти review text. Це дозволяє LLM врахувати не лише факт взаємодії, а й текстове пояснення користувача.

Було реалізовано три варіанти цього модуля:

- початкова генерація з popular candidates;
- retrieval-based generation із TF-IDF candidate retrieval;
- collaborative-aware generation із використанням схожих користувачів.

Вихідними файлами є synthetic interactions, mixed train set та summary/failure reports. Вони зберігаються у відповідних директоріях:

- dataset/dataset_temp_llm/;
- dataset/dataset_temp_llm_retrieval/;
- dataset/dataset_temp_llm_collaborative/.

Цей модуль важливий не лише як технічний етап, а і як частина дослідницької логіки роботи. Він дозволив експериментально перевірити, чи корисно напряду додавати LLM-generated user behavior до навчального набору.

Модуль генерації LLM-based item-item similarity є ключовим для фінального підходу роботи. Його завдання – сформувати семантичні item-item зв'язки на основі метаданих товарів, не змінюючи реальну user-item матрицю.

На першому етапі обчислюється item statistics: кількість взаємодій і середній рейтинг для кожного товару. Далі ці статистики об'єднуються з метаданими товарів. Для кожного товару формується текстове представлення з полів:

- title;
- main_category;
- categories;
- features;
- description;
- store.

Після цього будується TF-IDF матриця за допомогою TfidfVectorizer.

Для кожного source item обчислюється cosine similarity до інших товарів і відбираються найближчі кандидати. У поточній конфігурації:

```
N_SOURCE_ITEMS = 1000
CANDIDATES_PER_ITEM = 25
SIMILAR_ITEMS_PER_SOURCE = 10
```

LLM отримує source item і 25 candidate items. У prompt задається вимога повернути лише JSON-масив із полями similar_item_id і

similarity_score. LLM має обрати до 10 товарів, які є справді схожими, взаємозамінними або функціонально комплементарними.

Після отримання відповіді виконується валідація:

- відповідь має бути валідним JSON-масивом;
- similar_item_id має належати до candidate pool;
- similar_item_id не повинен збігатися з source item;
- similarity_score має бути в діапазоні від 0 до 1;
- дублікати видаляються.

Після цього створюється symmetric similarity file: для кожної пари (i, j) додається зворотна пара (j, i). Це необхідно для подальшого використання в Item-CF.

У процесі генерації реалізовано resume support. Якщо частина source items вже була оброблена, скрипт зчитує наявні файли item_similarity_llm.csv, item_similarity_failures.csv і item_similarity_candidate_debug.csv, після чого продовжує генерацію лише для необроблених товарів. Це важливо, оскільки генерація для 1000 source items потребує багатьох API-викликів і може перериватися через rate limit або технічні помилки.

Модуль оцінювання відповідає за навчання рекомендаційних моделей і розрахунок top-K метрик. Він читає підготовлені interactions та, за потреби, LLM item similarity file.

На першому етапі виконується leave-one-out split. Для кожного користувача остання взаємодія за timestamp або datetime стає тестовою, а всі попередні – навчальними. Потім формується user-item matrix у sparse CSR format. Для оцінювання кожного користувача формується candidate set, який складається з одного позитивного тестового товару та 100 негативних товарів, з якими користувач раніше не взаємодівав. Модель ранжує ці кандидати, після чого обчислюються Precision@K, Recall@K, HitRate@K і NDCG@K для (K = 5, 10, 20).

Модуль аналізу результатів відповідає за порівняння моделей, побудову таблиць і підготовку даних для графіків. На цьому етапі об'єднуються результати різних експериментів, розраховуються абсолютні та відносні зміни метрик, визначаються найкращі значення `llm_weight` і порівнюються різні масштаби LLM item similarity.

Основні результати зберігаються у CSV-файлах. Цей модуль дозволяє отримати таблиці «comparison vs baseline», «best by NDCG@20», «best by Recall@20» та інші підсумкові представлення, які використовуються в експериментальному розділі.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

4.1 Набір даних Amazon Appliances

Експериментальне дослідження виконано на локальному наборі даних Amazon Appliances. Набір містить реальні взаємодії користувачів із товарами, числові оцінки, текстові відгуки та метадані товарів. Його використання є доцільним для цієї роботи, оскільки він поєднує два типи інформації: поведінкову, представлену user-item взаємодіями, та семантичну, представлену назвами, описами, характеристиками й категоріями товарів.

Початкові дані складаються з двох JSONL-файлів:

- dataset/Appliances.jsonl файл із відгуками та user-item взаємодіями;
- dataset/meta_Appliances.jsonl файл із метаданими товарів.

Файл взаємодій містить такі поля: user_id, asin, parent_asin, rating, title, text, timestamp, helpful_vote, verified_purchase. Основними полями для рекомендаційної задачі є ідентифікатор користувача, ідентифікатор товару, рейтинг і час взаємодії. Текст відгуку використовувався на етапі побудови профілю користувача для експериментів із прямою генерацією synthetic user-item interactions.

Файл метаданих містить інформацію про товари: title, main_category, categories, features, description, store, average_rating, rating_number, price та інші поля. Ці дані використовувалися для побудови текстового представлення товару під час генерації LLM-based item-item similarity.

Особливістю набору Amazon Appliances є те, що багато товарів належать до близьких функціональних груп: аксесуари для кавоварок, фільтри, деталі для пральних і сушильних машин, холодильні компоненти, витратні матеріали та запасні частини. Це робить набір придатним для перевірки гіпотези про користь семантичної item-item подібності, оскільки

частина товарів може бути функціонально пов'язаною навіть за відсутності великої кількості спільних користувацьких взаємодій.

Розподіл кількості взаємодій на користувача подано на рисунку 4.1. З рисунка видно, що більшість користувачів має невелику кількість взаємодій, тоді як лише незначна частина користувачів є дуже активною. Такий розподіл має довгий хвіст і підтверджує розрідженість user-item матриці.

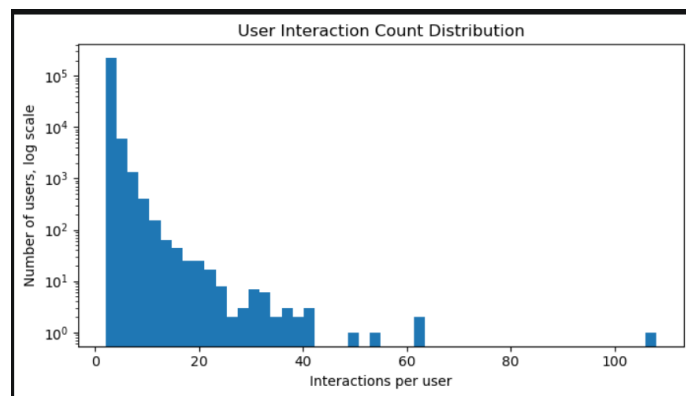


Рисунок 4.1 – Розподіл кількості взаємодій на користувача (рисунок створено самостійно)

Розподіл кількості взаємодій на товар подано на рисунку 4.2.

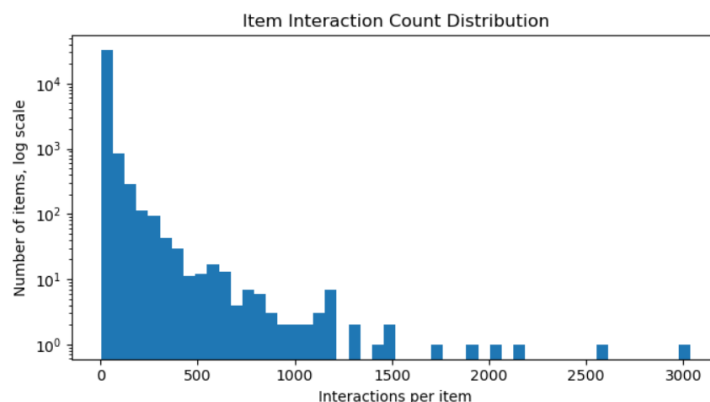


Рисунок 4.2 – Розподіл кількості взаємодій на товар (рисунок створено самостійно)

На рисунку показано, що більшість товарів має малу кількість взаємодій, а невелика група популярних товарів отримує значно більше відгуків і оцінок. Це свідчить про нерівномірність популярності товарів та наявність popularity bias у наборі даних.

4.2 Попередня обробка даних

Попередня обробка виконувалася локальним Python-скриптом. На першому етапі зчитувалися файли Appliances.jsonl та meta_Appliances.jsonl. Оскільки формат JSONL передбачає один JSON-об'єкт у кожному рядку, дані оброблялися послідовно. Для текстових полів виконувалося очищення від зайвих пробілів, нормалізація пропущених значень, перетворення списків і словників у текстовий формат.

Для таблиці взаємодій було сформовано такі основні поля: user_id, item_id, asin, rating, review_title, review_text, timestamp, datetime, helpful_vote, verified_purchase. Для таблиці товарів було сформовано поля item_id, title, main_category, categories, features, description, store, average_rating, rating_number, price.

Після первинного очищення застосовувалася ітеративна k-core у фінальному наборі залишалися лише користувачі з мінімум двома взаємодіями та товари з мінімум двома взаємодіями. Фільтрація виконувалася ітеративно, оскільки видалення користувачів може зменшити кількість взаємодій для товарів, а видалення товарів – зменшити кількість взаємодій для користувачів.

4.3 Методика експериментального оцінювання

Усі рекомендаційні моделі оцінювалися за схемою leave-one-out. Для кожного користувача остання взаємодія за часом використовувалася як тестова, а всі попередні взаємодії формували навчальну вибірку. Така схема

відповідає задачі передбачення наступної потенційної взаємодії користувача.

Під час оцінювання модель ранжувала тестовий товар разом із набором негативних кандидатів. Негативні кандидати – це товари, з якими користувач раніше не взаємодіяв.

Для кожного користувача обчислювалися $Precision@K$, $Recall@K$, $HitRate@K$ та $NDCG@K$, після чого значення усереднювалися за всіма користувачами тестової вибірки. У leave-one-out схемі $Recall@K$ і $HitRate@K$ мають однакове значення, оскільки для кожного користувача є один тестовий релевантний товар.

Основна увага приділялася $NDCG@K$, оскільки ця метрика враховує не лише факт потрапляння тестового товару до top-K, а й його позицію у списку. Для практичних рекомендаційних систем це важливо, оскільки товари на перших позиціях мають найбільшу ймовірність бути побаченими користувачем.

4.4 Реалізація рекомендаційних моделей

Most Popular ранжує товари за кількістю взаємодій у навчальній вибірці. Ця модель не персоналізує рекомендації, але є важливою базовою лінією, оскільки у sampled-negative evaluation популярні товари часто мають сильну перевагу над випадково вибраними негативними кандидатами.

Item-CF обчислює подібність між товарами на основі спільних користувачів. Для користувача рекомендації формуються через товари, подібні до його попередніх взаємодій. Ця модель є центральною для першого LLM-підходу, оскільки item-item similarity може бути безпосередньо доповнена семантичним сигналом.

BPR-MF використовує латентні представлення користувачів і товарів та оптимізує попарне ранжування позитивних взаємодій вище за негативні.

Ця модель дозволяє оцінити, як LLM-сигнал впливає на latent factor recommender.

LightGCN представляє user-item взаємодії як двочастковий граф і навчає embeddings через поширення сигналу по графу. GRU4Rec моделює послідовність взаємодій користувача та прогнозує наступний товар. Ці моделі використовувалися як сильніші сучасні baselines для перевірки того, чи може LLM-сигнал покращувати не лише класичні, а й нейронні рекомендаційні моделі.

4.5 Експеримент із прямою генерацією synthetic user-item interactions

Першою досліджуваною стратегією було пряме генерування synthetic user-item interactions за допомогою LLM. Ідея полягала в тому, щоб доповнити навчальну вибірку штучними записами про можливі взаємодії користувачів із товарами.

Для кожного цільового користувача формувався профіль на основі його train history. Історія поділялася на три групи:

- liked items: рейтинг 4 або 5;
- neutral items: рейтинг 3;
- disliked items: рейтинг 1 або 2.

До prompt додавалися назви товарів, рейтинги та короткі фрагменти review text. LLM отримувала також список товарів-кандидатів, з якими користувач раніше не взаємодіяв, і мала повернути JSON-масив із item_id та прогнозованим rating.

Початковий варіант генерації використовував переважно популярні товари як кандидатів. Такий підхід призвів до popularity collapse: LLM систематично вибирала невелику кількість очевидних популярних товарів. Було отримано 1000 синтетичних взаємодій, але лише 11 унікальних товарів. Частка оцінок rating = 5 становила приблизно 91,8 %, а top-10 товарів покривали майже всі синтетичні записи.

Такі результати показали, що LLM може створювати формально валідні взаємодії, але розподіл цих взаємодій не відповідає реальному поведінковому розподілу. Додавання таких записів до навчальної вибірки не покращило якість рекомендацій і в окремих випадках погіршило результати.

4.6 Retrieval-based та collaborative-aware synthetic interactions

Для зменшення popularity collapse було реалізовано retrieval-based generation. У цьому підході candidate pool формувався з трьох джерел:

- семантичні кандидати, знайдені за допомогою TF-IDF;
- популярні товари;
- випадкові long-tail товари.

TF-IDF retrieval дозволив знаходити товари, текстово схожі на ті, з якими користувач уже взаємодіяв. У результаті LLM отримувала більш різноманітний і змістовно релевантний candidate pool.

Retrieval-based generation створила 998 синтетичних рядків із 809 унікальними товарами. Частка rating = 5 знизилася приблизно до 48 %. Це означає, що генерація стала значно менш концентрованою на кількох популярних товарах.

У таблиці 4.1 наведено порівняння popular-only та retrieval-based генерації synthetic user-item interactions.

Таблиця 4.1 – Порівняння popular-only та retrieval-based генерації

Підхід	Synthetic rows	Unique items	Частка rating = 5
Popular-only generation	1000	11	91,8 %
Retrieval-based generation	998	809	48,0 %

Якісний аналіз показав, що retrieval-based synthetic interactions стали семантично більш правдоподібними. Наприклад, користувачі з історією

взаємодій із деталями для пральних або сушильних машин отримували синтетичні товари з близького ремонтного контексту. Користувачі, пов'язані з кавоварками або Keurig-аксесуарами, отримували кавові фільтри, капсули або сумісні аксесуари.

Проте покращення семантичної якості не забезпечило стабільного покращення рекомендаційних метрик. Це стало важливим проміжним результатом: синтетична взаємодія може бути змістовно логічною, але все одно не бути корисною для колаборативного навчання.

Наступним варіантом була collaborative-aware generation. Для кожного користувача знаходилися схожі користувачі за перетином train items. Candidate pool формувався з товарів, з якими взаємодіяли ці схожі користувачі, а також із TF-IDF semantic candidates. LLM могла повернути від 0 до 2 синтетичних взаємодій або порожній список, якщо сильного збігу не було.

Collaborative-aware generation дала окремі локальні покращення для Item-CF, зокрема для Recall@20 і NDCG@20, але ефект залишався слабким і нестабільним. Для Matrix Factorization і BPR-MF додавання synthetic interactions часто погіршувало результат, оскільки штучні записи вносили шум у latent factor structure.

Головний висновок цього етапу полягає в тому, що пряме додавання synthetic user-item interactions є ненадійним. User-item interaction є поведінковим фактом, а не лише семантичним зв'язком. LLM може визначити, що товар логічно підходить користувачу, але не може достовірно змодельовати експозицію товару, поточну потребу, ціну, доступність, часовий контекст і реальний намір покупки.

4.7 LLM-generated item-item similarity

Після нестабільних результатів із synthetic interactions було змінено основний напрям використання LLM. Замість генерації штучної поведінки

користувачів LLM використовувалася для побудови item-item semantic similarity. Цей підхід не змінює реальну user-item матрицю, а лише доповнює item neighborhoods семантичними зв'язками.

Для кожного товару формувався текстовий опис на основі таких полів:

- title;
- main_category;
- categories;
- features;
- description;
- store.

Далі за допомогою TF-IDF retrieval для кожного source item відбиралися найближчі товари-кандидати. У фінальній конфігурації для одного source item використовувалося 25 кандидатів. LLM отримувала source item і candidate items та мала вибрати до 10 товарів, які є справді схожими, взаємозамінними або функціонально комплементарними.

Відповідь LLM поверталася у JSON-форматі. Після цього виконувалася валідація:

- similar_item_id має належати до candidate pool;
- similar_item_id не повинен збігатися з source item;
- similarity_score має бути числом від 0 до 1;
- дублікати видаляються;
- item-item пари симетризуються.

Симетризація означає, що для кожної пари (i, j) створюється також пара (j, i) . Це потрібно для використання LLM-зв'язків у item-based моделях, де подібність між товарами розглядається як двонапрявлена.

У фінальній серії експериментів було сформовано LLM item similarity для 100, 500 та 1000 source items. Для 1000 source items отримано 11440 symmetric item-item pairs.

4.8 Hybrid Item-CF + LLM item similarity

Для використання LLM item similarity було реалізовано гібридну модель Item-CF. У ній фінальна item-item similarity обчислювалася як зважена комбінація колаборативної та LLM-згенерованої подібності:

$$\text{sim}_{final}(i, j) = (1 - \alpha) \cdot \text{sim}_{cf}(i, j) + \alpha \cdot \text{sim}_{llm}(i, j). \quad (4.1)$$

Ця формула застосовувалася лише для тих item-item пар, для яких існує LLM similarity. Якщо LLM-пари немає, модель використовує звичайну collaborative similarity. Такий підхід не змінює реальні user-item взаємодії й не створює штучних поведінкових записів.

У роботі перевірялися такі значення llm_weight: 0.10, 0.20, 0.30, 0.40. Окремо оцінювався baseline llm_weight = 0.0, який відповідає звичайному Item-CF.

У таблиці 4.2 наведено найкращі результати Hybrid Item-CF за метрикою NDCG@20.

Таблиця 4.2 – Найкращі результати Hybrid Item-CF за NDCG@20

Source items	LLM pairs	Найкращий llm_weight	Baseline NDCG@20	Best NDCG@20	Relative gain
100	836	0.40	0.070771	0.071605	+1.18 %
500	3820	0.10	0.070771	0.072234	+2.07 %
1000	11440	0.30	0.070771	0.073844	+4.34 %

Результати демонструють послідовне покращення зі збільшенням кількості LLM source items. Це підтверджує гіпотезу про важливість coverage item-similarity graph: чим більше товарів покрито LLM-згенерованими семантичними зв'язками, тим частіше модель може використати додатковий сигнал під час ранжування.

Найсильніші результати отримано для 1000 source items. Найкраще значення NDCG@20 досягнуто при $llm_weight = 0.30$, тоді як найкраще Recall@20 отримано при $llm_weight = 0.10$.

У таблиці 4.3 наведено порівняння базового Item-CF та Hybrid Item-CF для 1000 source items.

Таблиця 4.3 – Порівняння Item-CF та Hybrid Item-CF

Метрика	Original Item-CF	Hybrid, weight = 0.10	Hybrid, weight = 0.30
Precision@5	0.010288	0.010894	0.010974
Recall@5	0.051442	0.054468	0.054872
NDCG@5	0.031595	0.033992	0.034708
Precision@10	0.010410	0.010773	0.010672
Recall@10	0.104095	0.107726	0.106718
NDCG@10	0.048443	0.051021	0.051296
Precision@20	0.009673	0.009855	0.009845
Recall@20	0.193464	0.197095	0.196893
NDCG@20	0.070771	0.073347	0.073844

Найбільше відносне покращення спостерігається на NDCG@5. Це означає, що LLM item similarity насамперед покращує верхні позиції списку рекомендацій.

У таблиці 4.4 наведено найкращі відносні покращення Hybrid Item-CF для 1000 source items.

Таблиця 4.4 – Найкраще відносне покращення для 1000 source items

Метрика	Baseline	Best Hybrid	Relative gain	Найкращий weight
NDCG@5	0.031595	0.034708	+9.85 %	0.30
NDCG@10	0.048443	0.051296	+5.89 %	0.30
NDCG@20	0.070771	0.073844	+4.34 %	0.30
Recall@20	0.193464	0.197095	+1.88 %	0.10

Отже, для Item-CF найкращим способом використання LLM є не постобробка готового списку рекомендацій, а інтеграція LLM item similarity безпосередньо у функцію item-item similarity.

4.9 Вплив розміру user subset

Окремо аналізувався вплив розміру user subset. Для цього використовувалися підмножини з 1000, 5000, 10000 та 20000 користувачів. Мета цього експерименту – перевірити, чи зберігається користь LLM item similarity при різних обсягах колаборативних даних.

Для 500-source LLM similarity Hybrid Item-CF покращував якість для всіх протестованих розмірів user subset.

У таблиці 4.5 наведено вплив розміру user subset на ефективність LLM item similarity.

Таблиця 4.5 – Вплив розміру user subset для

User subset	Baseline NDCG@20	Hybrid NDCG@20	Relative gain
1000	0.076162	0.078685	+3.31 %
5000	0.071423	0.072920	+2.10 %
10000	0.072190	0.074091	+2.63 %
20000	0.070688	0.071300	+0.87 %

Найбільший відносний ефект спостерігався для меншого набору користувачів. Для 1000 користувачів NDCG@20 покращився з 0.076162 до 0.078685, тобто на +3.31 %. Для 20000 користувачів покращення було меншим +0.87 %.

Ці результати свідчать, що LLM item similarity особливо корисна в умовах слабшого колаборативного сигналу. Для менших user subsets модель має менше реальних взаємодій, тому семантичне доповнення item neighborhoods дає більший відносний ефект. Для більших підмножин

колаборативна модель має більше поведінкової інформації, тому внесок LLM-сигналу зменшується.

4.10 LLM recent-history reranking

Після перевірки Hybrid Item-CF було досліджено ще один спосіб використання LLM item similarity – reranking готових рекомендацій. У цьому підході основна рекомендаційна модель спочатку навчається лише на реальних взаємодіях. Після цього її score для кожного candidate item доповнюється LLM-сигналом, який вимірює семантичну близькість кандидата до останніх товарів в історії користувача.

Для користувача беруться останні $\text{recent_n} = 5$ train items. Для кожного кандидата обчислюється максимальна LLM similarity до цих останніх товарів із урахуванням recency decay.

У цьому експерименті використовувалися значення $\text{llm_weight} = 0.02, 0.05, 0.10, 0.20, 0.30$. Найкращим для більшості моделей стало значення $\text{llm_weight} = 0.30$.

Цей підхід має важливу методичну перевагу: LLM не змінює навчальні дані та не додає штучних user-item interactions. Вона лише коригує порядок кандидатів, які вже були оцінені основною рекомендаційною моделлю.

LLM recent-history reranking було застосовано до GRU4Rec і LightGCN. Експерименти виконувалися на однаковому 20000-user split після top-user selection і k-core фільтрації. Після розбиття було отримано:

- train rows: 69426;
- test rows: 19922;
- users: 19922;
- items: 12571.

Для GRU4Rec регулярна модель показала $NDCG@20 = 0.239081$ та $Recall@20 = 0.452967$. Після LLM reranking з $llm_weight = 0.30$ $NDCG@20$ зріс до 0.250784, а $Recall@20$ – до 0.458137.

У таблиці 4.6 наведено порівняння регулярної GRU4Rec та GRU4Rec з LLM reranking.

Таблиця 4.6 – GRU4Rec regular vs GRU4Rec + LLM reranking

Метрика	GRU4Rec	GRU4Rec + LLM, w = 0.30	Relative gain
Precision@5	0.048640	0.051019	+4.89 %
Recall@5	0.243198	0.255095	+4.89 %
NDCG@5	0.180166	0.193858	+7.60 %
Precision@10	0.033044	0.033837	+2.40 %
Recall@10	0.330439	0.338370	+2.40 %
NDCG@10	0.208279	0.220694	+5.96 %
Precision@20	0.022648	0.022907	+1.14 %
Recall@20	0.452967	0.458137	+1.14 %
NDCG@20	0.239081	0.250784	+4.89 %

Найбільше покращення для GRU4Rec спостерігається на $NDCG@5$. Це означає, що LLM-сигнал особливо корисний для перестановки верхніх позицій, а не лише для ширшого покриття top-20.

Для LightGCN регулярна модель показала $NDCG@20 = 0.302093$ та $Recall@20 = 0.508182$. Після LLM reranking з $llm_weight = 0.30$ $NDCG@20$ зріс до 0.308880, а $Recall@20$ – до 0.509889.

У таблиці 4.7 наведено порівняння регулярної LightGCN та LightGCN з LLM reranking.

Таблиця 4.7 – LightGCN regular vs LightGCN + LLM reranking

Метрика	LightGCN	LightGCN + LLM, w = 0.30	Relative gain
Precision@5	0.065987	0.067072	+1.64 %

Продовження таблиці 4.7

Метрика	LightGCN	LightGCN + LLM, w = 0.30	Relative gain
Recall@5	0.329937	0.335358	+1.64 %
NDCG@5	0.251465	0.259348	+3.13 %
Precision@10	0.041070	0.041376	+0.75 %
Recall@10	0.410702	0.413764	+0.75 %
NDCG@10	0.277577	0.284699	+2.57 %
Precision@20	0.025409	0.025494	+0.34 %
Recall@20	0.508182	0.509889	+0.34 %
NDCG@20	0.302093	0.308880	+2.25 %

LightGCN є найсильнішою моделлю серед перевірених підходів за абсолютним значенням NDCG@20. При цьому LLM reranking додатково покращує її результат. Це показує, що LLM item similarity може бути корисною не лише для класичних item-based моделей, а й для сучасних графових рекомендаційних моделей.

LLM recent-history reranking також було перевірено для Most Popular, Item-CF і BPR-MF.

Для Most Popular LLM reranking покращив NDCG, але трохи зменшив Recall@20. Регулярна Most Popular модель мала NDCG@20 = 0.275909, а варіант із LLM reranking при `llm_weight = 0.30` – NDCG@20 = 0.290833. При цьому Recall@20 зменшився з 0.530720 до 0.530017.

У таблиці 4.8 наведено порівняння регулярної Most Popular та Most Popular з LLM reranking.

Таблиця 4.8 – Most Popular regular vs Most Popular + LLM reranking

Метрика	Most Popular	Most Popular + LLM, w = 0.30	Relative gain
Recall@5	0.296858	0.310411	+4.57 %
NDCG@5	0.209376	0.228652	+9.21 %

Продовження таблиці 4.8

Метрика	Most Popular	Most Popular + LLM, w = 0.30	Relative gain
Recall@10	0.403875	0.408242	+1.08 %
NDCG@10	0.243845	0.260123	+6.68 %
Recall@20	0.530720	0.530017	-0.13 %
NDCG@20	0.275909	0.290833	+5.41 %

Цей результат означає, що LLM reranking піднімає релевантні товари вище в списку, але може трохи погіршувати ширше top-20 покриття. Для задач, де головною метрикою є NDCG, такий ефект є позитивним.

Для BPR-MF LLM reranking дав сильне покращення на всіх метриках. Регулярна модель мала $NDCG@20 = 0.103834$, а BPR-MF + LLM reranking при $llm_weight = 0.30$ – $NDCG@20 = 0.113843$. Відносне покращення становить +9.64 %.

У таблиці 4.9 наведено порівняння регулярної BPR-MF та BPR-MF з LLM reranking.

Таблиця 4.9 – BPR-MF regular vs BPR-MF + LLM reranking

Метрика	BPR-MF	BPR-MF + LLM, w = 0.30	Relative gain
Recall@5	0.090302	0.099890	+10.62 %
NDCG@5	0.062604	0.073538	+17.47 %
Recall@10	0.143008	0.151039	+5.62 %
NDCG@10	0.079438	0.089851	+13.11 %
Recall@20	0.240337	0.246762	+2.67 %
NDCG@20	0.103834	0.113843	+9.64 %

Це свідчить, що LLM semantic reranking особливо корисний для моделей, які формують загальний latent ranking, але можуть недостатньо точно впорядковувати близькі за score кандидати.

Для Item-CF post-hoc LLM reranking виявився неефективним. Регулярний Item-CF у цьому експерименті мав $NDCG@20 = 0.247558$, тоді як найкращий варіант із LLM reranking не перевищив базову модель. При $llm_weight = 0.30$ $NDCG@20$ знизився до 0.239882, а $Recall@20$ – з 0.433541 до 0.392280.

Цей результат не суперечить покращенням Hybrid Item-CF, отриманим раніше. Він показує, що для Item-CF спосіб інтеграції LLM-сигналу є критично важливим. Якщо LLM similarity використовується всередині item-item similarity, вона покращує Item-CF. Якщо ж вона застосовується як зовнішній reranker після вже сформованих Item-CF score, результат погіршується.

4.11 Порівняльний аналіз результатів

Отримані результати показують, що різні способи використання LLM мають різну ефективність. Найменш надійним підходом є пряме генерування synthetic user-item interactions. Навіть коли синтетичні взаємодії виглядають семантично правдоподібними, вони можуть не відповідати реальній поведінці користувачів і спотворювати колаборативну структуру.

Натомість LLM-generated item-item similarity виявилася корисним синтетичним сигналом. Вона не змінює реальні user-item взаємодії, а лише додає семантичну інформацію про зв'язки між товарами. Для Item-CF найкращий результат був отриманий при інтеграції LLM similarity безпосередньо в item-item similarity. Для BPR-MF, GRU4Rec і LightGCN ефективнішим виявився recent-history reranking.

Найкращий абсолютний результат серед перевірених моделей отримано для LightGCN + LLM reranking:

- $Recall@20 = 0.509889$;
- $NDCG@20 = 0.308880$.

Серед моделей без нейронної графової архітектури сильним результатом є Most Popular + LLM reranking за $NDCG@20 = 0.290833$. Однак цей результат потрібно інтерпретувати обережно: sampled-negative evaluation може підсилювати popularity-based моделі, оскільки випадкові негативні кандидати часто є товарами з довгого хвоста, які легше відрізнити від популярного тестового товару.

GRU4Rec + LLM reranking також показав стабільне покращення порівняно з регулярною GRU4Rec. Найбільший ефект спостерігається на $NDCG@5$, що підтверджує користь LLM-сигналу саме для верхньої частини рекомендаційного списку.

BPR-MF отримав одне з найбільших відносних покращень від LLM reranking: $NDCG@20$ зріс на +9.64 %, а $NDCG@5$ – на +17.47 %. Це означає, що LLM similarity добре працює як додатковий semantic prior для моделей, які навчають латентні представлення, але не використовують явну текстову інформацію про товари.

ВИСНОВКИ

У роботі виконано повний цикл дослідження застосування LLM-згенерованих синтетичних сигналів у рекомендаційних системах: від підготовки даних і реалізації базових моделей до порівняння різних способів інтеграції LLM у рекомендаційний pipeline.

Для експериментів було підготовлено локальний набір даних Amazon Appliances, який містить реальні user-item взаємодії, оцінки, текстові відгуки та метадані товарів. Було реалізовано очищення даних, нормалізацію текстових полів, об'єднання взаємодій із метаданими та ітеративну k-core фільтрацію. Це дало змогу отримати придатний для експериментів набір даних із достатньою кількістю реальних взаємодій і семантичної інформації про товари.

Поставлене завдання щодо реалізації та порівняння рекомендаційних моделей виконано. У роботі було досліджено Most Popular, Item-CF, BPR-MF, LightGCN і GRU4Rec. Для оцінювання застосовано leave-one-out схему та метрики Precision@K, Recall@K, HitRate@K і NDCG@K. Це дало змогу порівняти як класичні, так і нейронні підходи в однакових умовах.

Пряма генерація синтетичних user-item interactions за допомогою LLM виявилася ненадійною. Початковий підхід призвів до popularity collapse: LLM концентрувала генерацію на невеликій кількості популярних товарів і створювала надмірно позитивні оцінки. Навіть після покращення candidate retrieval синтетичні взаємодії залишалися недостатньо стабільними для покращення моделей.

Основною проблемою synthetic user-item interactions є те, що LLM генерує семантично правдоподібні, але не обов'язково поведінково реалістичні взаємодії. Такі записи не враховують експозицію товару, часовий контекст, ціну, доступність, реальний намір користувача та інші фактори, які визначають фактичну взаємодію. Через це їх додавання до train matrix може спотворювати колаборативну структуру.

Найбільш ефективним типом LLM-згенерованого сигналу виявилася item-item semantic similarity. На відміну від синтетичних взаємодій, вона не підміняє реальну поведінку користувачів, а лише доповнює модель семантичною інформацією про схожість або функціональний зв'язок між товарами.

Для Item-CF найкращим способом використання LLM стала інтеграція LLM item similarity безпосередньо у формулу item-item similarity. При 1000 source items і $llm_weight = 0.30$ NDCG@20 зріс з 0.070771 до 0.073844, тобто на +4.34%. Найбільший ефект спостерігався для NDCG@5, де приріст становив +9.85%. Це свідчить, що LLM-сигнал найбільше покращує верхні позиції рекомендаційного списку.

Збільшення кількості товарів, для яких генерувалася LLM item similarity, позитивно вплинуло на результати. Перехід від 100 до 500 і 1000 source items послідовно збільшував покращення Hybrid Item-CF. Це підтверджує, що для практичного використання такого підходу важливим є покриття item-similarity graph.

Експерименти з різним розміром user subset показали, що LLM item similarity особливо корисна в умовах більшої розрідженості даних. Для менших підмножин користувачів відносний приріст був вищим, оскільки колаборативного сигналу було менше, а семантичний item-item сигнал частково компенсував цю нестачу.

Для BPR-MF, GRU4Rec і LightGCN ефективним виявився інший спосіб інтеграції LLM – recent-history reranking. У цьому підході базова модель навчається лише на реальних взаємодіях, а LLM item similarity використовується для коригування порядку кандидатів відповідно до їхньої близькості до останніх товарів в історії користувача.

LLM recent-history reranking покращив результати сильних моделей. Для GRU4Rec NDCG@20 зріс з 0.239081 до 0.250784, для LightGCN – з 0.302093 до 0.308880, а для BPR-MF – з 0.103834 до 0.113843. Найбільші

відносні прирости зазвичай спостерігалися на $NDCG@5$, що підтверджує користь LLM-сигналу саме для покращення top-ranked рекомендацій.

Водночас було виявлено, що один і той самий LLM-підхід не є універсальним для всіх моделей. Post-hoc LLM reranking погіршив Item-CF, хоча внутрішня інтеграція LLM item similarity у саму Item-CF модель покращувала результати. Отже, спосіб використання LLM-сигналу має залежати від архітектури рекомендаційної моделі.

Найкращий абсолютний результат серед досліджених моделей отримано для LightGCN + LLM recent-history reranking. Цей результат показує, що LLM-сигнал може бути корисним не лише для класичних моделей, а й для сучасних нейронних рекомендаційних систем. Проте він виступає не заміною основної моделі, а допоміжним semantic reranking signal.

Окремим обмеженням дослідження є використання sampled-negative evaluation. У такій схемі популярні товари можуть отримувати перевагу, оскільки випадкові негативні кандидати часто належать до long-tail частини каталогу. Через це результати Most Popular і моделей, чутливих до популярності, потрібно інтерпретувати обережно.

Ще одним обмеженням є залежність ефективності LLM item similarity від якості метаданих товарів і покриття каталогу. Якщо товар має короткий або неінформативний опис, LLM може створити слабший або менш корисний semantic signal. Також обмежена кількість source items зменшує ймовірність того, що LLM-сигнал буде використаний під час ранжування.

Практична рекомендація за результатами роботи полягає в тому, що LLM не варто використовувати для масового створення штучних user-item interactions без додаткової перевірки. Надійнішим є використання LLM для генерації item-item semantic similarity, semantic reranking або регуляризації item embeddings.

Для Item-CF рекомендовано використовувати LLM item similarity всередині item-item similarity function. Для BPR-MF, GRU4Rec і LightGCN

доцільно використовувати LLM recent-history reranking із невеликим або помірним значенням ваги. У проведених експериментах найкращі результати часто давав $llm_weight = 0.30$, але це значення слід підбирати окремо для конкретного набору даних і метрики.

Для подальшого розвитку роботи доцільно перевірити запропонований підхід на інших категоріях Amazon Reviews, використати повнішу негативну вибірку або full-ranking evaluation, дослідити статистичну значущість приростів, а також реалізувати training-level підхід, у якому LLM item similarity використовується для регуляризації item embeddings.

Загальний результат дослідження полягає в тому, що LLM-згенерований синтетичний сигнал може покращувати рекомендаційні системи, але лише за умови правильного формулювання задачі. Найкращі результати отримано не від генерації штучної поведінки користувачів, а від використання LLM як джерела додаткового семантичного item-item сигналу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Recommender systems handbook / ed. by F. Ricci, L. Rokach, B. Shapira. New York, NY : Springer US, 2022. URL: <https://doi.org/10.1007/978-1-0716-2197-4> (date of access: 12.05.2026).
2. Item-based collaborative filtering recommendation algorithms / B. Sarwar et al. *The tenth international conference*, Hong Kong, Hong Kong, 1–5 May 2001. New York, New York, USA, 2001. URL: <https://doi.org/10.1145/371920.372071> (date of access: 12.05.2026).
3. Koren Y., Bell R., Volinsky C. Matrix factorization techniques for recommender systems. *Computer*. 2009. Vol. 42, no. 8. P. 30–37. URL: <https://doi.org/10.1109/mc.2009.263> (date of access: 12.05.2026).
4. BPR: bayesian personalized ranking from implicit feedback / S. Rendle et al. *arXiv*. 2012. URL: <https://doi.org/10.48550/arXiv.1205.2618> (date of access: 12.05.2026).
5. Neural collaborative filtering / X. He et al. *WWW '17: 26th international world wide web conference*, Perth Australia. Republic and Canton of Geneva, Switzerland, 2017. URL: <https://doi.org/10.1145/3038912.3052569> (date of access: 12.05.2026).
6. Steck H. Embarrassingly shallow autoencoders for sparse data. *The world wide web conference*, San Francisco, CA, USA, 13–17 May 2019. New York, New York, USA, 2019. URL: <https://doi.org/10.1145/3308558.3313710> (date of access: 12.05.2026).
7. Ning X., Karypis G. SLIM: sparse linear methods for top-n recommender systems. *2011 IEEE 11th international conference on data mining (ICDM)*, Vancouver, BC, Canada, 11–14 December 2011. 2011. URL: <https://doi.org/10.1109/icdm.2011.134> (date of access: 12.05.2026).
8. LightGCN / X. He et al. *SIGIR '20: The 43rd International ACM SIGIR conference on research and development in Information Retrieval*, Virtual Event

China. New York, NY, USA, 2020.

URL: <https://doi.org/10.1145/3397271.3401063> (date of access: 12.05.2026).

9. Session-based recommendations with recurrent neural networks / B. Hidasi et al. *arXiv*. 2015.

URL: <https://doi.org/10.48550/arXiv.1511.06939> (date of access: 12.05.2026).

10. Kang W.-C., McAuley J. Self-Attentive sequential recommendation. *2018 IEEE international conference on data mining (ICDM)*, Singapore, 17–20 November 2018. 2018.

URL: <https://doi.org/10.1109/icdm.2018.00035> (date of access: 12.05.2026).

11. Evaluating collaborative filtering recommender systems / J. L. Herlocker et al. *ACM transactions on information systems*. 2004. Vol. 22, no. 1. P. 5–53. URL: <https://doi.org/10.1145/963770.963772> (date of access: 12.05.2026).

12. Shani G., Gunawardana A. Evaluating recommendation systems. *Recommender systems handbook*. Boston, MA, 2010. P. 257–297. URL: https://doi.org/10.1007/978-0-387-85820-3_8 (date of access: 12.05.2026).

13. Ni J., Li J., McAuley J. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, Hong Kong, China. Stroudsburg, PA, USA, 2019. URL: <https://doi.org/10.18653/v1/d19-1018> (date of access: 12.05.2026).

14. Amazon reviews'23. *Amazon Reviews'23*. URL: <https://amazon-reviews-2023.github.io> (date of access: 12.05.2026).

15. Large language models for generative recommendation: a survey and visionary discussions / L. Li et al. *arXiv*. 2023. URL: <https://doi.org/10.48550/arXiv.2309.01157> (date of access: 12.05.2026).

16. Towards next-generation llm-based recommender systems: a survey and beyond / Q. Wang et al. *arXiv*. 2024. URL: <https://doi.org/10.48550/arXiv.2410.19744> (date of access: 12.05.2026).

17. Is ChatGPT a good recommender? A preliminary study / J. Liu et al. *arXiv*. 2023. URL: <https://doi.org/10.48550/arXiv.2304.10149> (date of access: 12.05.2026).

18. LLMRec: large language models with graph augmentation for recommendation / W. Wei et al. *WSDM '24: the 17th ACM international conference on web search and data mining*, Merida Mexico. New York, NY, USA, 2024. URL: <https://doi.org/10.1145/3616855.3635853> (date of access: 12.05.2026).

19. Scikit-learn: machine learning in Python / F. Pedregosa et al. *Journal of Machine Learning Research*. 2011. Vol. 12. P. 2825–2830. URL: <https://www.jmlr.org/papers/v12/pedregosa11a.html> (date of access: 12.05.2026).

20) Chalyi S., Leshchynskyi V., Leshchynska I. Detailing explanations in the recommender system based on matching temporal knowledge. *Eastern-European journal of enterprise technologies*. 2020. Vol. 4, no. 2 (106). P. 6–13. URL: <https://doi.org/10.15587/1729-4061.2020.210013> (date of access: 12.05.2026).