Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра комп'ютерного моделювання та інтелектуальних технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження рекомендаційних систем на основі колаборативної фільтрації для оброблення великих даних
(тема)

Виконав:
здобувач 2 року навчання,
групи СПРм-24-1

Володимир Бухало
(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 Системне проєктування
(повна назва освітньої програми)

Керівник проф. кафедри КМІТ Сергій Мінухін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри КМІТ _____ Ігор Гребеннік
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра комп'ютерного моделювання та інтелектуальних технологій

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 Системне проєктування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 6 » квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бухало Володимиру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження рекомендаційних систем на основі колаборативної фільтрації для оброблення великих даних

затверджена наказом університету від 6 квітня 2026 р. № 268 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 2026 р.
3. Вихідні дані до роботи матеріали науково-дослідної практики, огляд науково технічних джерел щодо рекомендаційних систем на основі колаборативної фільтрації для оброблення великих даних, методичні вказівки до організації виконання та захисту кваліфікаційної роботи на здобуття другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки освітньо-наукова програма «Системне проєктування», інтернет ресурси щодо теми кваліфікаційної роботи.

4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної галузі, методологічне обґрунтування дослідження, експериментальне дослідження, аналіз отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	09.04.2026	*виконано*
2	Визначення методики проведення дослідження	11.04.2026	*виконано*
3	Підготовка обчислювальних середовищ, наборів даних та розробка програмного забезпечення згідно методики	15.04.2026	*виконано*
4	Проведення експериментів та збір результатів	21.04.2026	*виконано*
5	Аналіз отриманих результатів	23.04.2026	*виконано*
6	Оформлення пояснювальної записки	03.05.2026	*виконано*
7	Подання роботи на перевірку на плагіат	12.05.2026	*виконано*
8	Підготовка доповіді та оформлення презентаційних матеріалів	13.05.2026	*виконано*
9	Попередній захист	14.05.2026	*виконано*
10	Подання роботи на рецензію	14.05.2026	*виконано*
11	Подання роботи на підпис завідувачу кафедри	16.05.2026	*виконано*
12	Подання роботи до екзаменаційної комісії	17.05.2026	*виконано*

Дата видачі завдання 6 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. кафедри КМІТ Сергій Мінухін
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи: 103 с., 4 табл., 49 рис., 2 додатки, 54 джерела.

РЕКОМЕНДАЦІЙНІ СИСТЕМИ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ВЕЛИКІ ДАНІ, АЛГОРИТМ, АГРЕГУВАННЯ РАНГІВ, ОБЧИСЛЮВАЛЬНА ЕФЕКТИВНІСТЬ, ЯКІСТЬ РЕКОМЕНДАЦІЙ, НЕЯВНИЙ ЗВОРОТНИЙ ЗВ'ЯЗОК

Об'єктом дослідження є процес формування рекомендацій на основі колаборативної фільтрації в умовах оброблення великих даних.

Предметом дослідження є алгоритми колаборативної фільтрації, методи їх комбінування й метрики оцінювання рекомендацій в умовах оброблення великих даних.

Мета роботи – дослідження рекомендаційних систем на основі колаборативної фільтрації із застосуванням алгоритмів та їхніх комбінацій для оброблення великих даних.

У процесі дослідження застосовано теоретичні (формалізація, узагальнення, інтерпретація), емпірико-теоретичні (аналіз, моделювання) та емпіричні (експеримент, вимірювання, порівняння) методи.

У роботі дістало подальшого розвитку порівняльне оцінювання рекомендаційних систем на основі алгоритмів, що охоплюють використані методи, за показниками якості рекомендацій і обчислювальної ефективності, а також уперше здійснено комбінування рекомендаційних списків алгоритмів методом WRRF, що дало змогу оцінити ефективність їх поєднання в умовах оброблення великих даних.

Отримані результати можуть слугувати основою для обґрунтованого вибору досліджуваних алгоритмів при побудові рекомендаційних систем для оброблення великих даних, а також підвищення якості вже існуючих.

ABSTRACT

Master's Thesis: 103 pages, 4 tables, 49 figures, 2 appendices, 54 references.

RECOMMENDER SYSTEMS, COLLABORATIVE FILTERING, BIG DATA, ALGORITHM, RANK AGGREGATION, COMPUTATIONAL EFFICIENCY, RECOMMENDATION QUALITY, IMPLICIT FEEDBACK

The object of this study is the process of generating recommendations based on collaborative filtering in the context of big data processing.

The subject of this study is collaborative filtering algorithms, methods for combining them, and metrics for evaluating recommendations in the context of big data processing.

The aim of the work is to investigate recommendation systems based on collaborative filtering using algorithms and their combinations for big data processing.

The research employs theoretical (formalisation, generalisation, interpretation), empirical-theoretical (analysis, modelling) and empirical (experiment, measurement, comparison) methods.

The work further developed the comparative evaluation of recommendation systems based on algorithms covering the methods used, in terms of recommendation quality and computational efficiency, and for the first time combined the recommendation lists of algorithms using the WRRF method, which made it possible to assess the effectiveness of their combination in big data processing.

The results obtained can serve as a basis for the informed selection of the algorithms under study when building recommendation systems for big data processing, as well as for improving the quality of existing ones.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ALS – Alternating Least Squares (метод почергових найменших квадратів)

BiVAE – Bilateral Variational Autoencoder (двосторонній варіаційний автокодер)

BPR – Bayesian Personalized Ranking (байєсівське персоналізоване ранжування)

GPU – Graphics Processing Unit (графічний процесор)

HDFS – Hadoop Distributed File System (розподілена файлова система Hadoop)

LightGCN – Light Graph Convolutional Network (полегшена графова згорткова нейронна мережа)

MAP – Mean Average Precision (середнє значення точності)

nDCG – normalized Discounted Cumulative Gain (нормалізований зважений кумулятивний приріст релевантності)

RRF – Reciprocal Rank Fusion (обернене злиття рангів)

SAR – Simple Algorithm for Recommendation (простий алгоритм рекомендацій)

WRRF – Weighted Reciprocal Rank Fusion (зважене обернене злиття рангів)

ВСТУП

Сучасні інформаційні системи характеризуються стрімким зростанням обсягів цифрового контенту, що зумовлює потребу в автоматизованих засобах персоналізації в умовах великих даних [1]. У цьому контексті рекомендаційні системи, що реалізуються у вигляді алгоритмів [2], набувають важливого значення як інструмент персоналізованого відбору та впорядкування інформації, забезпечуючи подання найбільш доцільних варіантів відповідно до уподобань користувачів [3]. Характерною особливістю багатьох практичних застосувань рекомендаційних систем є використання неявного зворотного зв’язку [4]. Це пояснюється тим, що такі дані легше збирати, і вони доступні в набагато більшому обсязі порівняно з явними оцінками [5].

Одним із поширених підходів до побудови рекомендаційних систем є колаборативна фільтрація [6], що ґрунтується на аналізі колективної поведінки користувачів без необхідності явного опису характеристик об’єктів. Такий підхід є незалежним від предметної області, що пояснює його популярність, а також зумовлює універсальність і можливість застосування в різних прикладних задачах.

Водночас за способом побудови рекомендацій колаборативну фільтрацію поділяють на пам’яті-орієнтовані, модельно-орієнтовані, нейромережеві методи та гібридні підходи [7]. Рекомендаційні системи на їх основі відрізняються за метриками якості рекомендацій та обчислювальної ефективності [8]. Це зумовлює вирішення проблеми щодо визначення таких алгоритмів побудови рекомендаційної системи, які забезпечують покращення якості рекомендацій за меншими обчислювальними витратами за однакових умов оброблення великих даних.

Тому актуальність роботи визначається потребою в обґрунтуванні вибору алгоритмів колаборативної фільтрації для побудови рекомендаційних систем в умовах оброблення великих даних. Підставою для її виконання стали відмінності таких систем, що зумовило потребу в їх порівняльному оцінюванні за однакових умов оброблення великих даних.

Метою роботи є дослідження рекомендаційних систем на основі колаборативної фільтрації із застосуванням алгоритмів та їхніх комбінацій для оброблення великих даних.

Завдання дослідження полягає у проведенні порівняльного аналізу рекомендаційних систем, побудованих із використанням одного та кількох алгоритмів колаборативної фільтрації, в умовах оброблення великих даних за показниками якості рекомендацій та обчислювальної ефективності за різних обсягів даних і конфігурацій обчислювальних ресурсів.

Об’єктом дослідження є процес формування рекомендацій на основі колаборативної фільтрації в умовах оброблення великих даних.

Предметом дослідження є алгоритми колаборативної фільтрації, методи їх комбінування й метрики оцінювання рекомендацій в умовах оброблення великих даних.

Для досягнення мети у роботі були застосовані теоретичні (формалізація, узагальнення, інтерпретація), емпірико-теоретичні (аналіз, моделювання) та емпіричні (експеримент, вимірювання, порівняння) методи дослідження.

У роботі дістало подальшого розвитку порівняльне оцінювання рекомендаційних систем на основі алгоритмів, що охоплюють усі описані методи, за показниками якості рекомендацій і обчислювальної ефективності, а також уперше здійснено комбінування рекомендаційних списків алгоритмів методом WRRF (Weighted Reciprocal Rank Fusion), що дало змогу оцінити ефективність їх поєднання в умовах оброблення великих даних.

Практичне значення отриманих результатів полягає у можливості їх використання для обґрунтованого вибору алгоритмів та їх комбінацій при побудові рекомендаційних систем для оброблення великих даних.

За результатами виконання кваліфікаційної роботи підготовлено матеріали апробації основних результатів дослідження. У тезах конференції [9] представлено застосування методу RRF (Reciprocal Rank Fusion) для задач колаборативної фільтрації. У науковій статті [10] досліджено вплив методу WRRF на якість рекомендаційних списків на різних наборах даних і для різних алгоритмів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Рекомендаційні системи як засіб персоналізації в умовах великих даних

Стрімке зростання обсягів цифрового контенту, кількості онлайн-сервісів і доступних альтернатив ускладнює пошук релевантних об’єктів без спеціалізованих засобів підтримки вибору [11]. У таких умовах виникає інформаційне перевантаження – стан, за якого обсяг і складність інформації перевищують здатність людини до її ефективного опрацювання та прийняття обґрунтованих рішень. Рекомендаційні системи не лише відбирають контент, а й знижують когнітивне навантаження, звужуючи множину потенційно релевантних варіантів.

Рекомендаційні системи – це програмні засоби, що формують індивідуалізовані пропозиції товарів, послуг або інформаційних об’єктів на основі даних про користувача, об’єкти та їхні взаємодії. Їх призначення має два взаємопов’язані аспекти: фільтрацію інформації та персоналізацію доступу до великих масивів даних, а також підтримку прийняття рішень шляхом виявлення релевантних, у тому числі менш очевидних, об’єктів. Це зумовлює їх розгляд як у межах інформаційного пошуку, так і в контексті підтримки користувацького вибору.

В умовах використання великих даних значення рекомендаційних систем збільшується [12]. Цифрові платформи накопичують значні обсяги інформації про поведінку користувачів і водночас пропонують широкий спектр об’єктів для вибору. Дані про взаємодії з контентом забезпечують можливість автоматичного виявлення закономірностей уподобань і формування персоналізованих рекомендацій. Таким чином, великі дані одночасно ускладнюють орієнтацію в інформації та створюють підґрунтя для її ефективного впорядкування через персоналізацію. Тому сучасний стан колаборативної фільтрації тісно пов’язаний із сутністю й вмістом великих даних (рисунок 1.1) та використанням розподілених обчислень [13].

Рисунок 1.1 – Визначення великих даних

У прикладному аспекті рекомендаційні системи є важливим компонентом цифрових сервісів у сферах електронної комерції, освіти, інформаційних ресурсів, туризму та інших галузях, де необхідний швидкий вибір серед великої кількості альтернатив. Вони не лише підвищують зручність доступу до контенту, а й впливають на ефективність сервісів, оптимізують корисність рекомендацій для різних зацікавлених сторін і впливають на продажі [14].

Прикладом застосування рекомендаційних систем є музичні стримінгові сервіси, зокрема YouTube Music [15] і Spotify [16], а також стримінговий сервіс відеоконтенту Netflix [17], які використовують великі обсяги даних про взаємодії користувачів (прослуховування, вподобання, історія переглядів) для формування персоналізованих добірок контенту; аналогічно у сфері електронної комерції платформа Temu [18] застосовує дані про перегляди, покупки та взаємодії користувачів для побудови персоналізованих рекомендацій, що підвищує релевантність пропозицій.

Зв’язок рекомендаційних систем із великими даними пояснюється реальними обсягами даних, з якими вони працюють у виробничому середовищі. Прикладом є Netflix, де на певному етапі оброблялося понад 5 мільярдів оцінок користувачів [19]. При цьому щоденний приріст даних становив приблизно 4 мільйони нових оцінок

[19], що свідчить про безперервне зростання інформаційного потоку та його високу інтенсивність.

Водночас, зростають і доступні обсяги даних для дослідження рекомендаційних систем (рисунок 1.2). Наприклад, якщо у 1998 році датасет MovieLens [20] містив лише 100 тисяч рейтингів, то станом на 2024 рік його обсяг зріс до 32 мільйонів. Щодо Amazon Reviews [21], то спостерігалося зростання: приблизно з 35 мільйонів рейтингів у 2013 році до 572 мільйонів у 2023 році.

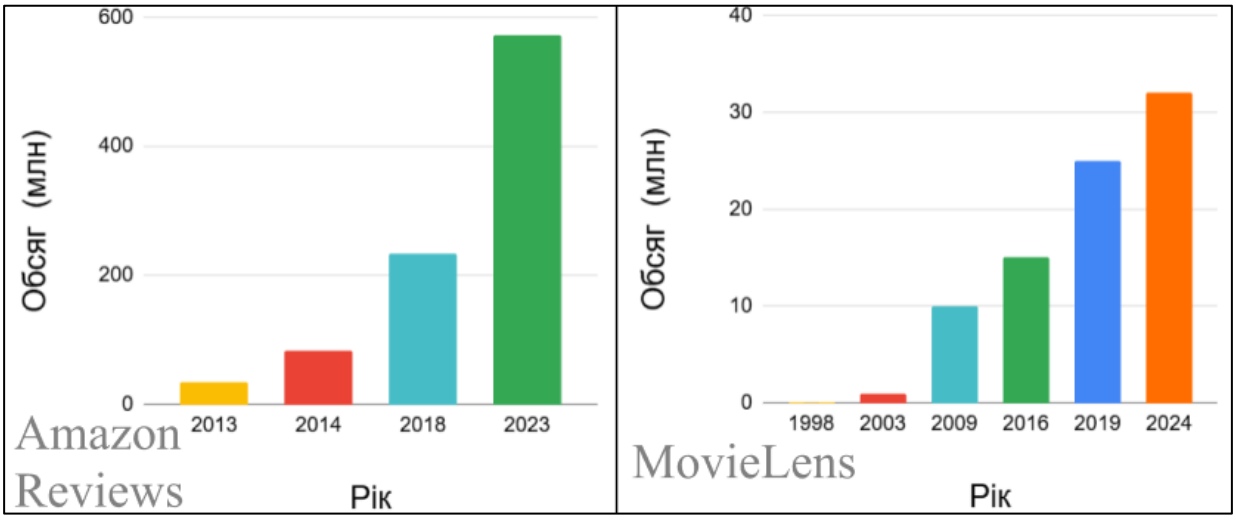

Рисунок 1.2 – Динаміка зростання обсягів даних у популярних наборах даних Amazon Reviews та MovieLens для рекомендаційних систем

Рекомендаційні системи відрізняються за структурою та підходами до реалізації. Основні напрями включають контентно-орієнтовані, колаборативні та гібридні підходи [22]. Окрім них, існують також їхні розширення та модифікації, спрямовані на врахування додаткових факторів, таких як контекст, поведінкові особливості користувачів або різні типи даних. Розширення сфер застосування та зростання обсягів і різноманітності даних зумовлюють подальший розвиток цих підходів.

Отже, рекомендаційні системи в умовах великих даних є ключовим засобом персоналізованого відбору інформації. Їх поширення зумовлене зростанням обсягів інформації та можливістю використання даних про взаємодії для формування

індивідуалізованих рекомендацій. Таким чином, вони виступають невід'ємним елементом сучасних інформаційних систем, підвищуючи якість користувацького досвіду.

1.2 Колаборативна фільтрація

Формування колаборативної фільтрації розпочалося у 1990-х роках із появою систем, у яких рекомендації будувалися на основі оцінок і дій інших користувачів (рисунок 1.3).

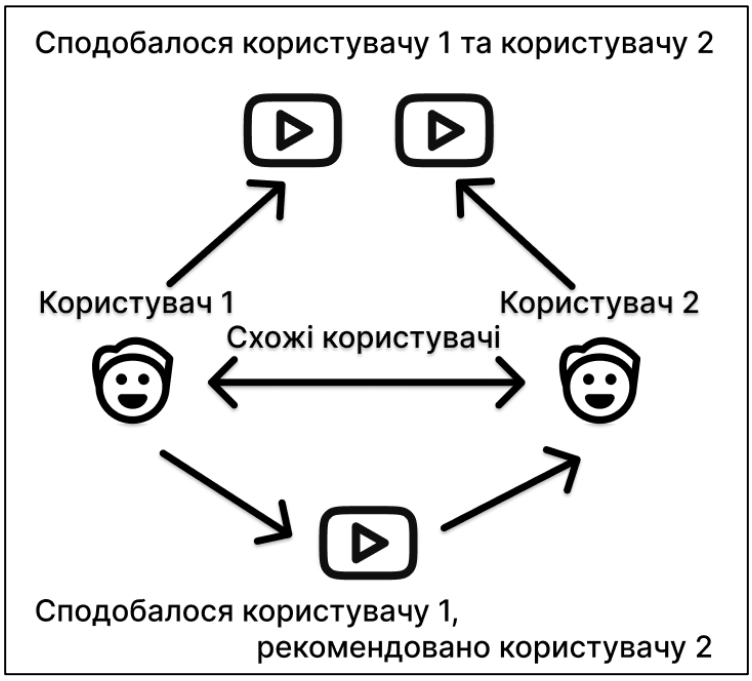

Рисунок 1.3 – Принцип роботи колаборативної фільтрації

Її сутність полягає у формуванні рекомендацій на основі взаємодій користувачів з об'єктами та закономірностей колективної поведінки: подібність у вподобаннях щодо частини об'єктів свідчить про ймовірну подібність і щодо інших. При цьому ключову роль відіграє структура зв'язків між користувачами та об'єктами, а не їхні власні характеристики.

На відміну від підходів, що спираються на описові характеристики об’єктів, колаборативна фільтрація є предметно-незалежною та орієнтується виключно на дані взаємодій, що зазвичай подаються у вигляді матриці «користувач-об’єкт». (рисунок 1.4). Це дозволяє формувати рекомендації без явного опису об’єктів, працювати з різнорідним контентом і пропонувати нові, неочевидні для користувача об’єкти, що забезпечує універсальність і широку застосовність підходу.

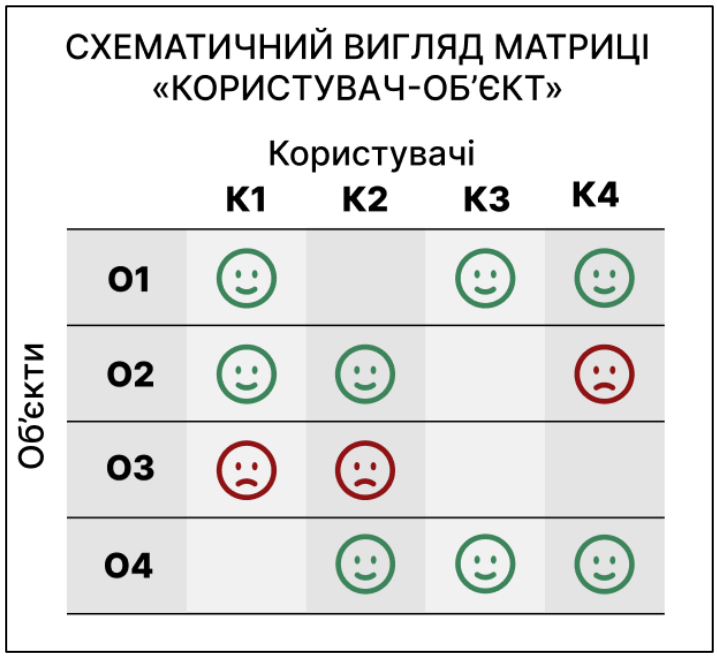

Рисунок 1.4 – Матриця «користувач-об’єкт»

Однією з основних проблем колаборативної фільтрації є «холодний старт» [6], який виникає за відсутності достатніх даних про нових користувачів або об’єкти, що ускладнює формування рекомендацій. Також характерною є висока розрідженість матриці «користувач-об’єкт», оскільки більшість можливих взаємодій відсутні, що знижує точність виявлення подібностей.

Попри свою універсальність і широке застосування, ефективність цього підходу обмежується наявністю даних та його характерними проблемами, такими як «холодний старт» і розрідженість взаємодій.

1.3 Класифікація методів колаборативної фільтрації

Розвиток колаборативної фільтрації відбувався від методів, що безпосередньо обчислюють подібність між користувачами або об’єктами, до підходів, які формують узагальнені моделі взаємодій. З часом сформувалися два основні напрями: пам’яттєво-орієнтовані методи [7], що використовують безпосередньо дані взаємодій, та модельно-орієнтовані [7], які базуються на побудові узагальнених моделей. Подальша еволюція пов’язана з використанням нейромережевих методів [7], здатних враховувати складні залежності в даних, а також із розвитком гібридних підходів, у яких колаборативна фільтрація поєднується з іншими підходами рекомендацій.

Пам’яттєво-орієнтовані методи ґрунтуються на використанні матриці взаємодій без побудови узагальнювальної моделі. Рекомендації формуються через пошук подібних користувачів або об’єктів і подальше оцінювання на основі локального оточення. Ці методи відзначаються простотою реалізації, прозорістю та можливістю інтерпретації результатів. Наприклад, колаборативна фільтрація заснована на користувачах пропонує об’єкти, які сподобались схожим користувачам, а колаборативна фільтрація заснована на об’єктах – схожі об’єкти до тих, які вже подобались користувачеві (рисунок 1.5).

Модельно-орієнтовані методи передбачають побудову узагальнювальної моделі на основі наявних взаємодій, яка використовується для прогнозування або впорядкування. До цього класу належать статистичні та машинні підходи, серед яких ключову роль відіграють латентно-факторні моделі. Вони дозволяють виявляти приховані закономірності у вподобаннях користувачів і ефективно працювати з частково спостережуваними даними. Порівняно з пам’яттєво-орієнтованими методами, ці підходи краще виявляють глобальні структури, але потребують навчання моделі, налаштування параметрів і зазвичай мають нижчу інтерпретованість.

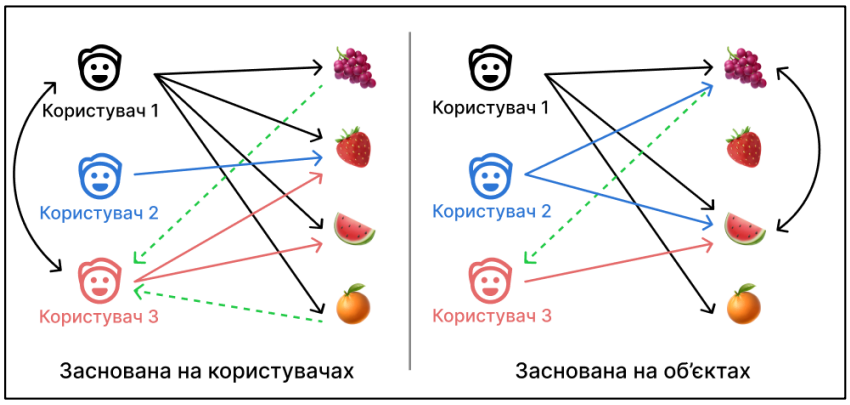

Рисунок 1.5 – Колаборативна фільтрація заснована на користувачах та об’єктах

Нейромережеві методи розширюють можливості модельно-орієнтованого підходу за рахунок використання багатошарових архітектур для моделювання складних нелінійних залежностей між користувачами та об’єктами. Вони забезпечують більшу гнучкість у представленні взаємодій і можуть інтегрувати різнорідні джерела даних [7]. Водночас такі методи висувають підвищені вимоги до обчислювальних ресурсів, обсягу навчальних даних і складності налаштування.

Гібридні підходи базуються на поєднанні різних підходів рекомендацій для компенсації їхніх обмежень. Комбінація може здійснюватися між різними рекомендаційними підходами, тоді як у межах колаборативної фільтрації вона виступає одним із компонентів такої комбінації. Це дозволяє підвищити якість і стійкість рекомендацій, однак супроводжуються ускладненням архітектури, процесів налаштування та оцінювання. Такі системи розрізняються за способами гібридизації [23], опис яких наведено в таблиці 1.1.

Таблиця 1.1 – Методи гібридизації систем рекомендацій

Метод гібридизації	Опис
Зважений [23]	Оцінки декількох підходів рекомендацій об’єднуються для формування єдиної рекомендації
Перемикання [23]	Система перемикається між підходів рекомендацій залежно від поточної ситуації
Змішаний [23]	Рекомендації від декількох різних рекомендаційних систем подаються одночасно

Кінець таблиці 1.1

Комбінація ознак [23]	Особливості з різних джерел даних рекомендацій об’єднуються в єдиний алгоритм рекомендацій
Каскад [23]	Одна рекомендаційна система уточнює рекомендації, надані іншою
Розширення ознак [23]	Результати одного підходу використовуються як вхідні дані для іншого
Метарівень [23]	Модель, вивчена одним рекомендаційним алгоритмом, використовується як вхідні дані для іншого

Таким чином, різні методи колаборативної фільтрації відрізняються способом використання даних взаємодій: від безпосереднього аналізу подібностей до побудови узагальнених моделей і складніших архітектур. Це визначає відмінності у здатності виявляти локальні або глобальні закономірності, вимог до навчання та обчислювальних ресурсів.

1.4 Типи взаємодій у рекомендаційних системах колаборативної фільтрації

Взаємодії у рекомендаційних системах на основі колаборативної фільтрації поділяються на явні і неявні. Явний зворотний зв’язок передбачає безпосереднє вираження оцінки користувачем (наприклад, рейтинги або відгуки), тоді як неявний формується на основі спостережуваних дій, що опосередковано відображають зацікавленість. До таких дій належать перегляди, кліки, покупки та інші взаємодії. Через обмеженість явних оцінок у реальних системах неявний зворотний зв’язок часто є основним джерелом даних [4].

Його практична цінність полягає в автоматичному накопиченні під час використання цифрових сервісів без додаткових дій користувача. Це забезпечує значно більші обсяги даних, що робить неявний зворотний зв’язок більш ефективним для систем, які працюють у середовищі великих даних.

Водночас такі дані не є точним відображенням уподобань, оскільки фіксують лише факт взаємодії без явного розмежування позитивних і негативних оцінок. Зокрема, відсутність взаємодії не може інтерпретуватися як негативний сигнал і може бути зумовлена різними зовнішніми факторами. Крім того, дані взаємодій містять шум і піддаються впливу популярності об’єктів та умов їх відображення, що призводить до систематичних викривлень.

У контексті великих даних ці особливості посилюються високою розрідженістю матриці взаємодій і масштабністю простору користувачів та об’єктів. Це ускладнює аналіз даних і зумовлює необхідність використання алгоритмів, здатних ефективно працювати з великими розрідженими даними.

Зазначені властивості впливають на постановку задачі рекомендації, де замість прогнозування оцінок застосовується персоналізоване впорядкування, а оцінювання ускладняється залежністю даних від попередніх рішень системи.

Отже, неявний зворотний зв’язок є основним джерелом даних у рекомендаційних системах великих масштабів, однак його використання пов’язане з неоднозначністю інтерпретації та наявністю викривлень, що потребує врахування при побудові моделей.

1.5 Методи колаборативної фільтрації для великих даних

Застосування методів колаборативної фільтрації для великих даних характеризується суттєвим зростанням обсягів інформації, розмірності простору користувачів і об’єктів, а також кількості взаємодій між ними [12]. У таких умовах визначальними стають властивості масштабованості, обчислювальної ефективності та здатності алгоритмів працювати з розрідженими даними. Різні методи колаборативної фільтрації по-різному реагують на ці виклики, що визначає їх придатність до використання у великомасштабних системах [7].

Перевагою пам’яттєво-орієнтованих методів є простота, однак зі зростанням кількості користувачів і об’єктів обчислення подібностей стає витратним, а розрідженість знижує надійність таких оцінок [7]. Тому у великих даних ці методи мають обмежену масштабованість, хоча можуть застосовуватися з додатковою оптимізацією.

Модельно-орієнтовані методи є більш придатними до великомасштабного використання, оскільки будують компактну модель взаємодій. Зокрема, латентно-факторні підходи зменшують розмірність подання користувачів і об’єктів, що підвищує ефективність обчислень [22]. Як і інші методи колаборативної фільтрації, вони можуть бути реалізовані у паралельних і розподілених середовищах, однак саме їх структура забезпечує більш ефективне масштабування при обробці великих обсягів даних [17].

Нейромережеві методи дозволяють моделювати складні нелінійні залежності між користувачами та об’єктами та за наявності великих обсягів даних можуть забезпечувати високу якість рекомендацій [22]. Водночас вони характеризуються підвищеною складністю навчання і значними вимогами до ресурсів, що ускладнює їх використання у великомасштабних системах.

Окремо слід виділити графові алгоритми, які подають взаємодії у вигляді графа «користувач-об’єкт» і враховують багатокрокові зв’язки між його вершинами. Вони можуть належати як до нейромережевих, так і до модельно-орієнтованих підходів. Такі алгоритми є перспективними для великих даних, оскільки дозволяють ефективно враховувати структурні залежності у взаємодіях [6]. Водночас їх застосування потребує оптимізації обчислень і ефективного представлення графа для забезпечення масштабованості.

У практичних системах важливим є баланс між якістю рекомендацій і обчислювальною ефективністю. Висока точність моделі є недостатньою без забезпечення прийнятного часу навчання, оновлення та відповіді. У великомасштабних середовищах обмеження ресурсів і вимоги до низької затримки роблять необхідним компроміс між точністю, швидкодією, вартістю обчислень і частотою оновлення моделей.

Отже, пам’яттєво-орієнтовані методи мають обмежену масштабованість, модельно-орієнтовані краще пристосовані до обробки великих даних, а нейромережеві підходи забезпечують ширші можливості моделювання складних залежностей за рахунок підвищених обчислювальних витрат.

1.6 Обґрунтування прийнятих припущень

Виявлені залежності між обчислювальною ефективністю алгоритмів і об’ємами даних зберігатимуться при подальшому масштабуванні. Обґрунтуванням цього є те, що обчислювальні характеристики алгоритмів визначаються їхньою внутрішньою структурою та складністю, які не змінюються зі зростанням обсягу даних, а лише масштабуються. Це узгоджується з підходами до аналізу великих даних, де зазначається, що обробка стає складною саме через обсяг даних [12], тоді як базові принципи алгоритмів залишаються незмінними.

Таким чином, якщо в межах досліджуваних умов алгоритм демонструє певну тенденцію зміни часу виконання, можна очікувати збереження цієї тенденції і для більших обсягів даних. Це дозволить використовувати результати експериментів, отриманих на обмежених наборах даних, для формування висновків саме щодо обчислювальної ефективності рекомендаційних систем у масштабніших середовищах, що є доцільним з огляду на обмеженість доступних обчислювальних ресурсів. Подібне твердження також підтверджується в сучасних дослідженнях, де зазначається збереження тенденції масштабування алгоритмів рекомендацій при збільшенні обсягів даних [12].

Додатково передбачається, що вплив горизонтального та вертикального масштабування на продуктивність також залишатиметься подібним і при зміні обсягу обчислювальних ресурсів, тобто спостережувана в дослідженні тенденція (зростання, зниження або стабільність продуктивності) збережеться і в реальних умовах, зокрема при роботі з великими даними. Це пояснюється тим, що при зміні

кількості вузлів розподіленої системи або окремих обчислювальних потужностей змінюється лише конфігурація ресурсів [13], тоді як принцип роботи алгоритмів, що їх використовують, та їхня обчислювальна складність залишаються незмінними.

Крім того, експериментальні набори даних можуть відтворювати властивості великих даних [8] за умови збереження їхніх структурних характеристик. Зокрема, це стосується властивостей розрідженості матриці «користувач–об’єкт» та співвідношення між кількістю користувачів і об’єктів, які не залежать безпосередньо від абсолютного розміру даних.

Обґрунтування цього полягає в тому, що якісні характеристики даних визначають доступність інформації для алгоритмів і впливають на результати рекомендацій. За умови збереження цих характеристик можливо відтворити типові умови функціонування алгоритмів колаборативної фільтрації навіть на обмежених наборах даних, що забезпечує коректність дослідження без використання повномасштабних обсягів даних та дає змогу формувати обґрунтовані висновки щодо якості рекомендацій у масштабніших середовищах.

Подібний підхід використання експериментальної вибірки, навіть якщо реальна рекомендаційна система оперує значно більшими обсягами даних, підтверджується практикою проведення конкурсу Netflix Prize [17], у межах якого експерименти здійснювалися на наборі даних обсягом близько 100 млн оцінок, тоді як реальна виробнича система Netflix на той момент оперувала значно більшими обсягами даних – мільярдами оцінок [19], що за своїм масштабом відповідає характеристикам великих даних.

1.7 Постановка задачі

Для досягнення поставленої мети необхідно реалізувати алгоритми колаборативної фільтрації в межах рекомендаційної системи та провести дослідження їх обчислювальної ефективності та якості рекомендацій в умовах

оброблення великих даних, що характеризуються високою розрідженістю даних та необхідністю масштабування обчислень.

Для цього необхідно вирішити такі завдання:

– обґрунтувати вибір наборів даних, алгоритмів, розподіленого середовища, стратегії комбінування алгоритмів, а також показників якості формування й упорядкування рекомендацій та обчислювальної ефективності алгоритмів;

– розробити методику проведення дослідження;

– виконати підготовку наборів даних та обчислювальних середовищ;

– реалізувати обрані алгоритми, їх попарну та повну комбінацію в межах рекомендаційної системи та інструменти для їх оцінювання;

- провести дослідження шляхом тестування розроблених систем на визначених наборах даних у обчислювальних середовищах;

– проаналізувати отримані результати.

Отримані результати дадуть змогу остаточно вирішити завдання дослідження шляхом порівняння їх між собою та надати обґрунтовану оцінку ефективності розроблених рекомендаційних систем колаборативної фільтрації для оброблення великих даних.

2 МЕТОДОЛОГІЧНЕ ОБҐРУНТУВАННЯ ДОСЛІДЖЕННЯ

2.1 Характеристика напрямку дослідження

Напрямок дослідження визначається необхідністю порівняльного аналізу рекомендаційних систем через дослідження алгоритмів, на основі яких вони будуються. Це зумовлено тим, що саме алгоритми колаборативної фільтрації безпосередньо визначають спосіб формування рекомендацій, характер обробки взаємодій користувачів з об’єктами, а також показники якості рекомендацій і обчислювальної ефективності системи.

Використання декількох алгоритмів пояснюється тим, що їх алгоритмічні підходи реалізують відмінні принципи побудови рекомендацій, по-різному реагують на структурні характеристики даних, а також характеризуються різною здатністю до масштабування та придатністю до розподіленої обробки. Додатково, через відмінності у принципах оцінювання релевантності об’єктів різні алгоритми, зазвичай, формують для одного й того самого користувача різні списки рекомендацій – як за складом, так і за порядком об’єктів [24].

Тому для забезпечення повноти дослідження доцільно охопити різні методи колаборативної фільтрації, а гібридний підхід розглядати не як поєднання різних рекомендаційних підходів, а як комбінування алгоритмів у межах одного підходу через концепцію гібридизації, що дозволяє дослідити як окремі алгоритми, так і результати їх поєднання без виходу за межі колаборативної фільтрації.

Попарне комбінування алгоритмів обрано через те, що таке поєднання забезпечує найбільшу кількість різних комбінацій за фіксованої множини алгоритмів. Це дозволяє дослідити ширший спектр варіантів рекомендаційних систем та оцінити вплив поєднання алгоритмів на якість сформованих рекомендацій. Крім того, попарне комбінування дає змогу ізольовано аналізувати взаємодію між конкретними алгоритмами, що дозволяє виявити їхню компліментарність.

Повну комбінацію алгоритмів обрано з метою порівняння з попарною, щоб визначити, чи достатньо використання мінімальної кількості різних рекомендаційних списків для досягнення суттєвого покращення якості, або доцільно залучати більшу кількість алгоритмів, що може забезпечити кращий результат, але ціною погіршення обчислювальної ефективності. Це особливо важливо в умовах великих даних, коли кожен алгоритм потребує окремого навчання та формування списків.

Використання неявного зворотного зв'язку пояснюється тим, що він є простішим для збору та доступним у значно більших обсягах порівняно з явними рейтингами, і частіше трапляється в умовах великих даних. Використання кількох різних наборів даних дозволяє оцінити роботу алгоритмів та їх комбінацій у різних умовах з урахуванням різноманітності користувачів і об'єктів.

Оцінювання якості алгоритмів з позиції формування та впорядкування рекомендаційних списків зумовлено використанням неявного зворотного зв'язку, де відсутня пряма числова оцінка релевантності об'єктів. Крім того, у реальних рекомендаційних системах результати подаються саме у вигляді упорядкувань, тоді як оцінки або подібності виступають як проміжні величини для їх формування.

Оцінювання обчислювальної ефективності обрано внаслідок того, що при роботі з великими наборами даних воно є необхідним для визначення, які серед обраних алгоритмів є більш ефективними з точки зору обчислювальних витрат.

Додаткове виконання дослідження у розподіленій конфігурації зумовлене тим, що для задач оброблення великих даних важливими є не лише якість рекомендацій, а й поведінка алгоритмів в умовах розподілу даних і обчислень. Хоча експериментальне середовище розгортатиметься локально, використання засобів, що відтворюють роботу розподіленої системи, дасть змогу дослідити особливості масштабування, розподіленого виконання та зміни обчислювальної ефективності алгоритмів у наближених до практичного застосування умовах.

Розробка єдиного експериментального сценарію зумовлена необхідністю забезпечення відтворюваності процесу формування рекомендацій та коректної порівнюваності результатів. Використання узгоджених умов проведення

експериментів дозволяє виключити вплив сторонніх факторів і зосередитися безпосередньо на властивостях алгоритмів та їх комбінацій.

2.2 Вибір та обґрунтування тестових наборів даних

Для проведення дослідження обрано 4 набори даних: MovieLens [20], Amazon Reviews 2023 Movies and TV (Movies and TV) [21], Amazon Reviews 2023 Video Games (Video Games) [21] та Jester [25]. Їх вибір зумовлений відкритістю та достатньою кількістю записів взаємодій для проведення експериментів. Крім того, ці набори даних суттєво відрізняються за масштабом, активністю користувачів, популярністю об’єктів та рівнем розрідженості у їх вихідному представленні, що є однією з ключових властивостей великих даних у рекомендаційних системах.

У перших 3-х наборах даних кожний запис подано у вигляді впорядкованої четвірки: ідентифікатори користувача та об’єкта, рейтинг та часова мітка.

Поряд з цим, між ними є певні відмінності:

– у вихідному вигляді наборів даних ідентифікатори в обох наборах даних Amazon Reviews 2023 мають символьний формат, тоді як у MovieLens вони подані у вигляді цілих чисел;

– у наборах Amazon Reviews 2023 значення рейтингу подано у форматі чисел з плаваючою крапкою, тоді як у MovieLens вони представлені цілими числами; час взаємодії в усіх наборах даних подано у вигляді Unix-мітки;

– у вихідній структурі файлів наборах даних Amazon Reviews 2023 файли містять рядок заголовків із назвами полів, тоді як у MovieLens – відсутній;

– набір даних Jester подано у вигляді матриці «користувач-об’єкт»; дані не містять часових міток, а оцінки подано в іншій шкалі та містять значення 99, що характеризує відсутність оцінки в матриці.

Таким чином, описані відмінно**сті** в обраних наборах даних зумовлюють необхідність попередньої обробки, що повинна включати: приведення

ідентифікаторів і типів змінних до узгодженого формату; приведення структури файлів до єдиного вигляду, а також очищення та фільтрацію даних. Крім того, з огляду на постановку задачі, значення рейтингів у подальшому повинні інтерпретуватися як неявний зворотний зв'язок шляхом виділення позитивних взаємодій, що також належатиме до етапу підготовки даних. У результаті будуть отримані такі дані, що забезпечать порівнюваність результатів дослідження.

2.3 Вибір та обґрунтування алгоритмів рекомендацій

Для проведення дослідження було обрано 6 алгоритмів рекомендацій: ALS (Alternating Least Squares) [26] в конфігурації неявного зворотного зв'язку [4], BPR (Bayesian Personalized Ranking) [5], BiVAE (Bidirectional Variational Autoencoder) [27], LightGCN (Light Graph Convolutional Network) [28], NCF [29] (Neural Collaborative Filtering) та SAR (Simple Algorithm for Recommendation) [30] у SARPlus реалізації [30]. Такий набір алгоритмів сформовано з урахуванням напрямку дослідження, відповідно до якого необхідно охопити різні методи колаборативної фільтрації.

Модельно-орієнтовані методи представлені алгоритмами ALS та BPR, нейромережеві – BiVAE, NCF і LightGCN, а пам'яттєво-орієнтовані – алгоритмом SAR. Алгоритм LightGCN належить до графових нейронних моделей, у яких взаємодії користувачів та об'єктів подаються у вигляді двочасткового графа. Використання алгоритмів, що відрізняються за принципами побудови рекомендацій, зумовлює доцільність їх комбінування в межах концепції гібридизації для дослідження впливу такого поєднання на якість рекомендацій.

Алгоритм ALS обрано як базовий представник моделей матричної факторизації. У цьому підході матриця взаємодій користувачів і об'єктів апроксимується добутком двох матриць латентних ознак: одна з них описує користувачів, а інша – об'єкти [26]. Навчання виконується почерговою

оптимізацією цих двох представлень методом найменших квадратів, завдяки чому алгоритм добре масштабується в розподіленому середовищі та придатний для обробки великих наборів даних. Для сценарію неявного зворотного зв'язку ALS інтерпретує наявні взаємодії як позитивні сигнали з різним рівнем довіри, тоді як відсутні взаємодії розглядаються як нульова перевага з низьким рівнем довіри. У результаті алгоритм формує компактне латентне представлення користувачів і об'єктів, на основі якого обчислюється оцінка релевантності невзаємодіяних об'єктів. Використання ALS є доцільним з огляду на його поширеність у задачах масштабованих рекомендаційних систем та придатності до розподілених обчислень.

Алгоритм BPR також належить до латентно-факторних методів, проте відрізняється цільовою постановкою задачі. Якщо ALS орієнтований на апроксимацію матриці взаємодій, то BPR безпосередньо оптимізує попарне впорядкування об'єктів. У процесі навчання для кожного користувача розглядаються трійки, що складаються з користувача, об'єкта з наявною взаємодією та об'єкта без зафіксованої взаємодії [5]. Мета навчання полягає в тому, щоб позитивний об'єкт отримував вищу оцінку, ніж об'єкт без взаємодії. Такий підхід добре узгоджується з природою неявного зворотного зв'язку, де відсутність взаємодії не може інтерпретуватися як достовірно негативна оцінка.

Алгоритм BiVAE було включено до дослідження як представника ймовірнісних нейронних моделей. У ньому латентне представлення задається у вигляді розподілу [27], а його модель є двонаправленою, оскільки симетрично враховує як представлення користувачів, так і об'єктів. Така побудова є особливо важливою для розріджених даних, оскільки дає змогу моделювати невизначеність у латентному просторі та зменшувати чутливість до нестачі спостережень. Під час навчання BiVAE поєднує реконструкцію матриці взаємодій користувачів і об'єктів із регуляризацією латентних розподілів, а після завершення навчання використовується для впорядкування кандидатів за прогнозованою релевантністю.

Використання цього алгоритму дає змогу включити до порівняння сучасний нейронний підхід, орієнтований на роботу з розрідженими даними взаємодій.

Алгоритм NCF було обрано як нейронне узагальнення матричної факторизації. Його головна ідея полягає у заміні простої взаємодії латентних векторів користувача й об’єкта на параметризовану нелінійну функцію [29]. Архітектурно NCF поєднує дві складові: узагальнену матричну факторизацію, яка моделює лінійні компоненти взаємодії, та багатошаровий перцептрон, який дає змогу виявляти складні нелінійні закономірності. Після об’єднання цих двох компонентів модель навчається прогнозувати релевантність об’єктів для користувача. Такий підхід є доцільним тоді, коли припускається, що структура переваг користувачів не може бути адекватно описана лише білінійною моделлю.

Алгоритм LightGCN представляє графовий напрям у колаборативній фільтрації. У цій моделі всі взаємодії користувачів та об’єктів подаються у вигляді двочасткового графа, вершинами якого є користувачі та об’єкти, а ребрами – зафіксовані позитивні взаємодії. На відміну від складніших графових нейронних мереж, LightGCN використовує спрощену схему поширення інформації по графу без додаткових нелінійних перетворень і без вагових матриць перетворення ознак на кожному рівні [28]. У результаті латентні представлення вершин формуються шляхом ітеративного агрегування представлень сусідів, що дає змогу безпосередньо враховувати структуру колаборативних зв’язків. Така побудова є придатною для рекомендаційних задач, оскільки подібність користувачів і об’єктів визначається не лише прямими, а й багатокроковими зв’язками в графі взаємодій.

Алгоритм SAR – це пам’яттєво-орієнтований метод рекомендацій на основі подібності об’єктів [30]. Його робота ґрунтується на припущенні, що об’єкти, з якими часто взаємодіють одні й ті самі користувачі, мають бути близькими у просторі рекомендацій. На першому етапі для всіх об’єктів обчислюється матриця подібності на основі зустрічальності, а на другому – для кожного користувача визначається оцінка релевантності об’єктів з урахуванням його попередньої історії взаємодій. Далі ці дві складові поєднуються для отримання підсумкових оцінок.

Оскільки класичні пам’яттєво-орієнтовані методи колаборативної фільтрації зазвичай характеризуються обмеженою масштабованістю, у роботі використано реалізацію алгоритму SAR у вигляді SARplus – оптимізованої версії, адаптованої

для роботи в середовищі Apache Spark [31], яке описується як уніфікований аналітичний рушій для обробки великих обсягів даних. Завдяки цьому, а також реалізації обчислювального ядра мовою програмування C++ і використанню механізмів спільного доступу до пам’яті для зберігання матриці подібності між об’єктами, такий підхід значною мірою нівелює проблему масштабованості.

Таким чином, обраний склад алгоритмів забезпечує охоплення різних методів колаборативної фільтрації та створює основу для їх подальшого комбінування і порівняння.

2.4 Вибір розподіленого середовища для оброблення великих даних

Для дослідження впливу розподілених середовищ на рекомендаційні системи як обчислювальне середовище було обрано Apache Spark (рисунок 2.1), який забезпечує виконання обчислень у кластерній архітектурі [32].

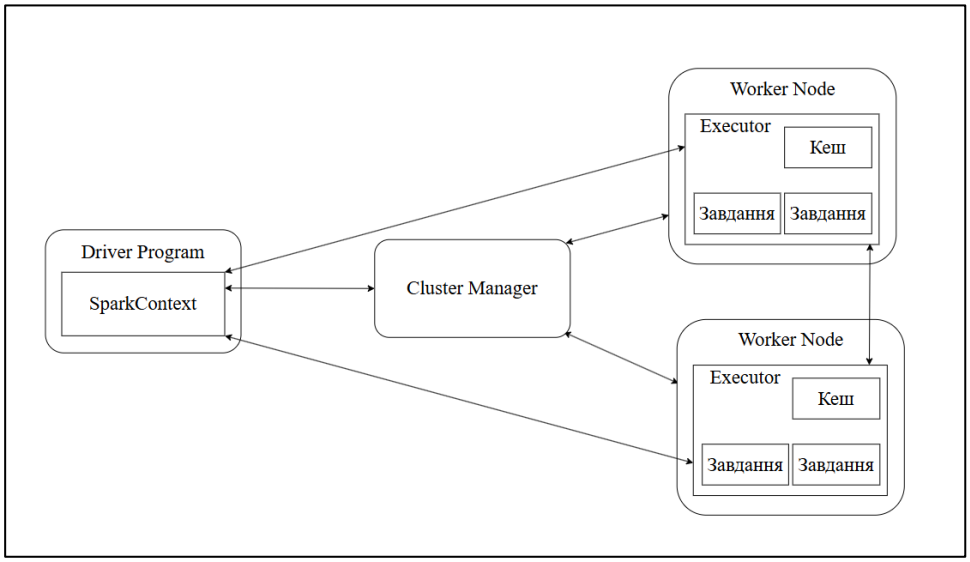

Рисунок 2.1 – Архітектура Apache Spark

У Spark керуюча програма (Driver Program) координує виконання застосунку, а контекст Spark (SparkContext) забезпечує взаємодію програми з кластером.

Розподіл ресурсів між застосунками виконує диспетчер кластера (Cluster Manager), після чого на робочих вузлах (Worker Nodes) запускаються виконавці (Executors), які виконують обчислення та можуть зберігати дані застосунку. Це дає змогу ефективно використовувати обчислювальні ресурси та масштабувати оброблення даних.

Як розподілене сховище даних було обрано Hadoop Distributed File System (HDFS) [33], яке інтегрується з Apache Spark і зберігає файли у вигляді блоків, розміщених на вузлах кластера. HDFS використовує керуючий вузол (NameNode), що підтримує простір імен файлової системи та зберігає її метадані, і вузли зберігання даних (DataNodes), на яких безпосередньо зберігаються блоки даних (рисунок 2.2).

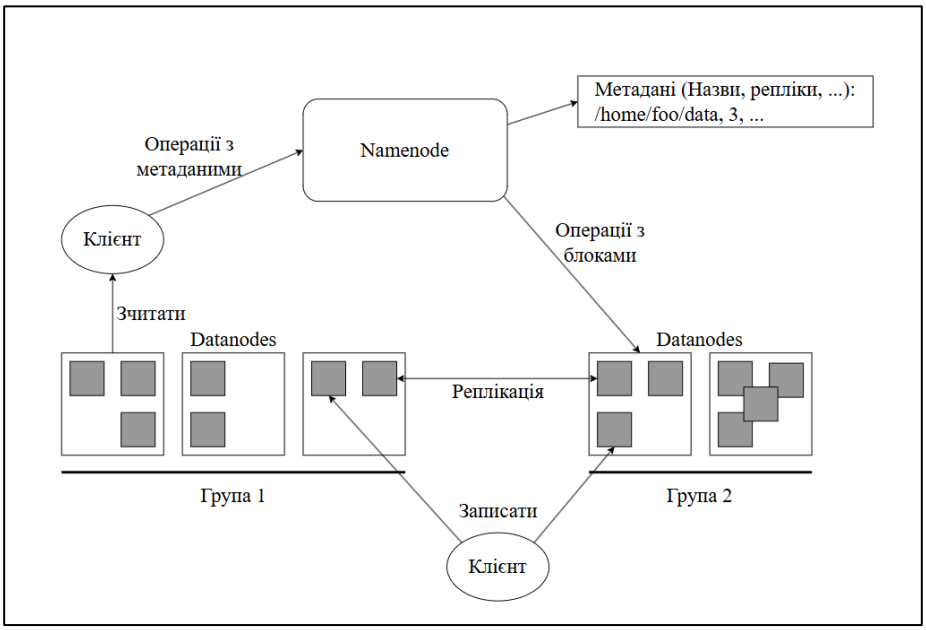

Рисунок 2.2 – Архітектура HDFS

Для забезпечення відмовостійкості блоки файлів реплікуються на вузли, що також підтримує паралельний доступ до даних. Це робить HDFS придатною для роботи з великими наборами даних.

Такий вибір обчислювального середовища зумовлений тим, що поєднання Apache Spark і HDFS забезпечує виконання повного циклу оброблення, зберігання та аналізу даних у межах розподіленого середовища. Крім того, така конфігурація

може бути розгорнута навіть у межах однієї фізичної машини із застосуванням технології контейнеризації. У цьому разі окремі компоненти обчислювального середовища та сховища даних можуть функціонувати як незалежні ізольовані середовища з визначеними обмеженнями обчислювальних ресурсів (процесор, оперативна пам’ять), що створює умови для дослідження масштабованості алгоритмів в умовах розподіленого зберігання та оброблення даних.

Таким чином, використання Apache Spark разом із HDFS є доцільним, оскільки це поєднання забезпечує розподілене середовище для зберігання та оброблення великих даних, підтримує виконання обраних алгоритмів і дає змогу досліджувати вплив масштабування на обчислювальну ефективність рекомендаційних систем. У межах такого середовища будуть реалізовані та досліджені алгоритми ALS, а також SAR у реалізації SARplus, що безпосередньо підтримують розподілені обчислення.

2.5 Стратегія та метод комбінування алгоритмів

Як стратегію комбінування алгоритмів було обрано пізнє злиття [34]. Оскільки в роботі використовується одна й та сама множина взаємодій користувачів та об’єктів без додаткових ознак, застосування раннього злиття [34] визначено недоцільним. Крім того, в умовах великих даних формування єдиного узгодженого представлення ознак супроводжується суттєвим зростанням розмірності даних та обчислювальної складності, що знижує ефективність такої стратегії. Також проміжне злиття [34] передбачає поєднання внутрішніх представлень моделей обраних алгоритмів, які є неоднорідними за своєю природою, а в окремих алгоритмах не формуються.

Пізнє злиття передбачає поєднання на рівні вихідних рекомендаційних списків без втручання у внутрішню логіку алгоритмів і після незалежного формування рекомендацій кожним із них [35], що дозволяє виконати як їх окреме

порівняння, так і порівняння з новими рекомендаціями, сформованими шляхом агрегування вже отриманих рекомендаційних списків кожного алгоритму. Оскільки алгоритми можуть формувати результати у різних шкалах оцінювання та з різними принципами впорядкування [36], що ускладнює безпосереднє об'єднання таких результатів, то у межах запропонованої стратегії використано принцип агрегування рангів об'єктів [37] у впорядкуваннях, що дозволяє ігнорувати абсолютні значення оцінок.

Окрім ігнорування абсолютних значень оцінок, використання принципу агрегування рангів є доцільним з огляду на те, що різні алгоритми формують неоднакові впорядкування об'єктів для одного користувача, оскільки базуються на різних принципах і враховують різні аспекти даних, а отже містять частково комплементарну інформацію [24]; при їх агрегуванні відбувається узагальнення цих оцінок, унаслідок чого релевантні об'єкти, які стабільно отримують високі позиції в різних впорядкуваннях, підсилюються, тоді як випадкові або шумові рекомендації окремих алгоритмів згладжуються, що в підсумку приводить до формування більш точних підсумкових рекомендацій.

Для подолання цієї проблеми було обрано метод WRRF [38], який застосовується для агрегування рангів кількох незалежних впорядкованих списків у межах концепції зваженої гібридизації, причому внесок кожного з них визначається як позицією об'єкта у списку, так і заданою вагою цього списку.

Порівняно з іншими методами агрегування рангів, такими як Borda Count [37] або Median [37], обраний підхід надає більшу вагу верхнім позиціям списків за рахунок оберненої залежності від рангу, що є критичним для задач формування верхньої частини рекомендаційного списку обмеженої довжини, тоді як зазначені методи або враховують позиції більш рівномірно, або використовують спрощене узагальнення позицій об'єктів.

Метод WRRF є узагальненням класичного методу RRF [39], призначеного для агрегування рангів документів у кількох незалежних системах впорядкування. У загальному випадку розглядається множина документів та сукупність незалежних впорядкованих списків, сформованих різними системами впорядкування. Кожен

такий список задає впорядкування документів за спаданням їх релевантності відповідно до внутрішнього критерію відповідної системи.

У рекомендаційних системах елементами впорядкування є об'єкти рекомендацій у списках, сформованих різними алгоритмами, а агрегування виконується окремо для кожного користувача. Кожний алгоритм формує для користувача впорядкований список, у якому об'єкти розташовані за спаданням їх прогнозованої релевантності. Для кожного об'єкта визначається його ранг у рекомендаційному списку відповідного алгоритму, якщо цей об'єкт присутній у списку. Для врахування можливості відсутності об'єкта у списку вводиться індикаторна функція присутності (саме ця деталізація відрізняє метод WRRF, запропонований у даній роботі, від методу, поданого в роботі [38]):

$$\mathbb{I}\left(i \in L_j(u)\right) = \begin{cases} 1, & i \in L_j(u) \\ 0, & i \notin L_j\,(u) \end{cases},$$

де u – користувач;

i – об'єкт рекомендації;

$L_j(u)$ – впорядкований список рекомендацій, сформований j-м алгоритмом для користувача u.

Для узагальнення методу вводяться вагові коефіцієнти, які визначають ступінь впливу кожного списку рекомендацій на агрегований результат. Умова нормалізації ваг задається наступним чином:

$$\sum_{j=1}^{N} w_j = 1, \qquad w_j \geq 0\,, j = 1, \dots, N,$$

де N – кількість незалежних впорядкованих списків, сформованих різними алгоритмами;

w_j – ваговий коефіцієнт j-го рекомендаційного списку.

З урахуванням введених позначень агрегований показник релевантності об’єкта для користувача за методом WRRF (рисунок 2.3) визначається за формулою:

$$WRRF(u,i) = \sum_{j=1}^{N} \frac{w_j}{k_{WRRF} + rank_j(u,i)} \mathbb{I}\Big(i \in L_j(u)\Big),$$

де k_{WRRF} – константа, що зменшує вплив значення рангу на підсумковий результат.

$rank_j(u,i)$ – ранг об’єкта i у списку рекомендацій $L_j(u)$.

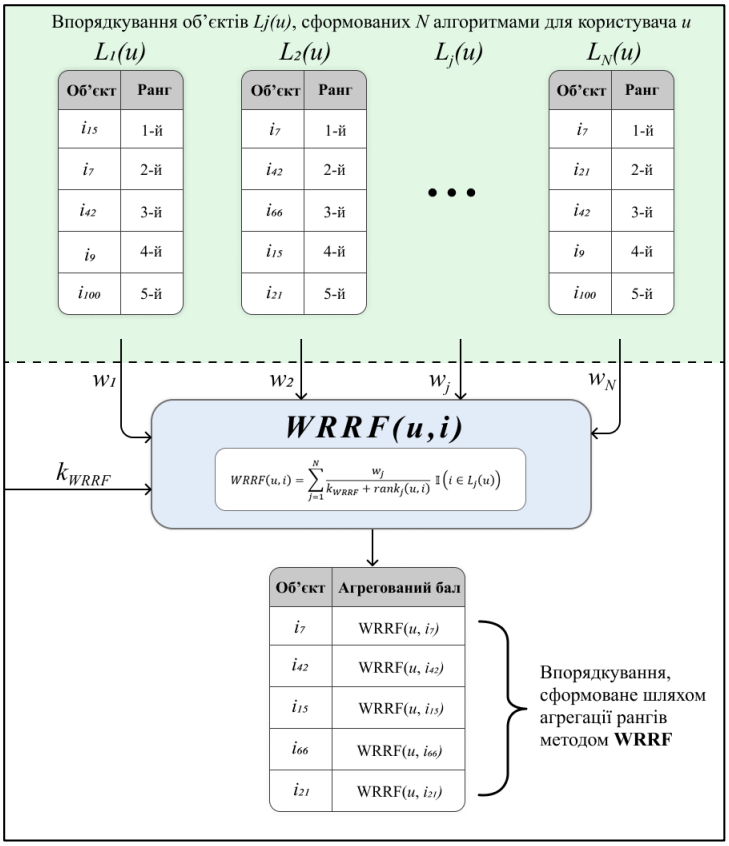

Рисунок 2.3 – Агрегація рангів упорядкувань за методом WRRF

Якщо об’єкт рекомендації відсутній у відповідному рекомендаційному списку, вважається, що його ранг перевищує ранг останнього елемента списку. Такий підхід дозволяє визначити ранг для всіх об’єктів множини рекомендацій та забезпечує коректність обчислення агрегованого показника релевантності.

Аналогічно формуванню підсумкового агрегованого списку документів, підсумковий агрегований список об’єктів для кожного користувача формується шляхом впорядкування їх за спаданням значення агрегованого показника:

$$L^*(u) = sort^{\downarrow}_{i \in I} WRRF(u, i),$$

де I – множина об’єктів рекомендацій;

$sort^{\downarrow}_{i \in I}$ –впорядкування об’єктів множини I за спаданням значення аргументу.

Таким чином, загальна формула WRRF може використовуватися як для попарного, так і для повного агрегування (комбінування) результатів рекомендаційних алгоритмів із більшою кількістю рекомендаційних списків.

2.6 Метрики обчислювальної ефективності алгоритмів

У межах роботи обчислювальна ефективність алгоритмів оцінюватиметься за часовими витратами обчислень. Таке рішення було прийнято з огляду на те, що для рекомендаційних систем практично значущим показником є тривалість виконання окремих етапів обробки даних, особливо в умовах великих даних, де обсяг інформації суттєво впливає на масштабованість та час виконання алгоритмів [13]. Залежно від типу алгоритму основні обчислювальні витрати визначаються як на етапі підготовки алгоритму, що включає побудову моделей або допоміжних структур та їх налаштування, так і на етапі формування рекомендацій.

Крім того, у разі використання методу WRRF для агрегування рангів об’єктів у кількох незалежно сформованих алгоритмами рекомендацій впорядкованих списках виникає додатковий етап обчислень, який включає послідовність операцій агрегації, обчислення, впорядкування та відбору, і тому потребує окремого врахування під час оцінювання обчислювальної ефективності.

З цього приводу оцінювання виконуватиметься за декомпозицією повного часу роботи на змістовно відокремлені його складові: час підготовки та час формування рекомендацій. Для комбінації алгоритмів додатково враховуватиметься час агрегування рангів у рекомендаційних списках. Така схема є коректною, оскільки дає змогу відокремити одноразові витрати етапу підготовки алгоритму від витрат, які безпосередньо виникають під час отримання рекомендацій.

Загальні часові витрати окремого алгоритму визначаються таким чином:

$$T_{full}(a) = T_{prep}(a) + T_{rec}(a),$$

де a – окремий алгоритм рекомендацій;

$T_{prep}(a)$ – час на підготовку алгоритму;

$T_{rec}(a)$ – час на формування рекомендацій.

Час на формування рекомендацій вимірюватиметься для всієї множини користувачів тестової вибірки, для яких у межах експерименту будуватимуться рекомендаційні списки. Це зумовлено тим, що такий підхід дозволяє отримати узагальнену оцінку обчислювальної ефективності алгоритмів у реальних умовах їх застосування зокрема в задачах оброблення великих даних, де рекомендації формуються одночасно для значної кількості користувачів. До цієї складової включатиметься обчислення рекомендацій та подальша обробка результатів: вилучення вже відомих взаємодій, повторне впорядкування кандидатів і відбір перших позицій підсумкового списку. Таким чином, ця величина відображатиме повний час отримання рекомендацій для всіх тестових користувачів.

Загальні часові витрати для комбінації двох алгоритмів визначаються як:

$$T_{full}(c) = T_{prep}(a_p) + T_{rec}(a_p) + T_{prep}(a_q) + T_{rec}(a_q) + T_{agg}(c),$$

де c – комбінація двох алгоритмів;

a_p, a_q – алгоритми, що входять до комбінації;

$T_{agg}(c)$ – час агрегування рангів рекомендаційних списків двох алгоритмів.

Агрегування рангів рекомендаційних списків двох алгоритмів виконується за трьома кроками.

Крок 1: повна агрегація списків різних алгоритмів за парами «користувач-об’єкт», після чого для кожного об’єкта визначаються його позиції у відповідних списках;

Крок 2: для кожної пари «користувач-об’єкт» обчислюється агрегований показник релевантності на основі цих позицій, що дозволяє сформувати узгоджену оцінку релевантності з урахуванням внеску кожного алгоритму;

Крок 3: ввиконується впорядкування об’єктів за спаданням отриманого агрегованого показника та відбір верхніх позицій підсумкового списку рекомендацій.

Таким чином, запропоновані обчислювальні витрати дозволяють окремо враховувати всі етапи методу й на цій основі порівнювати алгоритми та їх комбінації за повними або частковими часовими витратами.

2.7 Визначення метрик якості формування та впорядкування рекомендаційних списків

Оцінювання якості рекомендаційних алгоритмів у роботі виконуватиметься за включенням релевантних об’єктів до рекомендаційного списку та якістю їх розміщення у верхній частині [40]. Такий підхід є доцільним для рекомендаційних систем, оскільки практична цінність результату визначається не тільки наявністю релевантних об’єктів у списку, але й тим, наскільки рано вони з’являються для користувача. Це особливо важливо в умовах великих даних, де кількість потенційно релевантних об’єктів є значною, а користувач зазвичай взаємодіє лише з обмеженою верхньою частиною рекомендаційного списку.

Для оцінювання частки релевантних об’єктів серед тих, що входять до верхньої частини списку, використовуватиметься метрика точності [40]:

$$Precision@K = \frac{1}{|U_{eval}|} \sum_{u \in U_{eval}} \frac{|I_K^{rec}(u) \cap I_{rel}(u)|}{K},$$

де U_{eval} – множина користувачів, для яких виконується оцінювання;

$I_K^{rec}(u)$ – множина об’єктів, що входять до перших K позицій упорядкованого рекомендаційного списку для користувача u;

$I_{rel}(u)$ – множина релевантних об’єктів для користувача u;

K – довжина верхньої частини рекомендаційного списку.

Зі збільшенням значення метрики зростає частка релевантних об’єктів у рекомендаційному списку, що свідчить про підвищення точності рекомендацій.

Далі використовуватиметься метрика повноти [40], яка відображає частку релевантних об’єктів користувача у верхній частині списку. Зі збільшенням її значення зростає частка релевантних об’єктів, що свідчить про підвищення повноти рекомендацій. На відміну від точності, повнота відображає частку знайдених релевантних об’єктів від їх загальної кількості та обчислюється як:

$$Recall@K = \frac{1}{|U_{eval}|} \sum_{u \in U_{eval}} \frac{|I_K^{rec}(u) \cap I_{rel}(u)|}{|I_{rel}(u)|}.$$

Для подальшого обчислення усередненої точності для кожного користувача визначається допоміжний показник точності [40] за такою формулою:

$$Precision@p(u) = \frac{\left|I_p^{rec}(u) \cap I_{rel}(u)\right|}{p},$$

де p – кількість перших позицій упорядкованого рекомендаційного списку;

$I_p^{rec}(u)$ – множина об’єктів, що входять до перших p позицій упорядкованого рекомендаційного списку для користувача u.

Цей показник відображає частку релевантних об’єктів серед перших позицій списку при його поступовому розширенні.

На основі цього допоміжного показника для окремого користувача визначається AP (Average Precision) [40], яка використовується як складова для обчислення значення підсумкової метрики:

$$AP@K(u) = \frac{1}{\min(|I_{rel}(u)|, K)} \sum_{p=1}^{K} Precision@p(u) \cdot rel_p(u),$$

де $rel_p(u)$ – індикатор релевантності об’єкта, розміщеного на позиції p, який дорівнює 1, якщо об’єкт належить множині $I_{rel}(u)$, 0 – в іншому випадку.

Нормувальний множник у знаменнику обмежує кількість релевантних об’єктів, що можуть бути враховані в межах верхньої частини списку заданої довжини. Зі зростанням значення метрики підвищується узгодженість між наявністю релевантних об’єктів у списку та їх розташуванням на вищих позиціях.

Метрика точності MAP (Mean Average Precision) [40] визначається як середнє значення цієї метрики за всіма користувачами:

$$MAP@K = \frac{1}{|U_{eval}|} \sum_{u \in U_{eval}} AP@K(u).$$

Метрика MAP забезпечує інтегральну оцінку якості рекомендацій з урахуванням як відбору релевантних об’єктів, так і порядку їх розміщення у списку. Зі збільшенням значення метрики підвищується узгодженість між наявністю релевантних об’єктів у рекомендаційному списку та їх розташуванням на вищих позиціях списку.

Для оцінювання якості впорядкування рекомендаційного списку використовується допоміжна метрика DCG (Discounted Cumulative Gain) [40], яка застосовується як складова для подальшої нормалізації. Вона надає більшу вагу релевантним об’єктам на вищих позиціях і зменшує їх внесок зі зниженням позиції.

Для одного користувача ця метрика розраховується таким чином:

$$DCG@K(u) = \sum_{p=1}^{K} \frac{rel_p(u)}{log_2(p+1)}.$$

Отже, релевантні об’єкти, що розміщуються на початкових позиціях, й роблять більший внесок у значення метрики.

Для забезпечення коректного порівняння використовується допоміжне ідеальне IDCG (Ideal Discounted Cumulative Gain) [40] значення цієї метрики. Воно відповідає найкращому можливому впорядкуванню та використовується для нормалізації з урахуванням різної кількості релевантних об’єктів у рекомендаційному списку.

IDCG обчислюється таким чином:

$$IDCG@K(u) = \sum_{p=1}^{\min(|I_{rel}(u)|,K)} \frac{1}{log_2(p+1)}.$$

Метрика nDCG (Normalized Discounted Cumulative Gain) [40] визначається як середнє значення нормалізованих показників за всіма користувачами:

$$nDCG@K = \frac{1}{|U_{eval}|} \sum_{u \in U_{eval}} \frac{DCG@K(u)}{IDCG@K(u)}.$$

Зі збільшенням її значення підвищується якість розміщення релевантних об’єктів у верхній частині рекомендаційного списку.

Таким чином, сукупне використання метрик точності, повноти, MAP та nDCG забезпечує комплексне оцінювання якості формування та впорядкування рекомендаційних списків. Точність характеризує частку релевантних об’єктів серед рекомендованих, повнота – частку знайдених релевантних об’єктів відносно їх загальної кількості, MAP – якість відбору та впорядкування релевантних об’єктів, а nDCG – ступінь наближення фактичного порядку до ідеального.

Усі метрики набувають значень від 0 до 1, причому більше значення відповідає вищій якості рекомендацій. Цей набір метрик доцільний для рекомендацій при неявному зворотному зв’язку, де ключове значення має якість верхньої частини списку.

2.8 Розроблення методики проведення дослідження

Методика проведення дослідження передбачає експериментальний сценарій, у якому, згідно з постановкою задачі, слід використовувати множину записів позитивних взаємодій у форматі неявного зворотного зв’язку для обраних наборів даних, а також здійснити їх розбиття на навчальну та тестову вибірки. Для цього буде застосовано стратифіковане розбиття за користувачами у співвідношенні 75:25, за якого для кожного користувача зберігатиметься наявність взаємодій як у навчальній, так і в тестовій частинах. Такий спосіб формування вибірок дасть змогу уникнути ситуації, коли користувач буде повністю відсутній у навчальних даних, і тим самим забезпечить коректність побудови персоналізованих рекомендацій.

Навчальна вибірка використовуватиметься для навчання алгоритмів, тоді як тестова – для оцінювання якості рекомендацій на взаємодіях, що не брали участі в навчанні. Також, згідно з постановкою задачі, експериментальний сценарій буде запущено в різних обчислювальних середовищах і на різних наборах даних. При цьому в розподіленому середовищі він виконуватиметься на ширшому діапазоні

обсягів даних для повноцінного аналізу масштабованості та оцінки ефективності алгоритмів у задачах великих даних.

Підготовка алгоритму, вимірювання часу її виконання, формування рекомендацій, вимірювання часу їх побудови, а також їх оцінювання й інші супутні операції виконуватимуться послідовно в межах кожного алгоритму. Самі алгоритми виконуватимуться почергово (послідовно), один за одним. Водночас для кожного алгоритму використовуватиметься єдина постановка задачі, що відповідає конфігурації оброблення даних у форматі неявного зворотного зв'язку.

У межах цієї послідовності для кожного алгоритму на етапі формування рекомендацій спочатку генеруватиметься рекомендаційний список розширеної довжини (саме для нього буде вимірюватися час рекомендацій), обсяг якого перевищує розмір верхньої частини підсумкового рекомендаційного списку та містить кандидати-об'єкти для його подальшого формування:

$$K_{ext} = K + B,$$

де B – кількість додаткових рекомендацій.

Такий підхід зумовлений необхідністю подальшого виключення об'єктів, з якими користувач уже взаємодіяв у навчальній вибірці, що забезпечить включення до підсумкового рекомендаційного списку лише нових для користувача об'єктів. Після видалення таких об'єктів і повторного впорядкування формується підсумковий рекомендаційний список фіксованої довжини. Таким чином, використання розширеного початкового списку дає змогу забезпечити достатню кількість об'єктів після виключення та гарантує коректне подальше оцінювання якості рекомендацій за списком однакової фіксованої довжини.

Оцінювання якості рекомендацій здійснюватиметься для верхньої частини рекомендаційного списку довжиною 10. Для забезпечення достатньої кількості об'єктів-кандидатів кількість додаткових рекомендацій буде дорівнювати 200.

З метою забезпечення узгодженості експериментального процесу та можливості комбінування результатів зберігатимуться сформовані алгоритмами

рекомендаційні списки після виключення об’єктів, що вже містяться в історії взаємодій користувача. Ці результати будуть використані як вхідні для подальшого агрегування рангів об’єктів рекомендацій із застосуванням методу WRRF.

Такий підхід гарантуватиме коректність порівняльного оцінювання якості рекомендацій між алгоритмами та їх комбінаціями, оскільки всі вони оцінюватимуться за однакових умов, без повторного навчання, а комбінування здійснюватиметься на основі однакових вхідних даних, що використовуватимуться для формування підсумкових рекомендацій окремих алгоритмів.

Окрім забезпечення порівнюваності результатів, використання рекомендаційного списку розширеної довжини з кандидатами-об’єктами обґрунтовується тим, що різні алгоритми можуть формувати рекомендаційні списки за різними принципами впорядкування. Унаслідок цього підсумкові списки однакової довжини, отримані різними алгоритмами, можуть істотно відрізнятися за складом. Це призводитиме до недостатнього перетину між списками та, відповідно, обмежуватиме кількість об’єктів, для яких агрегування рангів є інформативним. У результаті це ускладнюватиме застосування методу WRRF.

Відповідно, використання такого списку забезпечуватиме розширення множини кандидатних об’єктів і підвищуватиме ймовірність їх одночасної присутності в результатах різних алгоритмів. Це, у свою чергу, сприятиме підвищенню ефективності агрегування під час формування підсумкового рекомендаційного списку обмеженої довжини.

Оскільки на якість підсумкового впорядкування, сформованого методом WRRF, впливатимуть не лише пари вхідних рекомендаційних списків, а й значення параметрів методу, зокрема вагові коефіцієнти списків і константа, що регулює вплив значення рангу на підсумковий результат, для кожної пари алгоритмів та кожного набору даних здійснюватиметься підбір таких значень параметрів, які забезпечуватимуть максимальну якість підсумкового рекомендаційного списку.

Для парних комбінацій оптимальні значення параметрів визначатимуться окремо для кожної пари алгоритмів шляхом перебірного пошуку на регулярній сітці параметрів (Grid Search) [41] за формулою 3.1.

$$(w_1^*, k_{WRRF}^*) = \arg \max_{\substack{w_1 \in \{0,1;\ 0,2;\ \ldots;\ 0,9\} \\ k_{WRRF} \in \{10;\ 20;\ \ldots;\ 100\}}} nDCG@K(w_1, k_{WRRF}), \quad (3.1)$$

де $nDCG@K(w_1, k_{WRRF})$ – значення метрики nDCG для списку довжини K за фіксованими значеннями параметрів w_1 та k_{WRRF}.

Перебір виконуватиметься на дискретних множинах значень параметрів: ваговий коефіцієнт змінюватиметься в межах від 0,1 до 0,9 з кроком 0,1, а константа методу – в діапазоні від 10 до 100 з кроком 10.

Для повної комбінації використано випадковий пошук (Random Search) [42], оскільки пошук на регулярній сітці є доволі обчислювально складним. Параметри генеруватимуться випадково: ваги моделей формуватимуться за допомогою розподілу Діріхле [43], що забезпечуватиме їх додатність і нормування до одиниці, а константа вибиратиметься рівномірно із заданого діапазону.

Вибір метрики nDCG як цільової функції зумовлений тим, що вона враховує порядок рекомендацій і надає більшу вагу релевантним елементам на початку списку, на відміну від метрик точності та повноти. Порівняно з MAP, nDCG оцінює наближеність отриманого порядку до ідеального, тому є більш придатною для задач ранжування, оскільки чутливіша до змін у верхній частині списку, яка має найбільше значення для користувача.

У результаті для кожної пари алгоритмів обиратиметься конфігурація параметрів, що забезпечує максимальне значення nDCG, після чого отримане впорядкування оцінюватиметься за іншими метриками. Це зумовлено тим, що для оцінювання недостатньо використовувати лише один показник, оскільки різні метрики відображають різні аспекти якості: точність добору, повноту охоплення релевантних об’єктів та якість їх розміщення у верхній частині списку.

Всі експерименти виконуватимуться за однаковим сценарієм, у межах якого відмінності в якості рекомендацій можуть бути пов’язані саме з властивостями алгоритмів, а не з різницею у даних, постановці задачі чи конфігурації середовища.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Препроцесне оброблення даних

Етап 1. Для уніфікації подальшої обробки всі записи з усіх наборів даних були приведені до єдиної схеми:

$$x = (user\_id, item\_id, rating, timestamp),$$

де $user\_id$ – ідентифікатор користувача;

$item\_id$ – ідентифікатор об’єкта;

$rating$ – значення рейтингу;

$timestamp$ – часова мітка взаємодії.

Це означало зіставлення назв полів записів у вихідних наборах даних із єдиною схемою з чітко визначеними атрибутами. Ідентифікатори користувачів та об’єктів у наборах Amazon Reviews 2023, які початково представлені символьними рядками, були перетворені на цілочисельні індекси шляхом перенумерації унікальних значень. Аналогічно значення рейтингу в наборі даних MovieLens приводилися до числового формату з плаваючою крапкою. В наборі Jester матрицю було перетворено у формат явних взаємодій шляхом виділення спостережуваних оцінок, а часові мітки заповнено поточним часом для коректної роботи алгоритмів, що залежать від цього параметра. У результаті всі записи були приведені до єдиного стандартизованого формату подання взаємодій, що дозволило виконувати їх подальше оброблення за однаковою процедурою.

Етап 2. Фільтрація некоректних записів. Взаємодія вважалася коректною, якщо для неї визначені ідентифікатор користувача, ідентифікатор об’єкта та часова мітка, а значення цих атрибутів є коректними. Ідентифікатори користувачів та об’єктів подаються як цілочисельні значення, тоді як часові мітки – у форматі Unix-часу. Належність атрибутів до визначених типів у межах прийнятої схеми подання

означає, що вони є визначеними, мають коректний формат і не містять пропущених значень. Таким чином, набір коректних записів визначався так:

$$D_{val} = \{ x \in D_{raw} \mid valid(x) \},$$

де D_{raw} – початкова множина записів;

$valid(x)$ – умова коректності взаємодії x.

Відсутність або некоректність значення рейтингу не призводила до автоматичного вилучення запису: у такому разі рейтинг приводився до числового формату з підстановкою нульового значення:

$$rating'(x) = \begin{cases} rating(x), & rating(x) \in R \\ 0{,}0, & rating(x) \notin R' \end{cases}$$

де R – множина допустимих значень рейтингу (Jester: від –10,00 до +10,00 з кроком 0,01; інші – від 1,0 до 5,0 з кроком 1,0) для вихідної множини записів.

Етап 3. У подальшій обробці використовувалося оброблене значення рейтингу, а нульове значення розглядалося як службове позначення відсутності коректно визначеного рейтингу, яке на наступних етапах інтерпретувалося як відсутність позитивного сигналу.

Далі виконувалося видалення дублікатів записів. Очищений набір визначався як:

$$D_{uniq} = uniq_{(user\_id,item\_id,timestamp)}(D_{val}),$$

де $uniq_{(user\_id,item\_id,timestamp)}$ – операція вилучення дублікатів записів за комбінацією атрибутів ідентифікатора користувача, ідентифікатора об’єкта та часової мітки, у результаті якої зберігається лише один запис для кожної унікальної трійки цих значень.

Такий підхід дає змогу усунути технічні повторення одних і тих самих взаємодій, які можуть спотворювати статистичні характеристики набору даних, зокрема кількість записів, щільність матриці взаємодій та частоти появи об’єктів.

Етап 4. Ключовим етапом підготовки даних була k-core-фільтрація [44], яка застосовувалася лише до наборів даних Movies and TV та Video Games. Її мета полягає в тому, щоб залишити в наборі лише тих користувачів, які мають не менше заданої кількості взаємодій, і лише ті об’єкти, з якими взаємодіяла достатня кількість користувачів. Оскільки видалення окремих користувачів або об’єктів може призвести до того, що інші елементи також перестануть задовольняти умови мінімальної кількості взаємодій, k-core-фільтрація виконується ітеративно до досягнення стабільного стану множини записів.

Початкова множина записів для процедури фільтрації задається як:

$$D_{uniq}^{(0)} = D_{uniq}.$$

Тоді кількість взаємодій користувачів та об’єктів на певній ітерації визначається як:

$$n_{user}^{(t)}(u\_id) = \left|\left\{x \in D_{uniq}^{(t)} \mid user\_id(x) = u\_id\right\}\right|,$$
$$n_{item}^{(t)}(i\_id) = \left|\left\{x \in D_{uniq}^{(t)} \mid item\_id(x) = i\_id\right\}\right|,$$

де t – номер ітерації;

$D_{uniq}^{(t)}$ – множина записів на ітерації t;

$n_{user}^{(t)}(u\_id)$ – кількість взаємодій користувача з ідентифікатором u_id у множині записів $D_{uniq}^{(t)}$;

$n_{item}^{(t)}(i\_id)$ – кількість взаємодій об’єкта з ідентифікатором i_id у множині записів $D_{uniq}^{(t)}$.

Умова належності запису до k-core на певній ітерації визначається як:

$$core^{(t)}(x) = \left(n_{user}^{(t)}(user\_id(x)) \geq k_{user} \wedge n_{item}^{(t)}(item\_id(x)) \geq k_{item}\right),$$

де k_{user} – мінімально допустима кількість взаємодій користувача;

k_{item} – мінімально допустима кількість взаємодій об’єкта.

Етап 5. Формування нової множини записів на новій ітерації:

$$D_{uniq}^{(t+1)} = \left\{x \in D_{uniq}^{(t)} \mid core^{(t)}(x)\right\}.$$

Ітеративна процедура фільтрації виконується доти, поки множина записів перестає змінюватися:

$$D_{uniq}^{(t+1)} = D_{uniq}^{(t)}.$$

Тоді отриману стабільну множину записів після k-core-фільтрації можна визначити як:

$$D_{core} = D_{uniq}^{(t^*)},$$

де t^* – ітерація, на якій уперше досягнуто стабільності.

Застосування цієї процедури було необхідним саме для наборів Amazon Reviews 2023, оскільки вони характеризуються значно більшою кількістю користувачів і об’єктів та значною часткою користувачів і об’єктів з дуже малою кількістю взаємодій. Така структура даних ускладнює навчання рекомендаційних алгоритмів і може призводити до нестабільності оцінок параметрів. Для набору MovieLens додаткова k-core-фільтрація не застосовувалася.

Додатково з набору Jester було сформовано експериментальну підвибірку, що зберігає ключові характеристики даних (перевагу користувачів над об’єктами) та зменшує обсяг вибірки для скорочення часу експериментів у локальному середовищі. Для цього відібрано лише записи з рейтингом понад 8,5.

Після завершення описаних етапів попередньої обробки даних для кожного набору даних визначалася розрідженість матриці взаємодій. Її обчислення виконувалося для тієї множини записів, яка була отримана після очищення та, за потреби, k-core-фільтрації або скорочення за значенням рейтингу: для наборів даних Movies and TV та Video Games – для стабільної множини записів після k-core-фільтрації, для набору даних MovieLens – для очищеного набору даних без дублікатів, для Jester – для експериментальної підвибірки, сформованої за порогом рейтингу.

Розрідженість визначалася як частка відсутніх взаємодій відносно загальної кількості всіх можливих пар «користувач-об’єкт» за такою формулою:

$$S(D') = 1 - \frac{|D'|}{\left|U'_{ids}\right| \cdot \left|I'_{ids}\right|},$$

де $D' \in \{D_{core}, D_{uniq}\}$ – оброблена множина записів, для якої обчислюється розрідженість;

$U'_{ids} = \{user\_id(x) \mid x \in D'\}$ – множина унікальних користувачів;

$I'_{ids} = \{item\_id(x) \mid x \in D'\}$ – множина унікальних об’єктів.

Така характеристика дає змогу оцінити, яка частина потенційно можливих взаємодій фактично відсутня в наборі даних. Для задач рекомендації це є типовою властивістю реальних даних, оскільки кожний користувач зазвичай взаємодіє лише з невеликою частиною доступних об’єктів.

Основні характеристики підготовлених наборів даних наведено в таблиці 3.1. Як видно з цієї таблиці, переважна більшість використаних наборів даних характеризується високим рівнем розрідженості [45].

Таблиця 3.1 – Характеристики обраних наборів даних

Назва набору	Розрідженість	Кількість рейтингів	Кількість унікальних користувачів	Кількість унікальних об’єктів
MovieLens 100K	0,936953	100000	943	1682
MovieLens 1M	0,955316	1000209	6040	3706
MovieLens 10M	0,986597	10000054	69878	10677
MovieLens 20M	0,994600	20000263	138493	26744
MovieLens 25M	0,997395	25000095	162541	59047
MovieLens 32M	0,998114	32000204	200948	84432
Movies and TV 25-core	0,971831	63333	1423	1580
Video Games 15-core	0,979659	45134	1846	1202
Jester	0,925959	92596	12506	100

Після цього всі підготовлені набори даних приводилися до формату неявного зворотного зв’язку. Для цього оброблене значення рейтингу замінювалося бінарною ознакою, яка відображала наявність або відсутність позитивної взаємодії.

Для наборів даних MovieLens, Movies and TV та Video Games це реалізовано за таким правилом:

$$rating_{impl}(x) = \begin{cases} 1, & rating'(x) \geq 4{,}0 \\ 0, & rating'(x) < 4{,}0 \end{cases}.$$

Порогове значення було обрано з урахуванням того, що в обраних наборах даних рейтинги задані за п’ятибальною шкалою в діапазоні від 1 до 5. Така шкала утворює впорядковану множину значень, у якій значення 3 є медіаною та інтерпретується як нейтральна оцінка, а значення 4,0 використовується як поріг для відокремлення позитивного сигналу від нейтрального або негативного. Також значення 0, яке було введено на попередньому етапі обробки, інтерпретується як відсутність достовірно встановленого позитивного сигналу, оскільки через пошкодження даних неможливо достовірно визначити реальну оцінку.

Оскільки з набору Jester вже було виокремлено підмножину за пороговим принципом, у цьому випадку всі значення рейтингів інтерпретуються як позитивний сигнал.

Відбір позитивних взаємодій визначається як:

$$p^{+}(x) = \left(x \in D' \wedge rating_{impl}(x) = 1\right).$$

Тоді множина записів позитивних взаємодій у форматі неявного зворотного зв'язку визначається як кортеж:

$$D_{impl} = \left\{ \left(user\_id(x), item\_id(x), rating_{impl}(x), timestamp(x)\right) \;\middle|\; p^{+}(x) \right\}.$$

Це означає, що до подальшої обробки включаються лише позитивні взаємодії, при цьому значення рейтингу замінюється бінарною ознакою. Відсутність взаємодії не зберігається явно у вигляді записів із нульовим значенням, а задається неявно як відсутність відповідних пар «користувач-об'єкт» у множині записів позитивних взаємодій.

Таке подання дозволяє перейти від числових рейтингових оцінок до єдиного формату позитивних взаємодій, придатного для подальшого навчання та порівняння рекомендаційних алгоритмів.

3.2 Підготовка обчислювальних середовищ

Для проведення експериментального дослідження підготовлено локальне обчислювальне середовище, побудоване на базі операційної системи Windows 11. Апаратна конфігурація обчислювальної системи включає центральний процесор із 14 ядрами та 20 потоками виконання, а також 32 ГБ оперативної пам'яті.

Для забезпечення сумісності з інструментами екосистеми Linux використовується підсистема Windows для Linux (WSL) з дистрибутивом довгострокової підтримки (LTS) Ubuntu 22.04.5. У межах цього середовища окремо було сформовано 3 підсередовища:

– 1 процесорні ядра, 24 ГБ оперативної пам’яті та 8 ГБ буферної пам’яті;

– 3 процесорні ядра, 24 ГБ оперативної пам’яті та 8 ГБ буферної пам’яті;

– 6 процесорних ядер, 24 ГБ оперативної пам’яті та 8 ГБ буферної пам’яті.

Оскільки деякі алгоритми належать до нейромережевих методів, які ефективніше виконуються на графічних прискорювачах, додатково було налаштовано середовище в Kaggle [46] з 3 процесорними ядрами, 29 ГБ оперативної пам’яті та GPU (Graphics Processing Unit) із підтримкою CUDA (Compute Unified Device Architecture) та обсягом відеопам’яті 16 ГБ (Tesla P100-PCIE-16GB).

Додатково було розгорнуто розподілене обчислювальне середовище на базі Apache Spark з можливістю варіювання конфігурації кластеру (рисунок Б.1). Керуючій програмі виділено до 6 процесорних ядер та 6 ГБ оперативної пам’яті, кожному робочому вузлу – від 2 до 4 процесорних ядер і від 4 до 8 ГБ оперативної пам’яті залежно від конфігурації (таблиця 3.2).

Для дослідження використано декілька його конфігурацій, що відрізняються кількістю робочих вузлів, а також обсягом виділених обчислювальних ресурсів.

Таблиця 3.2 – Конфігурації розподіленого обчислювального середовища Apache Spark

Конфігурація	Кількість робочих вузлів	Кількість виділених ядер на робочий вузол	Кількість виділеного ОЗУ на робочий вузол, ГБ
1	1	2	4
2	2	2	4
3	3	2	4
4	3	2	6

Кінець таблиці 3.2

Конфігурація	Кількість робочих вузлів	Кількість виділених ядер на робочий вузол	Кількість виділеного ОЗУ на робочий вузол, ГБ
5	3	2	8
6	3	3	4
7	3	4	4
8	3	4	8

Для зберігання та обробки даних використовується розподілене файлове сховище HDFS, яке складається з вузла імен (NameNode), що відповідає за управління метаданими, та вузлів даних (DataNode), на яких безпосередньо зберігаються дані. Вузли HDFS розгорнуті в тому ж обчислювальному середовищі та використовують доступні ресурси системи для забезпечення зберігання даних і доступу до них під час виконання обчислень.

Така конфігурація обчислювального середовища є достатньою для виконання експериментів з алгоритмами колаборативної фільтрації, обробки наборів даних різного обсягу, а також для запуску компонентів розподіленої обробки даних на локальному ресурсі.

3.3 Конфігурування експериментального програмного середовища

Програмну конфігурацію експериментального середовища реалізовано з використанням Conda як засобу керування пакетами та віртуальним середовищем. Для виконання дослідження створено окреме ізольоване середовище Python 3.9, що забезпечує відокремлення залежностей та узгодженість програмних компонентів. Для інтерактивного виконання обчислень створене середовище зареєстровано як окреме ядро Jupyter Notebook за допомогою пакета ipykernel.

Встановлення програмних компонентів виконувалося із застосуванням двох наступних механізмів.

Через Conda [47] інстальовано базові компоненти середовища, зокрема Python 3.9, OpenJDK 17 та окремі системні залежності. Через pip у межах створеного середовища встановлено спеціалізовані бібліотеки, необхідні для реалізації рекомендаційних алгоритмів, обробки даних та оцінювання результатів.

До складу програмної конфігурації включено бібліотеки загального призначення, що використовуються для оброблення даних і супровідних операцій: для роботи з числовими та табличними даними застосовано NumPy і Pandas, а для службових операцій – стандартні модулі Python warnings, os, logging, time та inspect [48].

Як основу експериментального сценарію використано бібліотеку Microsoft Recommenders [30]. Її засоби застосовано для завантаження набору даних MovieLens, стратифікованого розбиття даних на навчальну і тестову вибірки, підготовки вхідних даних до навчання моделей, реалізації окремих рекомендаційних алгоритмів та обчислення метрик якості. Інші набори даних завантажуються з попередньо підготовлених локальних CSV-файлів.

Для реалізації алгоритму ALS, а також частини процедур обробки, впорядкування та оцінювання використано екосистему Apache Spark через інтерфейс PySpark. У програмній реалізації застосовано модулі pyspark.sql.functions, pyspark.sql.window та pyspark.ml.recommendation. Для забезпечення коректної роботи Spark у складі середовища встановлено OpenJDK 17 і налаштовано локальне Java-оточення в межах Conda-середовища.

Окремо для проведення дослідження у рідному середовищі виконання алгоритму ALS було розгорнуто кластер Apache Spark із використанням технології контейнеризації Docker. Такий підхід дозволив відтворити типову архітектуру розподіленого обчислювального середовища, що включає керуючий вузол, робочі вузли, а також компоненти розподіленого сховища даних HDFS.

Для реалізації інших алгоритмів колаборативної фільтрації використано спеціалізовані бібліотеки. Зокрема, для алгоритму SARPlus застосовано пакет

pysarplus, для алгоритмів BPR та BiVAE – бібліотеку Cornac, а для нейромережевих алгоритмів NCF та LightGCN – відповідні компоненти бібліотеки Microsoft Recommenders. Додатково до конфігурації включено пакет scikit-surprise [49]. При цьому в межах розгорнутого кластера Apache Spark було модифіковано бібліотеку SARPlus з метою забезпечення її коректної роботи з HDFS, зокрема для підтримки зберігання та використання кешу в розподіленому файловому сховищі та його подальшого застосування під час формування рекомендацій.

Таким чином, сформоване програмне середовище є ізольованим, технічно узгодженим і придатним для проведення експериментального дослідження алгоритмів колаборативної фільтрації.

3.4 Виконання програмного експериментального дослідження

Програмне експериментальне дослідження виконується наступними етапами, що сформовані відповідно до методики проведення дослідження.

Етап 1. Виконується завантаження підготовленого набору даних із локального CSV-файл та стратифіковане розбиття на навчальну і тестову вибірки згідно з наступним кодом:

```
df = pd.read_csv(data_path)
df_train, df_test = python_stratified_split(df,
    ratio=0.75,
    min_rating=1,
    filter_by="user",
    col_user=DEFAULT_USER_COL,
    col_item=DEFAULT_ITEM_COL,
)
```

Етап 2. Виконується ініціалізація алгоритмів, параметри яких обрані згідно запропонованих в [30].

Ініціалізація алгоритму ALS відбувається із заданими параметрами, що визначають розмір латентного простору, кількість ітерацій оптимізації та параметри регуляризації. Це виконує наступний фрагмент коду:

```
als_est = SparkALS(
    rank=10,
    maxIter=20,
    regParam=0.05,
    implicitPrefs=True,
    alpha=40.0,
    nonnegative=False,
    userCol=DEFAULT_USER_COL,
    itemCol=DEFAULT_ITEM_COL,
    ratingCol=DEFAULT_RATING_COL,
    coldStartStrategy="drop",)
```

В алгоритмі BPR задаються параметри латентного простору, швидкість навчання та коефіцієнт регуляризації, що визначають оптимізацію моделі. Це виконується в наступному фрагменті коду:

```
bpr_model, _ = train_bpr({
        "k": 200,
        "max_iter": 200,
        "learning_rate": 0.01,
        "lambda_reg": 1e-3,
        "seed": SEED,
        "num_threads": N_CORES,
        "verbose": False,},
    bpr_train,)
```

Ініціалізація алгоритму BiVAE відбувається із заданими параметрами, що визначають структуру енкодера, функцію активації, тип правдоподібності та параметри навчання (за використання середовища з графічним процесором відповідний параметр вмикається). Це виконує наступний фрагмент коду:

```
bivae_model, _ = train_bivae({
        "k": 100,
        "encoder_structure": [200],
        "act_fn": "tanh",
        "likelihood": "pois",
        "n_epochs": 500,
```

```
        "batch_size": 1024,
        "learning_rate": 0.001,
        "seed": SEED,
        "use_gpu": False,
        "verbose": False,},
    bivae_train,)
```

Ініціалізація алгоритму LightGCN відбувається із заданими параметрами, що визначають кількість шарів графової моделі, розмір векторних представлень і параметри оптимізації. Це виконує наступний фрагмент коду:

```
lgcn_model, _ = train_lightgcn({
        "model_type": "lightgcn",
        "n_layers": 3,
        "batch_size": 1024,
        "embed_size": 64,
        "decay": 0.0001,
        "epochs": 20,
        "learning_rate": 0.005,
        "eval_epoch": 5,
        "top_k": TOPK_REQ,
        "metrics": ["recall", "ndcg", "precision", "map"],
        "save_model": False,
        "MODEL_DIR": ".",},lgcn_train,)
```

Ініціалізація алгоритму NCF відбувається із заданими параметрами, що визначають архітектуру нейронної мережі, кількість факторів і параметри навчання. Це виконує наступний фрагмент коду:

```
ncf_model, _ = train_ncf({
        "model_type": "NeuMF",
        "n_factors": 4,
        "layer_sizes": [16, 8, 4],
        "n_epochs": 15,
        "batch_size": 1024,
        "learning_rate": 1e-3,
        "verbose": 10,},
    ncf_train,
)
```

Ініціалізація алгоритму SARplus відбувається із заданими параметрами, що визначають модель та колонки, які використовуються для обчислення подібностей.

Це виконує наступний фрагмент коду:

```
sarplus_model = SARPlus(
    spark,
    col_user=DEFAULT_USER_COL,
    col_item=DEFAULT_ITEM_COL,
    col_rating=DEFAULT_RATING_COL,
)
```

Етап 3. Виконується навчання моделей та формування рекомендацій для користувачів тестової вибірки з урахуванням розширеного списку кандидатів.

Наступний фрагмент коду формує розширений список рекомендацій, вилучає вже відомі взаємодії та виконує повторне впорядкування об’єктів.

```
als_raw = (
    als_model.recommendForUserSubset(test_users_sdf, TOPK_REQ)
    .select(DEFAULT_USER_COL,
F.explode("recommendations").alias("rec"))
    .select(
        F.col(DEFAULT_USER_COL),
F.col("rec").getField(DEFAULT_ITEM_COL).alias(DEFAULT_ITEM_COL),
        F.col("rec").getField("rating").alias("score"),
    )
)
```

Етап 4. Агрегування рекомендаційних списків алгоритмів рекомендацій методом WRRF.

Наступний код виконує повну агрегацію рекомендаційних списків, обчислення агрегованого показника та формування підсумкового впорядкування:

```
def rrf_fuse(a_ranked, b_ranked, w_a=0.5, k0=60, topk=DEFAULT_K):
    a   =   a_ranked.select(DEFAULT_USER_COL,   DEFAULT_ITEM_COL,
F.col("rank").alias("rank_a"))
    b   =   b_ranked.select(DEFAULT_USER_COL,   DEFAULT_ITEM_COL,
F.col("rank").alias("rank_b"))
    fused = ( a.join(b, on=[DEFAULT_USER_COL, DEFAULT_ITEM_COL],
how="full_outer").withColumn(DEFAULT_PREDICTION_COL,
            F.when(F.col("rank_a").isNotNull(),  w_a  /  (k0  +
F.col("rank_a"))).otherwise(0.0)+
            F.when(F.col("rank_b").isNotNull(), (1 - w_a) / (k0
+             F.col("rank_b"))).otherwise(0.0)))w             =
```

```
Window.partitionBy(DEFAULT_USER_COL).orderBy(F.col(DEFAULT_PRED
ICTION_COL).desc())
    return fused.withColumn("rank", F.row_number().over(w)
)
```

Підбір параметрів методу WRRF відбувається шляхом перебору значень ваг і константи згідно з наступним кодом:

```
for current_w in w_range:
    for current_k0 in k0_range:
        fused_topk       =       rrf_fuse(ranked_by_algo_sdf[a],
ranked_by_algo_sdf[b], w_a=current_w, k0=current_k0)
        fused_metrics   =   ranking_metrics_pyspark(test_metrics,
fused_topk, K)
```

Етап 5. Оцінювання обчислювальної ефективності.

Наступний код виконує вимірювання часу підготовки моделі, формування рекомендацій та їх агрегування:

```
t0 = time.time()
model = als_est.fit(train_sdf)
time_train = time.time() - t0
t1 = time.time()
als_topk = ...
time_reco = time.time() - t1
t_fuse0 = time.time()
fused_topk = ...
time_fuse = round(time.time() - t_fuse0, 4)
```

Етап 6. Оцінювання якості рекомендаційних списків.

Оцінювання якості виконується за допомогою готових реалізацій метрик згідно з наступним фрагментом коду:

```
def ranking_metrics_pyspark(test, predictions, k=DEFAULT_K):
    rank_eval = SparkRankingEvaluation(
        test,
        predictions,
        k=k,
        relevancy_method="top_k",
        **COL_DICT
    )
```

```
    return {
        "MAP": rank_eval.map(),
        "nDCG@k": rank_eval.ndcg_at_k(),
        "Precision@k": rank_eval.precision_at_k(),
        "Recall@k": rank_eval.recall_at_k(),
}
```

У цьому коді використовується клас SparkRankingEvaluation [50], який забезпечує обчислення всіх необхідних метрик без додаткової реалізації.

Описані конфігурації відповідають коду як у середовищі Jupyter Notebook, так і в середовищі Apache Spark для алгоритмів ALS та SARplus, крім етапу завантаження даних , за винятком етапу завантаження даних.

3.5 Особливості виконання програмного експериментального дослідження в середовищі Apache Spark

Перед вимірюванням часу формування рекомендацій виконується попередній прогрів обчислювального середовища на обмеженій підмножині користувачів. Це дозволяє усунути накладні витрати, пов’язані з ініціалізацією обчислень і доступом до даних, та забезпечити більш стабільні результати вимірювань. Описана логіка реалізована в наступному фрагменті:

```
warm_users                                                    =
test_users.limit(10).persist(StorageLevel.MEMORY_ONLY)
_ = warm_users.count()
warm_history      =      history_sdf.join(broadcast(warm_users),
on=DEFAULT_USER_COL, how="inner")
warm_history_cols    =    [DEFAULT_USER_COL,    DEFAULT_ITEM_COL,
DEFAULT_RATING_COL]
if use_timestamp:
    warm_history_cols.append(DEFAULT_TIMESTAMP_COL)
warm_sar = sarplus_model.recommend_k_items(
    warm_history.select(*warm_history_cols),
    top_k=min(10, int(TOPK_REQ)),
    remove_seen=True,
    use_cache=True,
```

```
)
_ = warm_sar.count()
warm_users.unpersist()
```

Далі виконується завантаження підготовленого набору даних із розподіленої файлової системи та перевірка його структури згідно з наведеним кодом:

```
df_sdf = self.spark.read.parquet(dataset.path)

req_cols = [DEFAULT_USER_COL, DEFAULT_ITEM_COL,
DEFAULT_RATING_COL]
missing = [c for c in req_cols if c not in df_sdf.columns]
if missing:
    raise ValueError(f"Expected columns in dataset
```

У цьому фрагменті дані зчитуються з розподіленого сховища HDFS у колонковому форматі, що забезпечує ефективну обробку великих обсягів інформації. Додатково виконується перевірка наявності необхідних полів, що дозволяє запобігти помилкам під час подальшої обробки.

Особливістю реалізації є відкладений характер обчислень, за якого операції виконуються лише під час фактичного звернення до результату. У зв’язку з цим застосовується примусове виконання обчислень і використання збереження проміжних результатів у пам’яті, що дозволяє коректно вимірювати час і зменшити кількість повторних обчислень.

Таким чином, застосування попереднього прогріву, перевірки структури даних і збереження проміжних результатів забезпечують стабільність експериментів і підвищує достовірність оцінювання обчислювальної ефективності.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Аналіз якості формування та впорядкування рекомендацій

На рисунку 4.1 подано значення метрик якості формування та впорядкування рекомендацій для розглянутих алгоритмів на наборах даних MovieLens 100K та Jester для верхньої частини рекомендаційного списку довжиною 10.

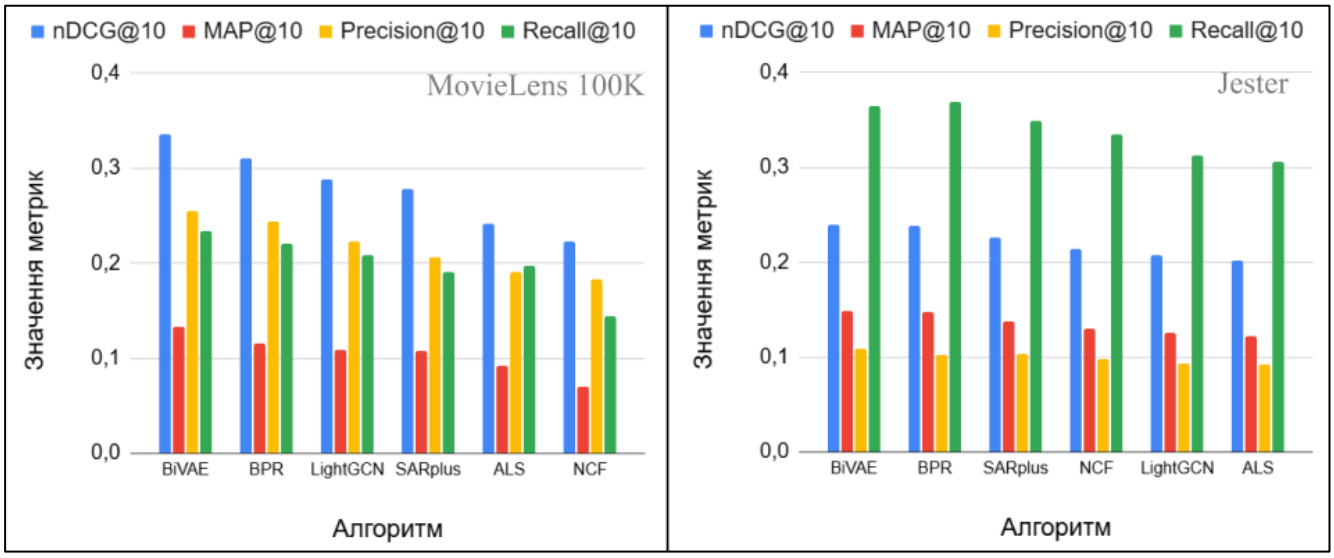

Рисунок 4.1 – Значення метрик якості алгоритмів для наборів даних MovieLens 100K та Jester

Найкращі значення метрик спостерігаються на наборах даних MovieLens 100K та Jester, причому в першому кращі значення має nDCG та точність, тоді як у другому – MAP та повнота.

Аналогічно на рисунку 4.2 наведено результати для наборів даних Amazon Reviews'23 Movies and TV 25-core та Video Games 15-core, на яких отримано найгірші результати.

Це свідчить про те, що якість рекомендацій залежить від набору даних, проте розрідженість (за умови її високого рівня в усіх наборах із незначними відмінностями) та розмір набору даних не є визначальними факторами. Зокрема, результати метрик, отримані для Video Games 15-core, є кращими, ніж для Movies

and TV 25-core, хоча перший набір має вищий рівень розрідженості. Аналогічний висновок можна зробити й для пари MovieLens 100K та Jester, де спостерігається подібна тенденція, але для різних метрик. Водночас кращі результати отримано на найбільших наборах даних: наприклад, MovieLens 100K, який є найбільшим серед розглянутих і більш ніж у два рази перевищує Video Games 15-core.

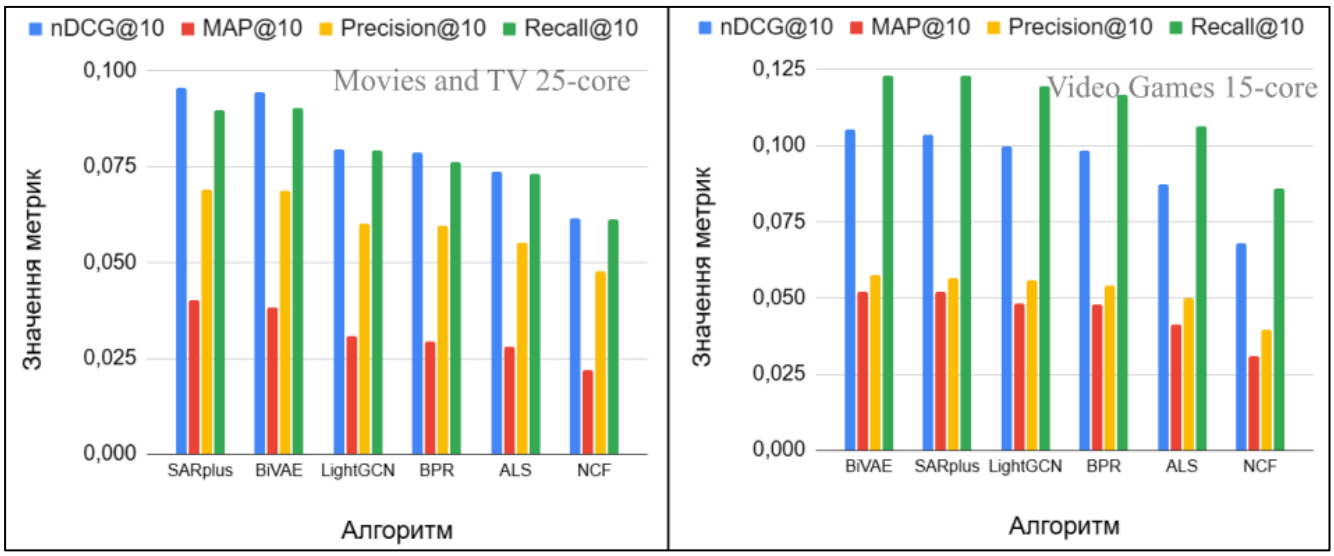

Рисунок 4.2 – Значення метрик якості алгоритмів для наборів даних Amazon Reviews'23 Movies and TV 25-core та Video Games 15-core

Отже, відмінності у якості рекомендацій не узгоджуються ані з рівнем розрідженості, ані з розміром наборів даних, що вказує на наявність додаткових факторів впливу. Це дає підстави припустити вплив додаткових чинників, пов'язаних зі структурою взаємодій між користувачами та об'єктами, зокрема:

– розподіл активності користувачів (концентрація взаємодій серед окремих користувачів);

– розподіл популярності об'єктів (ступінь домінування окремих об'єктів);

– щільність локальних перетинів взаємодій між користувачами та об'єктами;

– наявність груп користувачів із подібними інтересами.

Узагальнення результатів за всіма наборами даних показує, що найякісніші рекомендації формує алгоритм BiVAE, тоді як NCF у більшості випадків демонструє найнижчі або близькі до найнижчих показники. Найбільш стабільними

серед інших є алгоритми LightGCN і ALS, які посідають третє та п’яте місця відповідно в більшості наборах даних. Водночас SARplus демонструє вищу якість в умовах більшої розрідженості порівняно з іншими алгоритмами, що свідчить про ефективність підходів, заснованих на локальній подібності користувачів та об’єктів, а BPR, натомість, демонструє зниження позицій у рейтингу відносно інших алгоритмів.

Таким чином, встановлено найбільш придатний алгоритм колаборативної фільтрації серед розглянутих за якістю формування та впорядкування рекомендацій для побудови рекомендаційної системи. Проте ефективність його застосування залежить від характеристик структури взаємодій між користувачами та об’єктами набору даних, які визначають здатність алгоритмів виявляти закономірності, зокрема й у великих даних.

4.2 Аналіз обчислювальної ефективності

У середовищі, побудованому на базі WSL, найбільш показовими є результати, отримані для конфігурації з одним обчислювальним ядром, оскільки вони дозволяють оцінити власну обчислювальну складність алгоритмів.

Для набору даних MovieLens 100K (рисунок 4.3) найменший загальний час демонструє алгоритм LightGCN. Далі за зростанням загального часу розташовуються BPR, SARPlus, ALS, NCF та BiVAE. Такий розподіл пояснюється особливостями співвідношення часу підготовки та часу формування рекомендацій для кожного алгоритму.

Зокрема, LightGCN випереджає інші алгоритми насамперед завдяки дуже малому часу формування рекомендацій. BPR і SARPlus демонструють порівняно низький загальний час переважно за рахунок невисоких витрат на етапі підготовки. Для ALS і NCF спостерігається найбільш рівномірний розподіл обчислювальних витрат між етапами підготовки та формування рекомендацій, що забезпечує

збалансовану структуру їхнього загального часу. Найбільш витратним виявляється BiVAE, причому основна частина обчислювальних витрат припадає на етап підготовки.

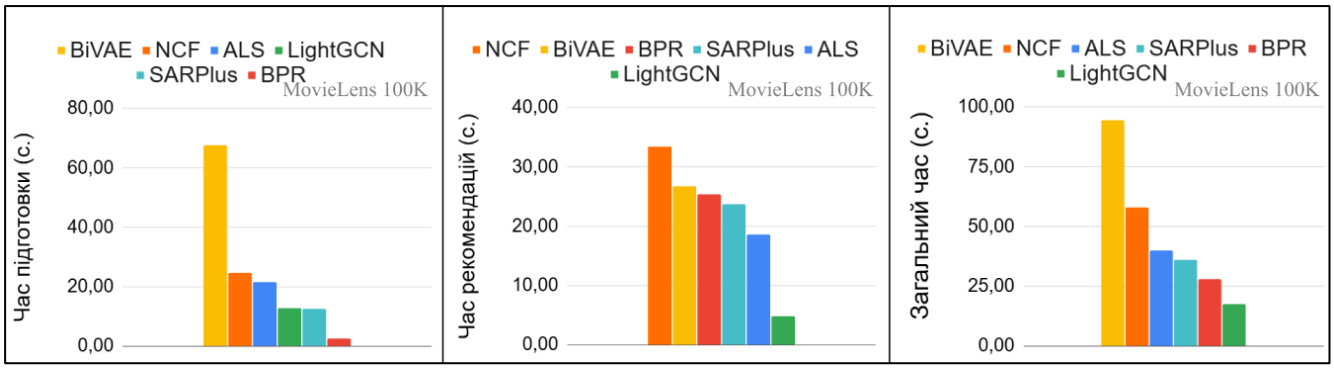

Рисунок 4.3 – Часові витрати алгоритмів (1 ядро; набір MovieLens 100K)

При збільшенні обсягу даних (рисунок 4.4) ці закономірності посилюються.

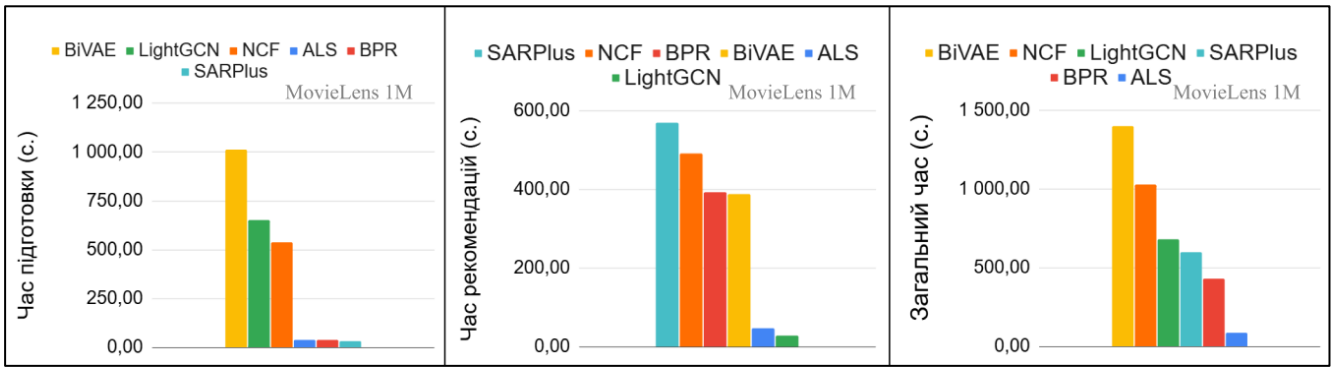

Рисунок 4.4 – Часові витрати алгоритмів (1 ядро; набір MovieLens 1M)

Водночас змінюється і лідер – ним стає алгоритм ALS, який демонструє найменший загальний час. Зокрема, його результат приблизно у 5 разів кращий порівняно з наступним за швидкістю алгоритмом – BPR. Для алгоритмів BPR і SARPlus основним обмежувальним фактором стає етап формування рекомендацій, тоді як для LightGCN і BiVAE найбільш витратним є етап підготовки моделі. Алгоритм NCF, у свою чергу, характеризується високими обчислювальними витратами на обох етапах.

При значному переважанні кількості користувачів над кількістю об’єктів у наборі даних Jester (рисунок 4.5) низька ефективність алгоритму NCF ще посилюється: етап підготовки стає вузьким місцем і займає приблизно у 10 разів більше часу, ніж навіть у BiVAE.

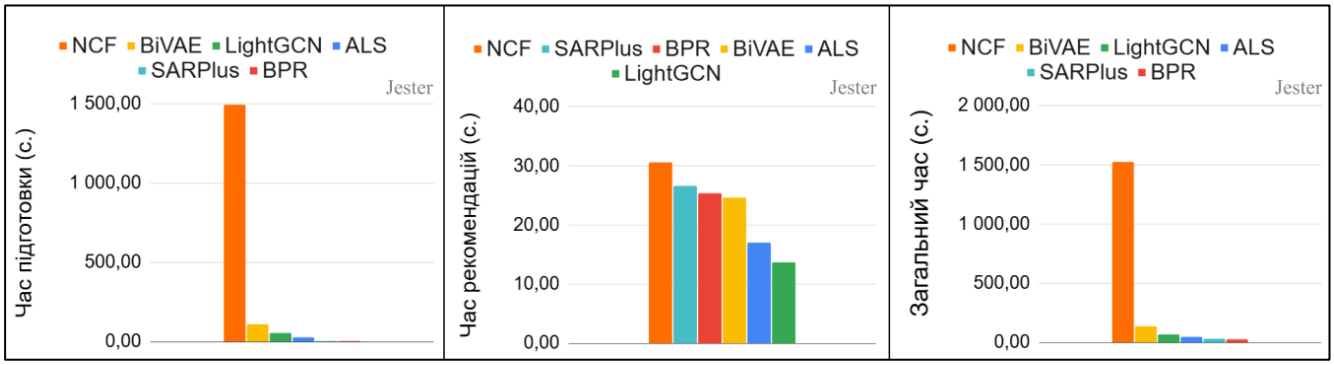

Рисунок 4.5 – Часові витрати алгоритмів (1 ядро; набір Jester)

Отже, з огляду на алгоритмічну складність, ALS можна розглядати як найбільш репрезентативний та збалансований вибір.

Порівняння конфігурацій із 1, 3 і 6 ядрами свідчить (рисунки 4.6-4.7), що найбільш впливовим є фактор збільшення кількості ядер для комбінації ALS і SARplus.

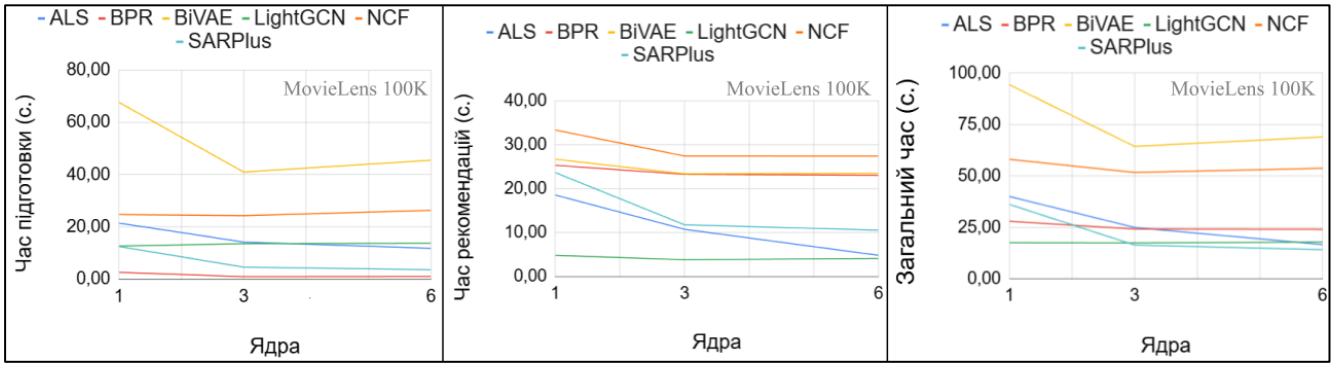

Рисунок 4.6 – Часові витрати алгоритмів (1, 3, 6 ядер; MovieLens 100K)

Для MovieLens 100K їхній загальний час скорочується приблизно у 2 рази, причому в ALS найбільше зменшується час рекомендацій, тоді як у SARplus – час підготовки. Для BPR час скорочується лише приблизно у 1,2 рази, навіть попри

вбудовану підтримку використання декількох ядер. Для BiVAE, NCF та LightGCN скорочення часу також не є значним і коливається в діапазоні до 1,4 раза, причому для двох останніх воно наближається до нуля. Це свідчить про те, що ефект прискорення при вертикальному масштабуванні (збільшенні кількості ядер на одній машині) в межах однієї системи суттєво залежить від внутрішньої структури обчислень конкретного алгоритму.

Аналогічна тенденція зберігається і для MovieLens 1M (рисунок 4.7).

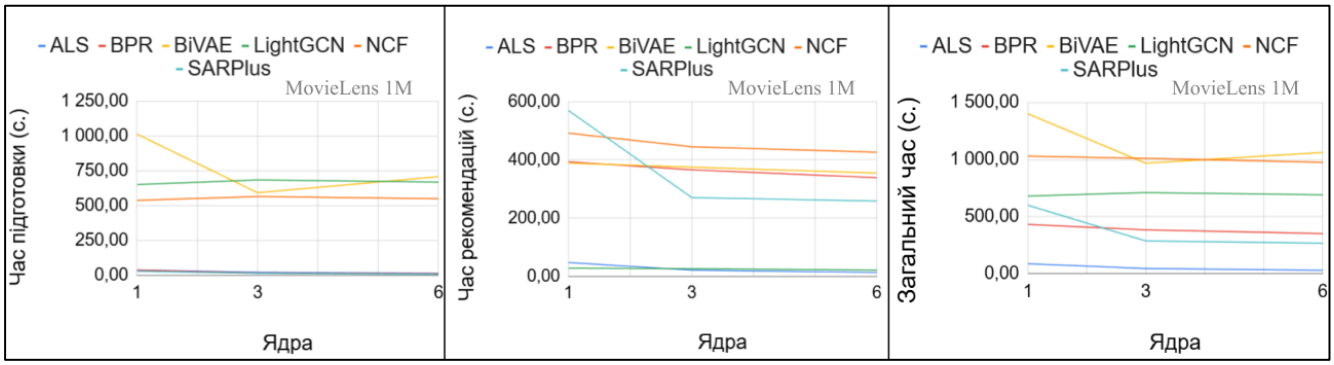

Рисунок 4.7 – Часові витрати алгоритмів (1, 3, 6 ядер; MovieLens 1M)

BiVAE отримує найбільший виграш від використання GPU, тоді як для NCF і LightGCN цей ефект стає помітним лише зі збільшенням обсягу даних (рисунок 4.8).

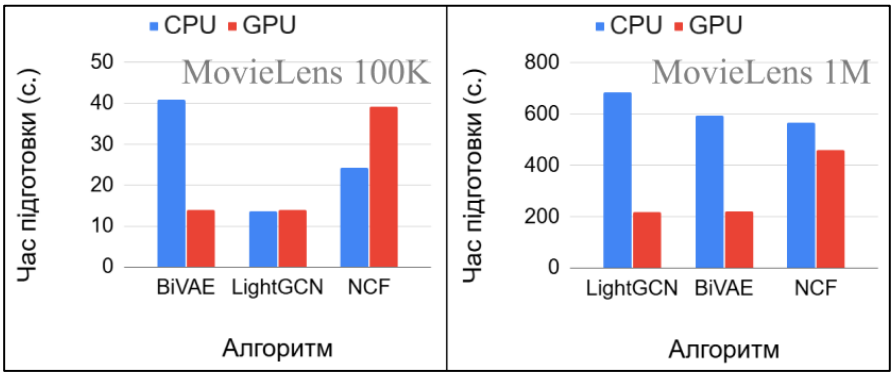

Рисунок 4.8 – Час підготовки (3 ядра та GPU)

Водночас покращення для NCF є менш вираженими порівняно з LightGCN і BiVAE. Використання GPU є доцільним насамперед для алгоритмів із дорогим

етапом навчання нейромережевої моделі, хоча ступінь такої доцільності передусім залежить від особливостей самого алгоритму.

Результати використання розподіленого обчислювального середовища підтверджують, що прискорення оброблення даних може досягатися не лише шляхом вертикального масштабування ресурсів однієї обчислювальної системи, а й шляхом збільшення кількості обчислювальних вузлів (воркерів) у кластері (рисунок 4.9).

Для Apache Spark та алгоритмів SARplus і ALS доведено наступне:

– найбільший вплив на обчислювальну ефективність має збільшення кількості робочих вузлів (горизонтальне масштабування);

– вертикальне масштабування зменшує часові витрати меншою мірою, ніж горизонтальне;

– зміна обсягу оперативної пам’яті в досліджених конфігураціях практично не впливає на результати.

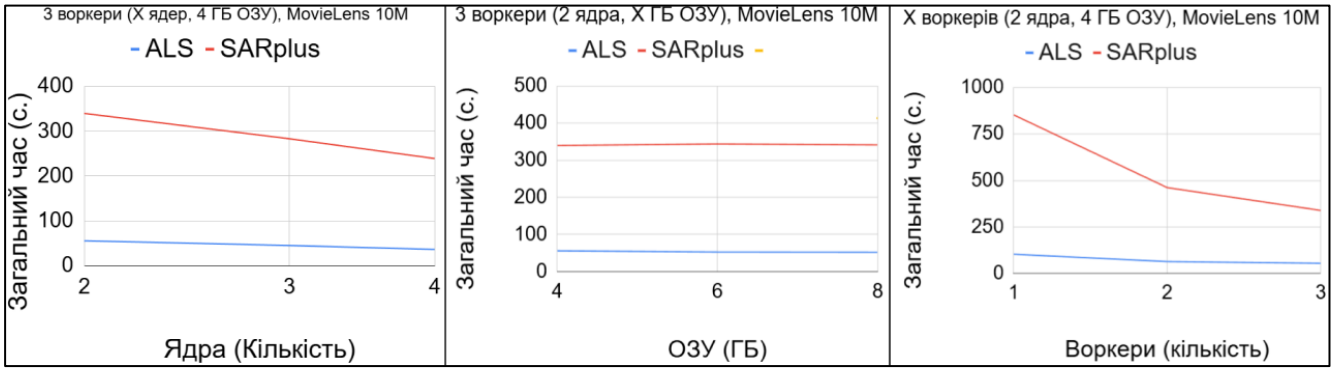

Рисунок 4.9 – Загальні часові витрати алгоритмів ALS та SARplus на різних конфігураціях Apache Spark

Масштабування за обсягом набору даних підкреслює принципову різницю в обчислювальній ефективності між ALS і SARplus (рисунок 4.10).

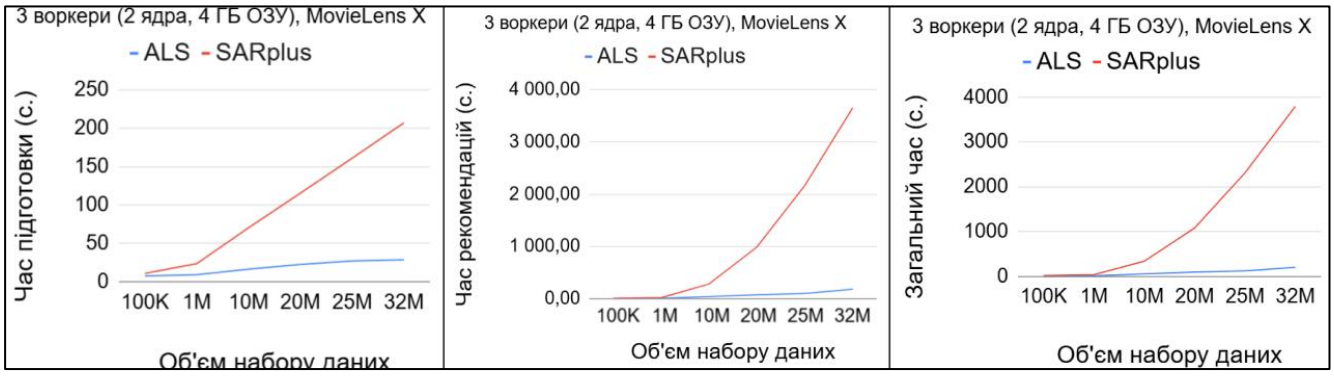

Рисунок 4.10 – Загальні часові витрати алгоритмів ALS та SARPlus для наборів даних різного об’єму в Apache Spark

Для ALS час зростає практично лінійно, тоді як для SARplus спостерігається зростання, наближене до квадратичного. Причому час підготовки для SARplus може перевищувати відповідний показник для ALS приблизно у 10 разів, тоді як час формування рекомендацій – уже до 20 разів. Наприклад, загальний час на наборі даних MovieLens 32M для SARplus сягає близько 1 год., тоді як для ALS становить менше ніж 5 хв. Це означає, що зі зростанням обсягу даних ALS зберігає передбачувану масштабованість, тоді як SARplus швидко втрачає обчислювальну придатність для реалізації у великомасштабному сценарії.

Це також дає підстави зробити висновок, що навіть за однакових умов обчислювального середовища ефективність рекомендаційних систем для оброблення великих даних визначається насамперед алгоритмічними властивостями, а не обчислювальною інфраструктурою (середовищем).

Таким чином, отримані результати дослідження в усіх середовищах свідчать, що з погляду на обчислювальну ефективність найбільш ефективним алгоритмом для оброблення великих даних є ALS, оскільки він поєднує передбачувану масштабованість, помірні часові витрати на обох етапах і найкращу реакцію на збільшення обчислювальних ресурсів.

4.3 Аналіз впливу методу WRRF на якість рекомендаційних систем

4.3.1 Аналіз впливу методу WRRF на якість рекомендацій парних комбінацій агрегування

На рисунку 4.11 наведено значення метрик якості формування та впорядкування рекомендацій для розглянутих пар алгоритмів на всіх наборах даних для верхньої частини рекомендаційного списку для 10 елементів.

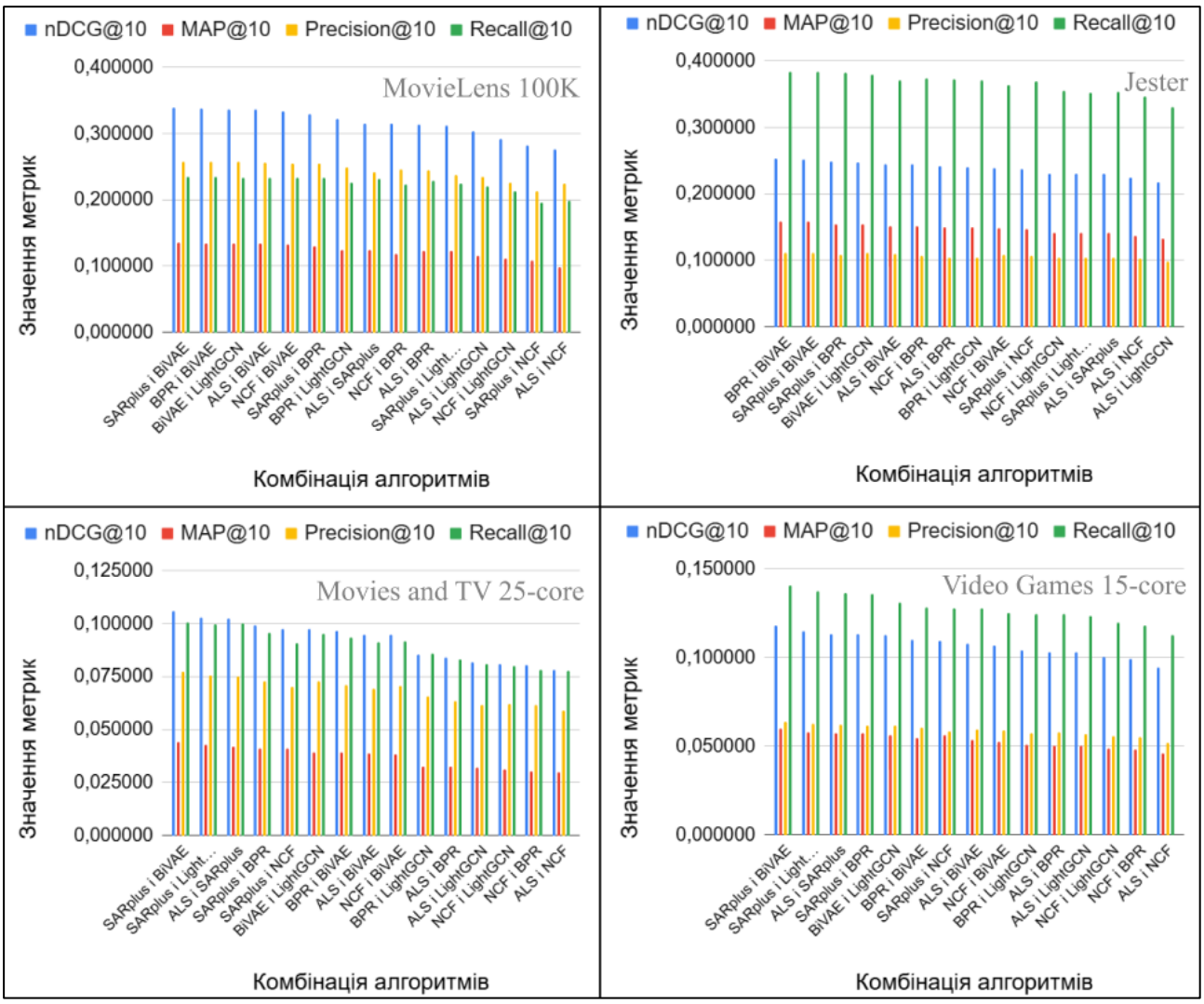

Рисунок 4.11 – Значення метрик якості пар алгоритмів

Найкращою комбінацією на більшості наборів даних є SARplus і BiVAE, тоді як найгіршою – ALS і NCF, що узгоджується з базовою якістю цих алгоритмів

окремо. Для набору даних Jesker ситуація подібна, проте зазначені найкраща та найгірша комбінації займають відповідно друге та передостаннє місця. Результати, отримані для найкращих комбінацій, формують підсумкове впорядкування, яке перевершує результати будь-якого з базових алгоритмів.

Для оцінювання ефекту агрегування використано відносне відхилення значень метрик якості окремих алгоритмів від значень найкращого алгоритму в парі (рисунок 4.12), яке характеризує покращення (погіршення) результатів, за формулою:

$$\Delta M_{\%} = \left(\frac{M^{WRRF}}{\max\left(M^{(1)}, M^{(2)}\right)} - 1 \right) \cdot 100\%,$$

де M^{WRRF} – значення відповідної метрики якості для агрегованого списку рекомендацій, сформованого методом WRRF;

$M^{(1)}$ і $M^{(2)}$ – значення цієї метрики для першого та другого алгоритмів.

Для набору MovieLens 100K приріст за відносним відхиленням за метриками nDCG і MAP спостерігається у 14 із 15 пар (ALS і NCF, ALS і SARplus, SARplus і LightGCN), причому поодинокі випадки зниження мають незначну величину.

Для набору Jester незначне погіршення зафіксовано лише для однієї пари – NCF і BiVAE.

Для набору Amazon Reviews'23 Movies and TV 25-core усі 15 пар алгоритмів продемонстрували приріст за відносним відхиленням метрик nDCG, точності та повноти, тоді як незначне зниження за метрикою MAP (0,52%) зафіксовано лише для комбінації NCF і BiVAE.

Для Amazon Reviews'23 Video Games 15-core усі 15 пар забезпечили покращення за метриками nDCG, MAP і точності, тоді як за повнотою лише одна комбінація – NCF і LightGCN – не продемонструвала приросту, але й не призвела до погіршення результатів.

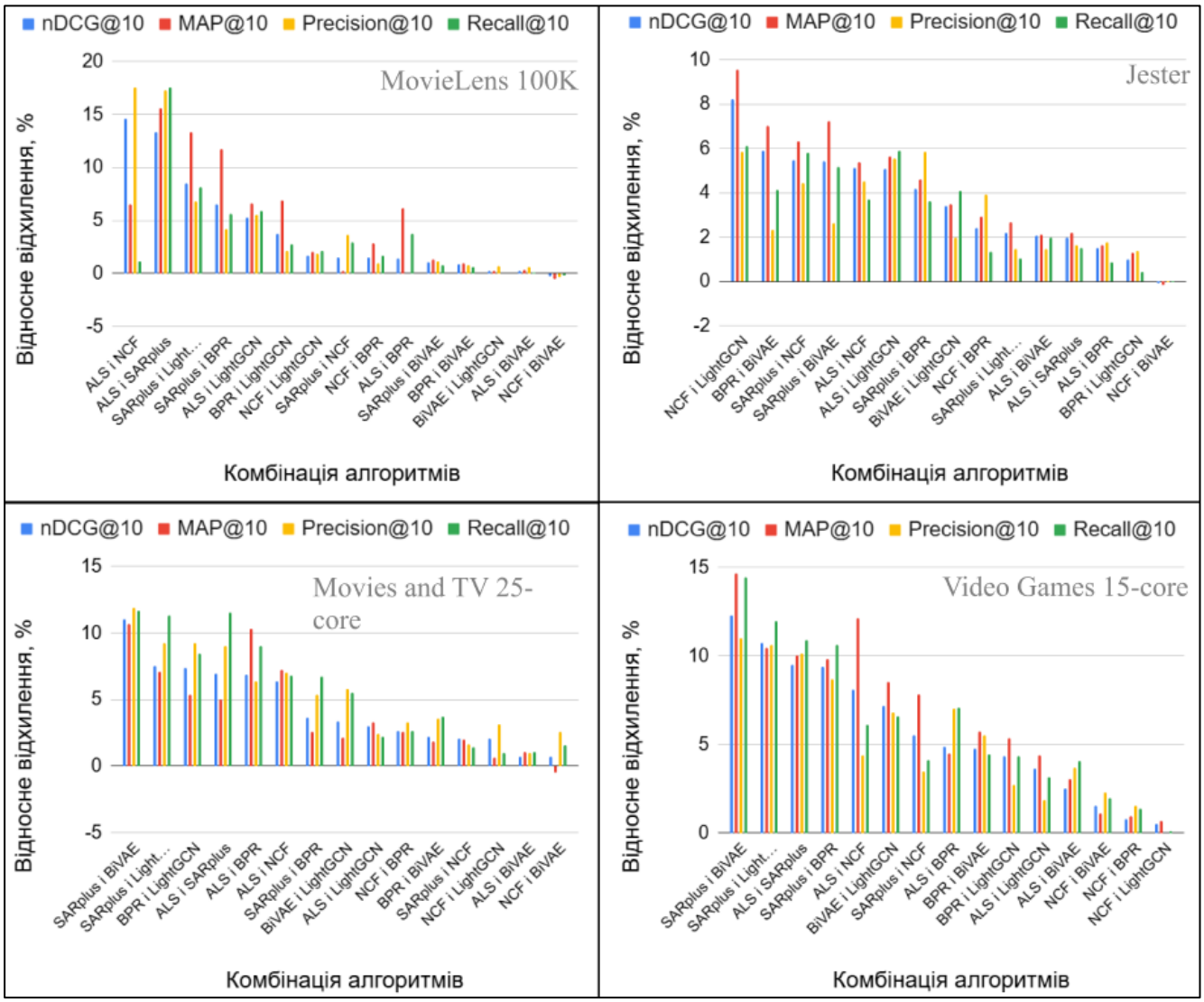

Рисунок 4.12 – Відносне відхилення значень метрик якості пар алгоритмів

Узагальнюючи отримані результати для всіх наборів даних, можна зробити висновок, що найбільший ефект від агрегування, визначений як середнє арифметичне відносних відхилень за метрикою nDCG (рисунок 4.13), забезпечує комбінація ALS і NCF – 8,56 %.

Друге місце посідає комбінація ALS і SARplus – 7,95 %, третє – SARplus і BiVAE (7,49 %), четверте – SARplus і LightGCN (7,27 %), і п’яте – SARplus і BPR (5,95 %).

Найменші значення демонструють такі комбінації: останнє місце – NCF і BiVAE (0,46 %), передостаннє – ALS і BiVAE (1,38 %).

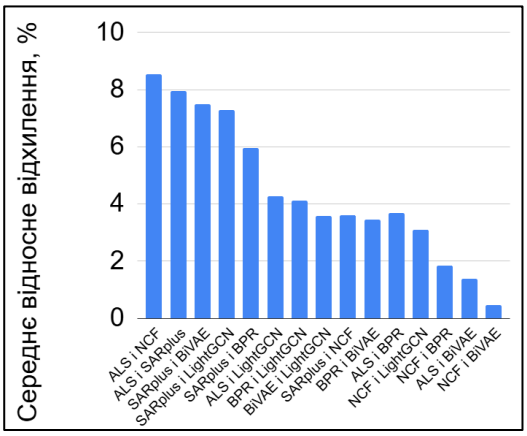

Рисунок 4.13 – Середнє відносне відхилення значень метрик якості пар алгоритмів

Згідно з цим рисунком наочно видно, що найбільший ефект досягається при поєднанні двох алгоритмів, які відрізняються методами побудови та підходами до впорядкування.

Водночас, коли один алгоритм значно переважає інший за метрикою nDCG, їхнє агрегування є менш ефективним. Слабкіший алгоритм у такій парі не додає цінної інформації, а лише виступає джерелом «шуму», що й погіршує загальний результат.

Це дає підстави сформувати припущення про наявність лінійної залежності між різницею в якості рекомендаційних списків за метрикою nDCG та іншими метриками якості, сформованих кращим і гіршим алгоритмами у парі, та значенням виграшу або програшу від їхнього агрегування. Щоб перевірити це припущення, було сформульовано та перевірено такі статистичні гіпотези:

– нульова гіпотеза (H_0): відсутня лінійна залежність між різницею якості рекомендаційних списків, сформованих кращим і гіршим алгоритмами у парі (за використаними метриками оцінювання), та значенням виграшу або програшу від їхнього агрегування;

– альтернативна гіпотеза (H_1): існує лінійна залежність між цими величинами.

Для перевірки висунутих гіпотез використано коефіцієнт кореляції Пірсона. Для кожної метрики значення інтерпретується таким чином: додатне значення свідчить про прямий лінійний зв’язок між різницею у якості базових алгоритмів і

виграшем від агрегування (зі збільшенням розриву виграш зростає); від'ємне – про обернений лінійний зв'язок (зі збільшенням різниці виграш зменшується); значення, близьке до нуля, вказує на відсутність лінійної залежності між цими величинами.

Для визначення коефіцієнта кореляції Пірсона обчислюється різниця між метриками якості рекомендаційних списків, сформованих базовими алгоритмами у відповідній парі (формула 4.1), виграш (або програш) якості їх агрегованого результату (формула 4.2), середні значення відповідних величин (формули 4.3-4.4) і коефіцієнт Пірсона (формула 4.5):

$$\Delta M_{base,i} = \max\left(M_i^{(1)}, M_i^{(2)}\right) - \min\left(M_i^{(1)}, M_i^{(2)}\right), \tag{4.1}$$

$$\Delta M_{abs,i} = M_i^{WRRF} - \max\left(M_i^{(1)}, M_i^{(2)}\right), \tag{4.2}$$

$$\overline{\Delta M_{base}} = \frac{1}{N}\sum_{i=1}^{N} \Delta M_{base,i}, \tag{4.3}$$

$$\overline{\Delta M_{abs}} = \frac{1}{N}\sum_{i=1}^{N} \Delta M_{abs,i}, \tag{4.4}$$

$$r_M = \frac{\sum_{i=1}^{N}(\Delta M_{base,i} - \overline{\Delta M_{base}})(\Delta M_{abs,i} - \overline{\Delta M_{abs}})}{\sqrt{\sum_{i=1}^{N}(\Delta M_{base,i} - \overline{\Delta M_{base}})^2} \cdot \sqrt{\sum_{i=1}^{N}(\Delta M_{abs,i} - \overline{\Delta M_{abs}})^2}}, \tag{4.5}$$

де N – загальна кількість спостережень (пар алгоритмів на всіх наборах даних).

Для метрик точності, повноти, MAP та nDCG значення коефіцієнта Пірсона становлять відповідно –0.1051, –0,327, –0,241 та –0,3818.

Отже, отримані результати свідчать про наявність від'ємної лінійної залежності між досліджуваними величинами різних значень залежно від метрики. Найбільш виражена залежність спостерігається для метрики nDCG, менш виражена – для повноти, ще слабша – для MAP, а найменш виражена – для точності. Загалом це означає, що зі збільшенням різниці в якості базових алгоритмів виграш від агрегування зменшується, але характер цієї залежності відрізняється за значеннями різних метрик.

4.3.2 Аналіз впливу методу WRRF на якість рекомендацій комбінацій агрегування за повним перебором

В порівняння з попарною, повна комбінація (рисунок 4.14) переважно забезпечує лише незначне покращення якості рекомендацій порівняно з попарною: метрики nDCG (0,78–3,33 %), MAP (0,71–2,48 %), точність (0,76-2,84 %) та повнота (0,2–2,08 %).

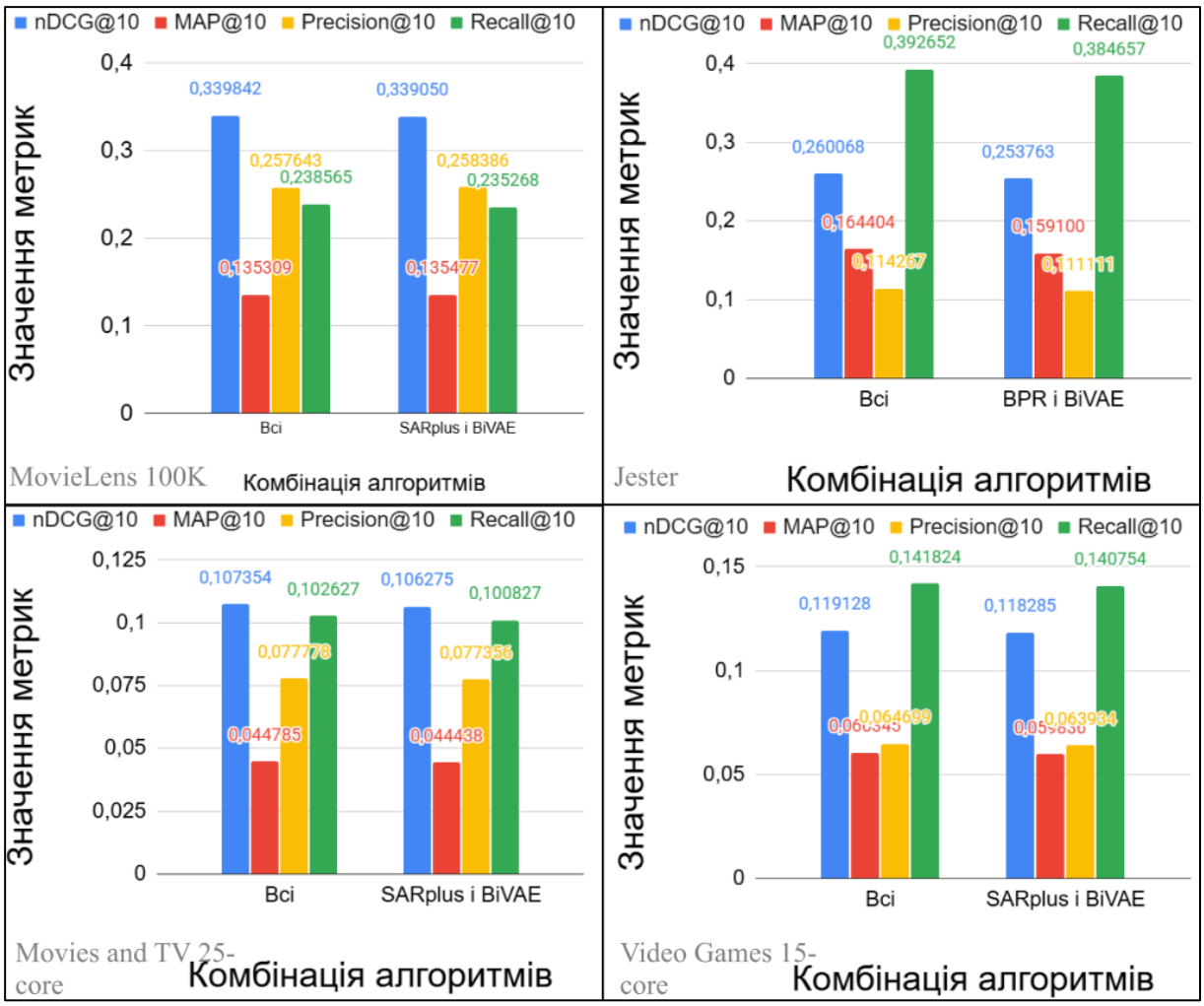

Рисунок 4.14 – Значення метрик якості повної комбінації та кращої пари алгоритмів

Це досягається ціною збільшення обчислювальних витрат, оскільки передбачається навчання більшої кількості моделей (6 проти 2). Незначні

погіршення на наборі даних MovieLens 100K за метриками nDCG (0,12 %) та точністю (0,29 %) можуть бути пояснені використанням випадкового пошуку, який не дозволяє знайти оптимальну комбінацію параметрів методу.

До того ж визначено, що найбільший внесок у повну комбінацію (рисунок 4.15) забезпечують алгоритми, які формують найкращу пару при попарному комбінуванні (BiVAE та SARplus або BPR): частка цього внеску варіюється в діапазоні 54–85%. Це свідчить, що інші досліджувані алгоритми генерують менше нових релевантних об’єктів – переважно вони повторюють ті, що вже включені як найефективніші.

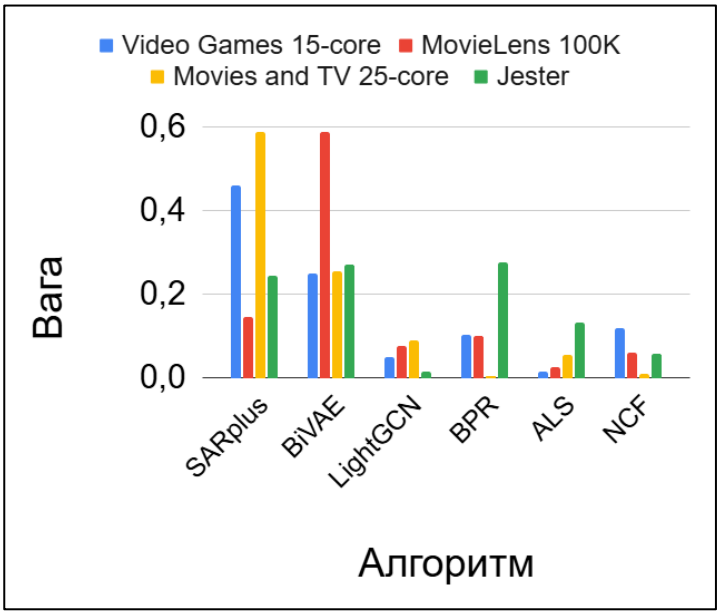

Рисунок 4.15 – Внесок кожного алгоритму в кінцевий список рекомендацій

Таким чином, застосування повної комбінації є доцільнішим для стабілізації якості рекомендацій, ніж для покращення кінцевого рекомендаційного списку порівняно з попарними комбінаціями.

Загалом метод WRRF демонструє здатність до стабільного покращення якості рекомендацій як у задачі відбору релевантних об’єктів, так і у задачі їх впорядкування у верхній частині рекомендаційного списку. Результати експериментів свідчать, що ефективність його застосування значною мірою визначається рекомендаційними списками, сформованими базовими алгоритмами.

Водночас характеристики набору даних впливають на ефективність методу опосередковано, через їхній вплив на результати окремих алгоритмів.

4.3.3 Аналіз впливу методу WRRF на обчислювальну ефективність

Аналогічно до аналізу обчислювальної ефективності алгоритмів, при дослідженні впливу методу WRRF на обчислювальну ефективність спочатку аналізуються результати, отримані в середовищі, побудованому на базі WSL, для конфігурації з 1 обчислювальним ядром.

Результати показують, що час агрегування рекомендаційних списків є на порядок меншим за витрати на підготовку і формування рекомендацій на основі комбінації алгоритмів (рисунки 4.16-4.17).

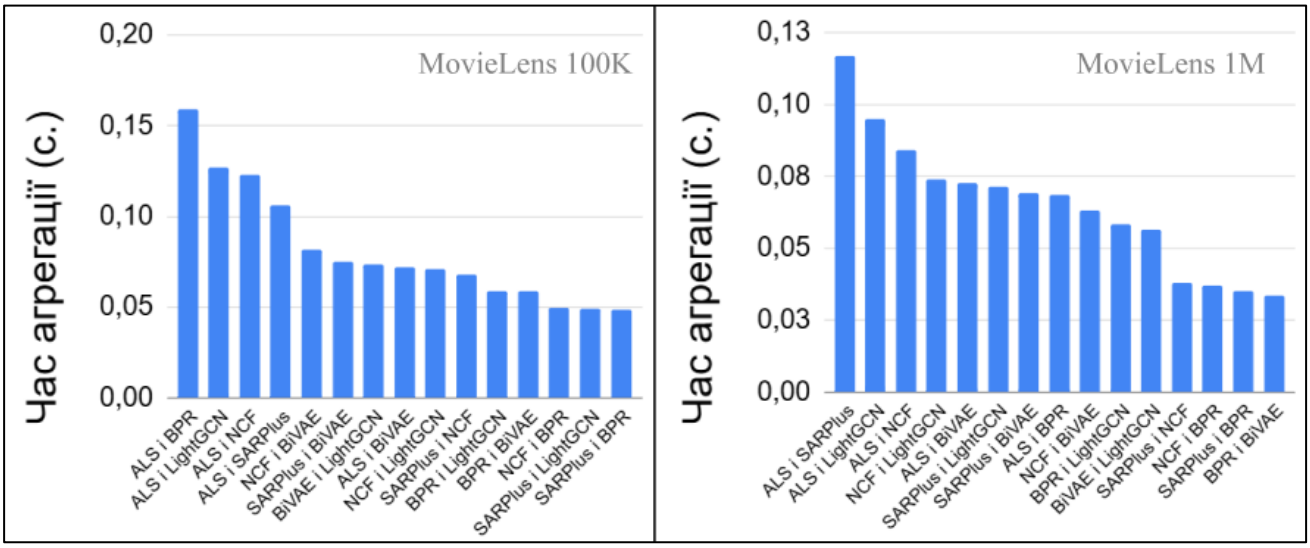

Рисунок 4.16 – Час, витрачений на агрегацію рекомендацій алгоритмів (1 ядро)

Причому на наборі даних MovieLens 100K загальний час перевищує час агрегації приблизно у 1 000 разів, а на 1M – приблизно у 20 000 разів. Водночас на цих наборах даних час агрегації залишається практично сталим.

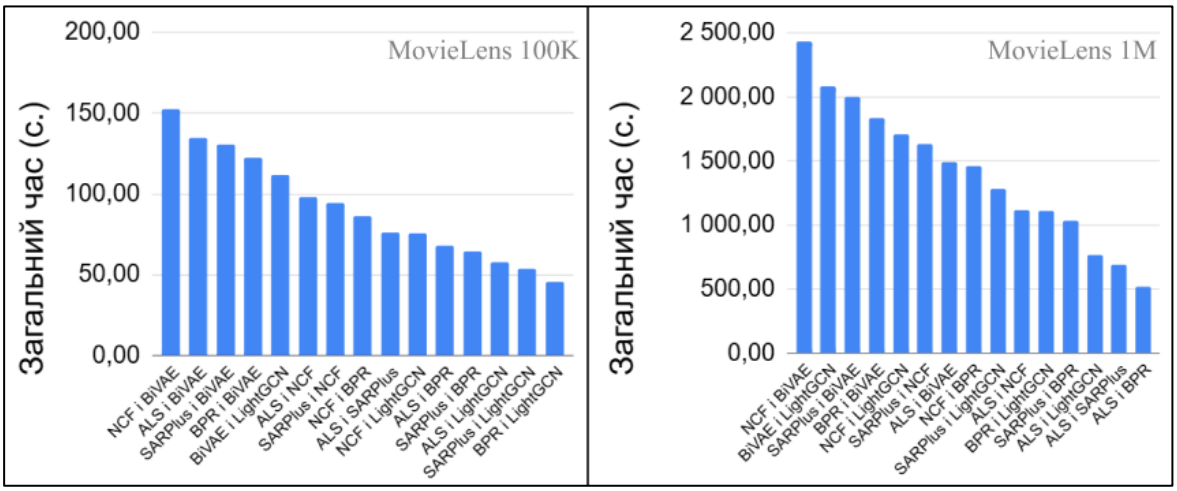

Рисунок 4.17 –Загальні часові витрати для парної комбінації алгоритмів (1 ядро)

Такі ж результати отримані і в розподіленому обчислювальному середовищі Apache Spark на різних його конфігураціях (рисунки 4.18–4.19).

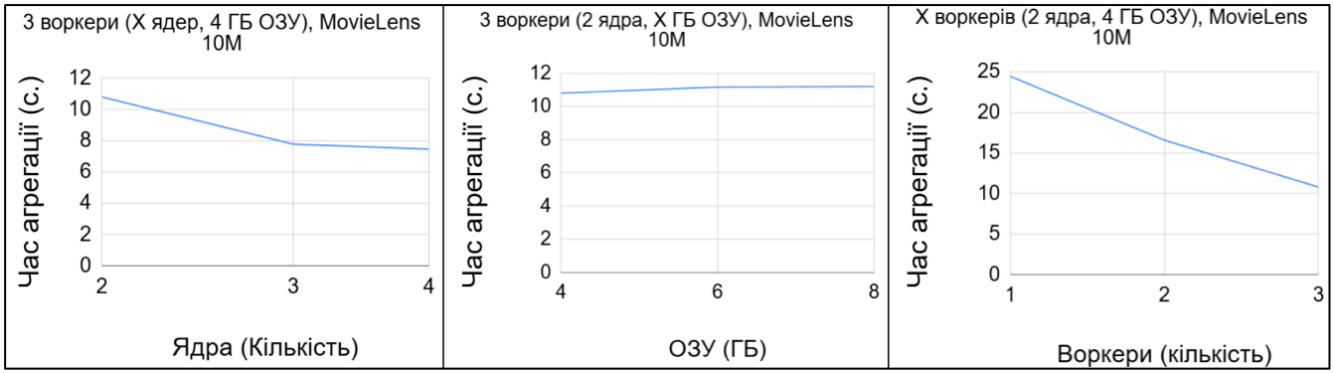

Рисунок 4.18 – Час на агрегацію рекомендаційних списків алгоритмів ALS та SARplus на різних конфігураціях Apache Spark

Отже, загальний час не перевищує час агрегації у стільки ж разів, як у середовищі WSL, оскільки додатково присутні накладні витрати самого обчислювального середовища Apache Spark, пов’язані з ініціалізацією задач, розподілом обчислень між вузлами та передачею даних між ними.

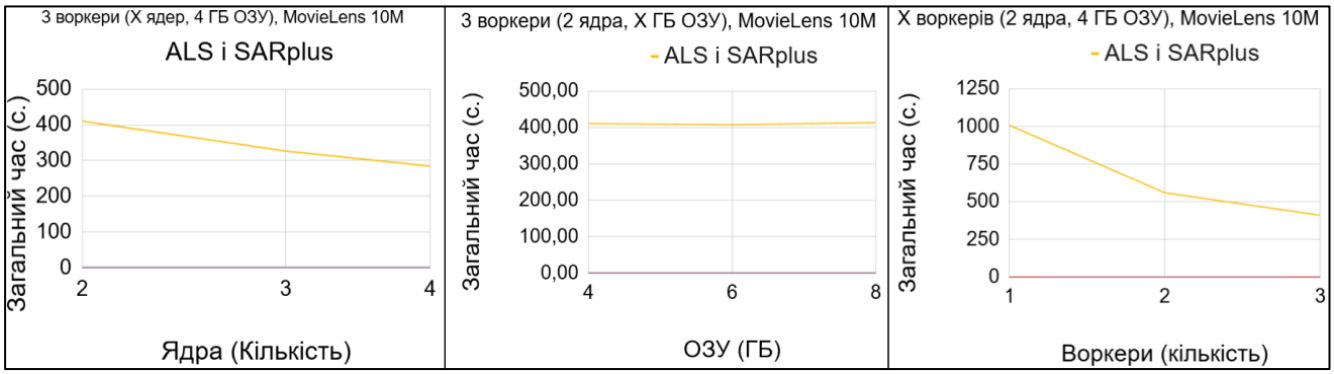

Рисунок 4.19 – Загальні часові витрати алгоритмів ALS та SARplus та їх комбінації на різних конфігураціях Apache Spark

При збільшенні обсягу даних час агрегування для певної конфігурації все ж таки зростає (рисунок 4.20).

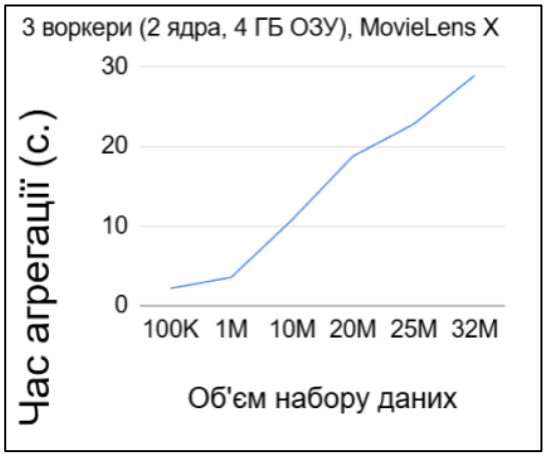

Рисунок 4.20 – Час агрегації рекомендаційних списків алгоритмів ALS та SARplus для різних обсягів наборів даних

Це пояснюється тим, що зі збільшенням набору даних зростає кількість користувачів і сформованих рекомендаційних списків, які підлягають агрегації, що призводить до збільшення обсягу обчислень під час їх агрегації та повторного впорядкування.

Водночас, при порівнянні із загальним часом виконання зі збільшенням обсягу даних (рисунок 4.21) час агрегування залишається незначним. Отже, навіть

у великомасштабному сценарії WRRF не є основним джерелом обчислювальних витрат.

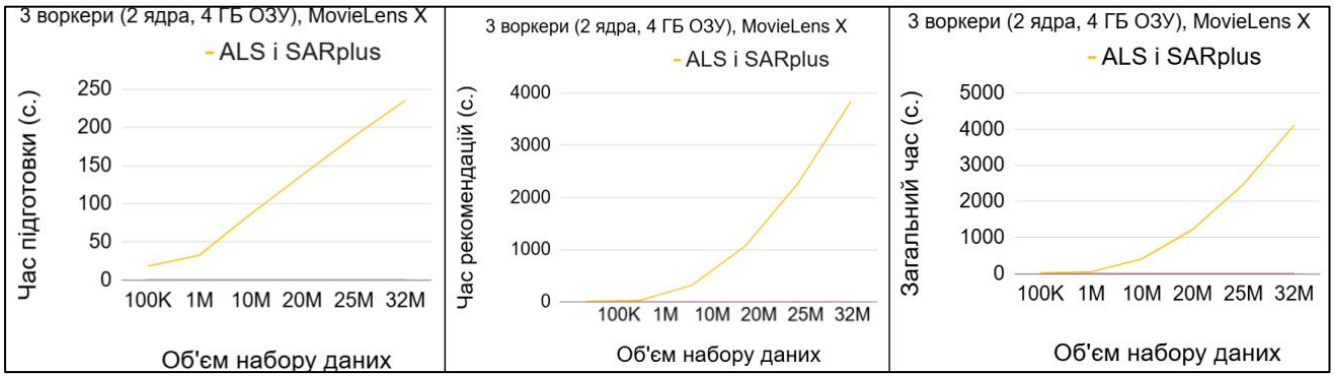

Рисунок 4.21 – Загальні часові витрати алгоритмів ALS та SARPlus і їх комбінацій для наборів даних різного розміру в Apache Spark

Таким чином, застосування WRRF не створює критичного додаткового обчислювального навантаження, а загальний час роботи комбінації визначається насамперед складністю базових алгоритмів.

4.3.4 Аналіз доцільності застосування WRRF в умовах великих даних з огляду на отримані покращення

Для аналізу результатів експериментів та оцінювання виправданості збільшення обчислювальних витрат відносно покращення якості рекомендаційних списків використано результати, отримані в розподіленому обчислювальному середовищі Apache Spark на наборі даних MovieLens 100K.

Вибір саме цих результатів обґрунтований темою роботи, оскільки для обробки великих даних зазвичай використовується розподілене середовище, яке, згідно з раніше отриманими результатами, додає додаткові накладні витрати при застосуванні методу WRRF.

Вибір набору даних зумовлений тим, що він демонструє найгірший випадок з точки зору співвідношення обсягу даних та впливу часу агрегації на загальний час виконання, оскільки саме на малих вибірках внесок накладних витрат агрегації є найбільш відчутним у відносному вимірі.

Це підтверджується аналізом частки часу агрегації у загальних часових витратах (таблиця 4.1). Для набору даних 100K вона становить приблизно 9,2 %, для 1M – близько 6,8 %, для 10M – 2,6 %, для 20M – 1,5 %, для 25M – 0,9 %, а для 32M – приблизно 0,7 %.

Таблиця 4.1 – Частки часу агрегації у загальних часових витратах алгоритмів

Набір даних	Загальні часові витрати комбінації алгоритмів ALS та SARplus (с.)	Час агрегації (с.)	Частка (%)
MovieLens 100K	23,73	2,18	9,19
MovieLens 1M	52,27	3,56	6,81
MovieLens 10M	410,46	10,8	2,63
MovieLens 20M	1224,22	18,78	1,53
MovieLens 25M	2474,03	22,83	0,92
MovieLens 32M	4107,33	28,93	0,70

В свою чергу, згідно з раніше отриманими результатами, відносне відхилення метрик якості алгоритмів залежить від різниці у значеннях метрик якості між кращим і гіршим алгоритмами. У свою чергу, ця різниця залежить від значень метрик якості цих алгоритмів, а значення метрик якості залежать не від розміру набору даних, а від його структури.

Оскільки серед усіх досліджуваних алгоритмів Apache Spark підтримує лише ALS та SARplus, і саме для них отримані результати, аналіз виконується на прикладі оцінювання впливу часу агрегації на загальні часові витрати цих алгоритмів. При цьому використовується відносне відхилення загальних часових витрат їхньої комбінованої роботи від суми індивідуальних часових витрат, яке характеризує збільшення або зменшення часу виконання, за формулою 4.6.

$$\Delta T_{full_{\%}} = \left(\frac{T_{full}(ALS, SARplus)}{T_{full}(ALS) + T_{full}(SARplus)} - 1 \right) \cdot 100\%, \quad (4.6)$$

де $T_{full}(ALS, SARplus)$ – загальні часові витрати для комбінації алгоритмів ALS і SARplus;

$T_{full}(ALS)$ – загальні часові витрати алгоритму ALS;

$T_{full}(SARplus)$ – загальні часові витрати алгоритму SARplus.

Отже, відносне відхилення часових витрат становить 10,12 % (збільшення часу виконання) і не перевищує відносного відхилення значень метрик якості, які демонструють загальне покращення результатів: MAP@10 – ↑15,57 %, nDCG@10 – ↑13,32 %, Precision@10 – ↑17,33 %, Recall@10 –↑17,60 %.

Зі збільшенням обсягу набору даних очікується зменшення відносної частки часових витрат, тоді як за незмінної загальної структури даних покращення результатів залишатиметься на тому ж рівні. Таким чином, використання методу WRRF в умовах великих даних є виправданим.

4.4 Узагальнення отриманих результатів

Серед розглянутих алгоритмів у контексті обробки великих даних у рекомендаційних системах на основі колаборативної фільтрації алгоритм ALS виступає найбільш практичним рішенням для їх побудови, оскільки забезпечує передбачувану масштабованість, ефективно використовує доступні обчислювальні ресурси та зберігає стабільний рівень якості порівняно з іншими алгоритмами.

Аналогічно, перспективною альтернативою є LightGCN за умови використання GPU, що дозволяє компенсувати його дорогий етап підготовки моделі, при цьому час формування рекомендацій не поступається відповідному етапу в

ALS, а якість рекомендацій залишається стабільною та перевищує результати, отримані за допомогою ALS.

З урахуванням того, що до LightGCN можуть бути застосовані загальні принципи розподілених обчислень для графових нейронних мереж (GNN) [51], час підготовки може бути додатково суттєво скорочений. Хоча аналогічні підходи можуть бути застосовані і до BiVAE, для нього залишається суттєвим обмеженням великий час формування рекомендацій, що навіть за умови забезпечення найвищої якості робить його використання обмеженим у високонавантажених системах.

Подібний недолік спостерігається також у BPR і SARplus, де саме етап формування рекомендацій стає вузьким місцем і може бути критичним у високонавантажених системах з великою кількістю користувачів, які потребують постійного отримання рекомендацій. NCF, своєю чергою, поступається іншим алгоритмам як за обчислювальною ефективністю, так і за якістю рекомендацій, що обмежує доцільність його використання.

Метод WRRF показав доцільність застосування як засобу підвищення якості рекомендацій без істотного додаткового обчислювального навантаження. За умови використання в межах рекомендаційної системи декількох алгоритмів колаборативної фільтрації, достатньо близьких за значеннями метрик якості, але різних за принципом побудови, ефект від використання методу WRRF підвищується. При цьому парне комбінування практично не поступається повному, але значною мірою зменшує обчислювальне навантаження.

Додатково, при збільшенні обсягу набору даних за незмінної їхньої загальної структури очікується, що покращення результатів залишатиметься на тому ж рівні, а виправданість використання методу WRRF зростатиме, оскільки відносне відхилення часових витрат зменшуватиметься.

Таким чином, для рекомендаційних систем колаборативної фільтрації в умовах оброблення великих даних визначальним є використання одного або декількох алгоритмів, що забезпечують збалансоване поєднання якості рекомендацій, обчислювальної ефективності, масштабованості та придатності до розподіленої обробки.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи проведено дослідження рекомендаційних систем на основі колаборативної фільтрації із застосуванням алгоритмів та їхніх комбінацій для оброблення великих даних. Проаналізовано предметну галузь, обґрунтовано методологію, виконано експериментальні дослідження та проаналізовані результати.

У ході роботи встановлено, що якість рекомендацій залежить не лише від обсягу та розрідженості даних, а й від структури взаємодій користувачів і об’єктів. Ефективне масштабування досягається передусім за рахунок розподілених обчислень, а результативність систем визначається переважно алгоритмічними властивостями алгоритмів. Використання GPU може пришвидшити навчання нейромережевих методів, однак ефект залежить від конкретного алгоритму.

За результатами дослідження визначено, що для задач оброблення великих даних найбільш доцільним є використання алгоритмів ALS або LightGCN. У випадках, коли пріоритетом є максимальна якість рекомендацій, доцільно застосовувати BiVAE, який стабільно демонструє найкращі результати серед окремих алгоритмів. Застосування методу WRRF у варіанті пар у більшості випадків покращує якість рекомендацій порівняно з окремими алгоритмами та практично не поступається повній комбінації. Найбільший ефект досягається при поєднанні близьких за якістю моделей, що відрізняються принципами побудови. При цьому витрати на агрегування залишаються незначними порівняно з витратами на підготовку і формування рекомендацій, що робить придатним його застосування в умовах великих даних.

Проведене дослідження узгоджується з вітчизняними [3] аналогами, оскільки ґрунтується на дослідженні рекомендаційних систем, та світовими [8, 12, 13] аналогами, оскільки спрямоване на вивчення колаборативної фільтрації в умовах оброблення великих даних. Водночас воно відрізняється постановкою задачі й обраною методологією її вирішення. Робота також відповідає науковим напрямам

кафедри комп’ютерного моделювання та інтелектуальних технологій у сфері систем підтримки прийняття рішень і розподілених систем.

За результатами виконання кваліфікаційної роботи підготовлено матеріали апробації основних результатів дослідження. У тезах конференції [9] представлено застосування методу RRF (Reciprocal Rank Fusion) для задач колаборативної фільтрації з акцентом на покращенні якості рекомендацій. У науковій статті [10] досліджено вплив методу WRRF на формування та впорядкування рекомендаційних списків, зокрема подано формалізацію методу та аналіз результатів на різних наборах даних і алгоритмах.

У подальших дослідженнях доцільно зосередитися на використанні розподілених обчислень для нейромережевих методів колаборативної фільтрації, зокрема із застосуванням GPU та графових нейронних мереж (GNN) в умовах оброблення великих даних. Також перспективним є дослідження багаторівневої агрегації, за якого поєднуються як базові, так і попередньо агреговані рекомендаційні списки.

Практичне значення отриманих результатів полягає у можливості їх використання для обґрунтованого вибору алгоритмів або їх комбінацій при побудові рекомендаційних систем для оброблення великих даних. Матеріали роботи можуть бути використані під час вивчення технологій оброблення великих даних та розподілених обчислень.

Пояснювальна записка до кваліфікаційної роботи оформлена згідно ДСТУ та методичними вказівками щодо розробки та оформлення кваліфікаційної роботи [52-54].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Roy D., Dutta M. A systematic review and research perspective on recommender systems. *Journal of Big Data*. 2022. Vol. 9, no. 1. P. 59. URL: https://doi.org/10.1186/s40537-022-00592-5 (date of access: 05.03.2026).

2. A Comprehensive Review of Recommender Systems: Transitioning from Theory to Practice. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2407.13699 (date of access: 25.03.2026).

3. Гребенюк М., Ситнікова П. Метод коеволюційного налаштування параметрів нечітких множин та вагових коефіцієнтів у рекомендаційних системах. *Наука і техніка сьогодні*. 2026. № 1 (55). С. 2118–2133. URL: https://doi.org/10.52058/2786-6025-2026-1(55)-2118-2133 (дата звернення: 05.03.2026).

4. Hu Y., Koren Y., Volinsky C. Collaborative Filtering for Implicit Feedback Datasets. *2008 Eighth IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, 15–19 December 2008. 2008. URL: https://doi.org/10.1109/icdm.2008.22 (date of access: 25.03.2026).

5. BPR: Bayesian Personalized Ranking from Implicit Feedback. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.1205.2618 (date of access: 25.03.2026).

6. Li Y. A Review of Collaborative Filtering Recommendation Systems. *European Journal of AI, Computing & Informatics*. 2025. Vol. 1, no. 1. P. 1-11. URL: https://pinnaclepubs.com/index.php/EJACI/article/view/8 (date of access: 05.03.2026).

7. Rajesh D. B., Kumar A. Collaborative filtering models: an experimental and detailed comparative study. *Scientific Reports*. 2025. Vol. 15, no. 1. P. 31667. URL: https://doi.org/10.1038/s41598-025-15096-4 (date of access: 15.03.2026).

8. Khouibiri N., Farhaoui Y., El Allaoui A. Design and Analysis of a Recommendation System Based on Collaborative Filtering Techniques for Big

Data. *Intelligent and Converged Networks*. 2023. Vol. 4, no. 4. P. 296–304. URL: https://doi.org/10.23919/icn.2023.0024 (date of access: 25.03.2026).

9. Бухало В. О. Дослідження методу взаємного злиття рангів у методах колаборативної фільтрації рекомендаційних систем на основі неявного зворотного зв'язку / В. О. Бухало, С. В. Мінухін // *Радіоелектроніка та молодь у XXI столітті: матеріали 30-го Міжнар. молодіж. форуму*, 22–24 квітня 2026 р. – Харків : ХНУРЕ, 2026 . – Т. 6. – С. 409–412. URL: https://drive.google.com/file/d/1S1pjnCsfpmXD6FO5zqSq80esTiMQ-Pmf/view (date of access: 13.05.2026).

10. Minukhin S. V., Bukhalo V. O. A study of the impact of the weighted reciprocal rank fusion method on the quality of recommender systems. *Наука і техніка сьогодні*. 2026. No. 3(57). P. 1865–1879. DOI: https://doi.org/10.52058/2786-6025-2026-3(57)-1865-1879.

11. Recommender Systems: An Introduction / D. Jannach et al. Cambridge University Press, 2010.

12. Yin N. A Big Data Analysis Method Based on Modified Collaborative Filtering Recommendation Algorithms. *Open Physics*. 2019. Vol. 17, no. 1. P. 966–974. URL: https://doi.org/10.1515/phys-2019-0102 (date of access: 25.03.2026).

13. Product collaborative filtering based recommendation systems for large-scale E-commerce / T. Trinh et al. *International Journal of Information Management Data Insights*. 2025. Vol. 5, no. 1. P. 100322. URL: https://doi.org/10.1016/j.jjimei.2025.100322 (date of access: 25.03.2026).

14. He X., Liu Q., Jung S. The Impact of Recommendation System on User Satisfaction: A Moderated Mediation Approach. *Journal of Theoretical and Applied Electronic Commerce Research*. 2024. Vol. 19, no. 1. P. 448–466. URL: https://doi.org/10.3390/jtaer19010024 (date of access: 05.03.2026).

15. YouTube Music - Wikipedia. URL: https://en.wikipedia.org/wiki/YouTube_Music (date of access: 31.03.2026).

16. Spotify - Wikipedia. URL: https://en.wikipedia.org/wiki/Spotify (date of access: 31.03.2026).

17. 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: 28.12.2025).

18. Temu - Wikipedia. URL: https://en.wikipedia.org/wiki/Temu (date of access: 31.03.2026).

19. Amatriain X., Basilico J. Recommender Systems in Industry: A Netflix Case Study. *Recommender Systems Handbook*. Boston, MA, 2015. P. 385–419. URL: https://doi.org/10.1007/978-1-4899-7637-6_11 (date of access: 25.04.2026).

20. MovieLens | GroupLens. URL: https://grouplens.org/datasets/movielens/ (date of access: 05.03.2026).

21. Amazon Reviews'23. URL: https://amazon-reviews-2023.github.io/index.html (date of access: 05.03.2026).

22. Trabelsi F., Khtira A., El Asri B. Hybrid Recommendation Systems: A State of Art. *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering – Volume 1: ENASE*. 2021. P. 281–288. URL: https://doi.org/10.5220/0010452202810288 (date of access: 28.12.2025).

23. Burke R. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*. 2002. Vol. 12. URL: https://doi.org/10.1023/A:1021240730564 (date of access: 28.12.2025).

24. Bałchanowski M., Boryczka U. A Comparative Study of Rank Aggregation Methods in Recommendation Systems. *Entropy*. 2023. Vol. 25, no. 1. P. 132. URL: https://doi.org/10.3390/e25010132 (date of access: 05.03.2026).

25. Takács G., Pilászy I., Németh B., Tikk D. Scalable Collaborative Filtering Approaches for Large Recommender Systems. *Journal of Machine Learning Research*. 2009. Vol. 10, no. 22. P. 623–656. URL: http://jmlr.org/papers/v10/takacs09a.html (date of access: 25.04.2026).

26. Jester Datasets. URL: https://eigentaste.berkeley.edu/dataset/ (date of access: 12.04.2026).

27. Variational Autoencoders for Collaborative Filtering / D. Liang et al. *the 2018 World Wide Web Conference*, Lyon, France, 23–27 April 2018. New York, New York, USA, 2018. URL: https://doi.org/10.1145/3178876.3186150 (date of access: 25.03.2026).

28. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2002.02126 (date of access: 25.03.2026).

29. Neural Collaborative Filtering. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.1708.05031 (date of access: 25.03.2026).

30. Graham S., Min J.-K., Wu T. Microsoft Recommenders: Tools to Accelerate Developing Recommender Systems. *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*, Copenhagen. 2019. P. 542–543. URL: https://doi.org/10.1145/3298689.3346967 (date of access: 05.03.2026).

31. Apache Spark™ - Unified Engine for large-scale data analytics. *Apache Spark™ - Unified Engine for large-scale data analytics*. URL: https://spark.apache.org/ (date of access: 24.03.2026).

32. Мінухін С., Коптілов Н. Метод збільшення продуктивності Apache Spark на основі сегментування даних і налаштувань конфігураційних параметрів // *Сучасний стан наукових досліджень та технологій в промисловості*. 2024. № 1(27). С. 128-139. URL: https://doi.org/10.30837/ITSSI.2024.27.128.

33. HDFS Architecture Guide. *Apache Hadoop*. URL: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (date of access: 25.03.2026).

34. Мінухін С. В., Рудой І. М. Дослідження архітектур нейронних мереж для обробки мультимодальних даних // *Concepts for the development of society's scientific potential: proceedings of the 8th International scientific and practical conference*, 19–20 October 2025, Prague. Prague, 2025. С. 214–222. URL: https://repository.hneu.edu.ua/handle/123456789/37424 (дата звернення: 14.03.2026).

35. Yilma B. A., Leiva L. A. MOSAIC: Multimodal Multistakeholder-aware Visual Art Recommendation. *arXiv*. 2024. 2407.21758. URL: https://doi.org/10.48550/arXiv.2407.21758 (date of access: 05.03.2026).

36. How scales influence user rating behaviour in recommender systems / F. Cena et al. *Behaviour & Information Technology*. 2017. Vol. 36, no. 10. P. 985–1004. URL: https://doi.org/10.1080/0144929X.2017.1322145 (date of access: 05.03.2026).

37. A Survey on Rank Aggregation / S. Wang et al. *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)* / ed. by K. Larson. 2024. P. 8281–8289. URL: https://doi.org/10.24963/ijcai.2024/915.

38. MMMORRF: Multimodal Multilingual Modularized Reciprocal Rank Fusion / S. Samuel et al. *arXiv*. 2025. 2503.20698. URL: https://doi.org/10.48550/arXiv.2503.20698 (date of access: 05.03.2026).

39. Cormack G. V., Clarke C. L. A., Buettcher S. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *the 32nd international ACM SIGIR conference*, Boston, MA, USA, 19–23 July 2009. New York, New York, USA, 2009. URL: https://doi.org/10.1145/1571941.1572114 (date of access: 25.03.2026).

40. 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: 24.03.2026).

41. De Biasio A., Jannach D., Navarin N. Model-based approaches to profit-aware recommendation. *Expert Systems with Applications*. 2024. Vol. 249. P. 123642. URL: https://doi.org/10.1016/j.eswa.2024.123642 (date of access: 05.03.2026).

42. Nugroho A., Suhartanto H. Hyper-Parameter Tuning based on Random Search for DenseNet Optimization. *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia, 24–25 September 2020. 2020. URL: https://doi.org/10.1109/icitacee50144.2020.9239164 (date of access: 25.04.2026).

43. Dirichlet's principle - Wikipedia. URL: https://en.wikipedia.org/wiki/Dirichlet's_principle (date of access: 13.04.2026).

44. A Critical Study on Data Leakage in Recommender System Offline Evaluation / Y. Ji et al. *ACM Transactions on Information Systems*. 2023. Vol. 41, no. 3. P. 27. URL: https://doi.org/10.1145/3569930 (date of access: 15.03.2026).

45. Personalized Federated Recommendation With Knowledge Guidance. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2511.12959 (date of access: 30.12.2025).

46. Kaggle - Wikipedia. URL: https://en.wikipedia.org/wiki/Kaggle (date of access: 26.04.2026).

47. conda - anaconda | Anaconda.org. URL: https://anaconda.org/anaconda/conda (date of access: 26.04.2026).

48. PyPI · The Python Package Index. URL: https://pypi.org/ (date of access: 26.04.2026).

49. Hug N. Surprise: A Python library for recommender systems. *Journal of Open Source Software*. 2020. Vol. 5, no. 52. P. 2174. URL: https://doi.org/10.21105/joss.02174 (date of access: 13.05.2026).

50. recommenders.evaluation.spark_evaluation – Recommenders 1.1.1 documentation. URL: https://microsoft-recommenders.readthedocs.io/en/latest/_modules/recommenders/evaluation/spark_evaluation.html (date of access: 13.05.2026).

51. Large Graph Convolutional Network Training with GPU-Oriented Data Communication Architecture. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2103.03330 (date of access: 29.03.2026).

52. ДСТУ 3008:2015 «Звіти у сфері науки і техніки. Структура та правила оформлення». Київ: Держстандарт України, 2017. 31 с.

53. ДСТУ 8302:2015 «Бібліографічне посилання. Загальні положення та правила складання». Київ: Держстандарт України, 2017. 20 с.

54. Методичні вказівки до організації виконання та захисту кваліфікаційної роботи на здобуття другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки, освітньо-наукова програма «Системне проєктування» / Упорядники: І.В. Гребеннік, Н.І. Калита, І.М. Рябченко, З.А Імангулова, О.Б. Колесник, М.Ю. Вишняк. Харків: ХНУРЕ, 2021. 56 с.