Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Інтелектуальна система прогнозування поведінки користувачів
на основі методів підкріплювального навчання
(тема)

Виконав:
здобувач другого року навчання,
групи ДСм-24-1

Мацицький А.О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані (Data Science)

(повна назва спеціалізації)
Керівник доц. Дейнеко А.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Лариса ЧАЛА
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Науки про дані (Data Science)
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Мацицькому Артему Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система прогнозування поведінки користувачів на основі методів підкріплювального навчання

затверджена наказом університету від 24 листопада 20 25 р. № 1057Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 грудня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації з проблематики рекомендаційних систем та навчання з підкріпленням (Reinforcement Learning); документація бібліотек машинного навчання (TensorFlow/Keras або PyTorch); набір даних (dataset) MovieLens для тренування та тестування моделі; методичні вказівки до виконання роботи

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні основи прогнозування поведінки в інтелектуальних системах
2) Розробка моделей та методів інтелектуальної системи
3) Програмна реалізація і експериментальні дослідження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	24.11.2025	виконано
2	Аналіз предметної галузі та літературних джерел	25.11.2025	виконано
3	Математична формалізація задачі (MDP, простір	26.11.2025	виконано
	станів та дій)		
4	Обґрунтування вибору методу Double DQN та	27.11.2025	виконано
	функцій винагороди		
5	Розробка архітектури системи та програмна	29.11.2025	виконано
	реалізація		
6	Проведення експериментальних досліджень та	01.12.2025	виконано
	аналіз результатів		
7	Оформлення пояснювальної записки	04.12.2025	виконано
8	Перевірка на академічний плагіат	12.12.2025	виконано
9	Проходження нормоконтролю	14.12.2025	виконано
10	Підготовка презентаційних матеріалів та	15.12.2025	виконано
	доповіді		
11	Попередній захист на кафедрі	16.12.2025	виконано
12	Рецензування роботи	17.12.2025	виконано
13	Захист роботи перед Екзаменаційною комісією	18.12.2025	

Дата видачі завдання 24 листопада 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Дейнеко А.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 65 с., 4 рис., 1 табл., 1 дод., 30 джерел.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА, НЕЙРОННА МЕРЕЖА, ПІДКРІПЛЮВАЛЬНЕ НАВЧАННЯ, ПРОГНОЗУВАННЯ ПОВЕДІНКИ, РЕКОМЕНДАЦІЙНА СИСТЕМА, DOUBLE DQN, MARKOV DECISION PROCESS, TENSORFLOW.

Об’єкт дослідження – процес прогнозування поведінки користувачів у рекомендаційних системах.

Предмет дослідження – методи глибокого навчання з підкріпленням для формування персоналізованих рекомендацій.

Мета роботи – підвищення ефективності прогнозування поведінки користувачів шляхом розробки інтелектуальної системи на основі методу Double Deep Q-Network.

Методи дослідження: теоретичний аналіз, математичне моделювання (MDP), методи машинного навчання (Deep RL), експериментальний метод.

Результати та новизна. Розроблено архітектуру інтелектуальної системи, що враховує динаміку інтересів користувача. Запропоновано модифіковану функцію винагороди.

Практичне значення. Створено програмний продукт мовою Python (TensorFlow) для генерації рекомендацій фільмів. Експериментально підтверджено здатність системи адаптуватися до вподобань користувачів.

Сфера застосування – рекомендаційні сервіси та стрімінгові платформи.

Висновки. Використання глибокого навчання з підкріпленням дозволяє ефективно вирішувати задачі рекомендації в динамічних середовищах.

ABSTRACT

Master’s thesis contains: 65 pp., 4 fig., 1 tabl., 1 ann., 30 references.

BEHAVIOR PREDICTION, DOUBLE DQN, INTELLIGENT SYSTEM, MDP, RECOMMENDATION SYSTEM, REINFORCEMENT LEARNING, TENSORFLOW.

The object of study is the process of user behavior prediction in recommendation systems.

The subject of study is deep reinforcement learning methods for personalized recommendations.

The goal of the work is to improve user behavior prediction efficiency by developing an intelligent system based on the Double Deep Q-Network method.

Research methods: theoretical analysis, mathematical modeling (MDP), machine learning methods (Deep RL), experimental method.

Results and novelty. An intelligent system architecture considering user interest dynamics has been developed. A modified reward function is proposed, and the Double DQN algorithm is applied to stabilize training and avoid Q-value overestimation.

Practical value. A Python-based software product (TensorFlow) for movie recommendations has been created. Experiments confirmed the system's ability to adapt to user preferences.

Relationship with other works. The work was performed independently.

Conclusions. Deep reinforcement learning effectively solves recommendation tasks in dynamic environments.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕОМ – електронна обчислювальна машина;

ПЗ – програмне забезпечення;

API – Application Programming Interface – інтерфейс прикладного програмування;

Deep RL – Deep Reinforcement Learning – глибоке навчання з підкріпленням;

DQN – Deep Q-Network – Глибока Q-мережа;

Double DQN – Double Deep Q-Network – подвійна глибока Q-мережа;

MDP – Markov Decision Process – марковський процес прийняття рішень;

MSE – Mean Squared Error – середньоквадратична помилка;

ReLU – Rectified Linear Unit – випрямлений лінійний вузол;

RL – Reinforcement Learning – навчання з підкріпленням;

SGD – Stochastic Gradient Descent – стохастичний градієнтний спуск;

A – простір можливих дій агента;

a_t – дія, яку виконує агент у момент часу t;

γ – (гама) коефіцієнт дисконтування ($0 \leq \gamma \leq 1$);

є – (епсилон) параметр жадібної стратегії, що визначає ймовірність вибору випадкової дії;

θ – (тета) параметри (ваги) нейронної мережі;

P – функція ймовірності переходу між станами;

Q(s,a) – функція цінності дії (Q-функція);

R – функція винагороди;

r_t – миттєва винагорода, отримана агентом у момент часу t;

S – простір можливих станів середовища;

s_t – стан середовища у момент часу t.

ВСТУП

Стрімкий розвиток інформаційних технологій та глобальної мережі Інтернет призвів до експоненційного зростання обсягів цифрового контенту, що, своєю чергою, актуалізувало проблему інформаційного перевантаження користувачів. У сучасних умовах функціонування стрімінгових платформ та сервісів електронної комерції ключовим фактором конкурентоспроможності стає здатність системи надавати персоналізовані пропозиції, які максимально відповідають поточним інтересам та потребам конкретної людини. Традиційні підходи до побудови рекомендаційних систем, такі як колаборативна фільтрація або контент-орієнтовані методи, тривалий час залишалися стандартом індустрії, однак вони мають суттєві обмеження, пов'язані здебільшого зі статичним характером моделювання вподобань. Такі системи розглядають процес рекомендації як одномоментну задачу прогнозування рейтингу, ігноруючи динаміку зміни інтересів користувача в часі та послідовний характер його взаємодії з платформою.

Сучасний етап розвитку методів штучного інтелекту характеризується переходом до більш складних, адаптивних моделей, серед яких особливе місце посідає навчання з підкріпленням. Ця парадигма дозволяє формалізувати задачу рекомендації як Марковський процес прийняття рішень, де інтелектуальний агент навчається обирати оптимальну послідовність дій шляхом постійної взаємодії із середовищем. Актуальність теми кваліфікаційної роботи зумовлена необхідністю подолання недоліків класичних підходів та створення інтелектуальних систем нового покоління, здатних не просто передбачати реакцію на окремий об'єкт, а й стратегічно максимізувати довгострокову задоволеність користувача. Застосування методів глибокого навчання з підкріпленням відкриває нові перспективи для обробки високорозмірних просторів станів, характерних для реальних рекомендаційних задач, проте водночас

породжує нові виклики, пов'язані зі стабільністю навчання та обчислювальною складністю алгоритмів.

Метою роботи є підвищення ефективності прогнозування поведінки користувачів шляхом розробки та програмної реалізації інтелектуальної системи на основі модифікованих методів глибокого підкріплювального навчання. Для досягнення цієї мети у роботі вирішується наукова задача адаптації алгоритму Double Deep Q-Network до специфіки рекомендаційного середовища, що дозволяє усунути проблему систематичної переоцінки цінності дій, властиву базовим архітектурам. Об'єктом дослідження виступає процес інтелектуального аналізу даних та прогнозування поведінки користувачів у рекомендаційних системах, а предметом дослідження є моделі та методи глибокого навчання з підкріпленням, спрямовані на оптимізацію стратегій персоналізації контенту.

Наукова новизна отриманих результатів полягає у подальшому розвитку методології побудови рекомендаційних агентів, які, на відміну від «жадібних» стратегій, орієнтованих на миттєву вигоду, враховують довгостроковий вплив рекомендацій на лояльність користувача. У роботі запропоновано вдосконалену структуру винагороди агента та архітектуру нейронної мережі, що забезпечує більш точну апроксимацію функції цінності в умовах розрідженості даних. Практичне значення роботи підтверджується розробкою дієздатного програмного прототипу системи та проведенням серії експериментів на реальному наборі даних MovieLens, результати яких демонструють перевагу запропонованого підходу над статичними методами за критерієм сумарної отриманої винагороди. Отримані результати можуть бути використані при проектуванні промислових рекомендаційних систем для відеосервісів, інтернет-магазинів та новинних агрегаторів, сприяючи покращенню користувацького досвіду та підвищенню ефективності бізнес-процесів.

1 ТЕОРЕТИЧНІ ОСНОВИ ПРОГНОЗУВАННЯ ПОВЕДІНКИ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

Сучасний етап розвитку інформаційних технологій характеризується експоненційним зростанням обсягів даних, що генеруються користувачами в глобальній мережі. В умовах цифрової економіки дані про поведінку, вподобання та історію взаємодії клієнтів стають найціннішим активом для бізнес-структур, особливо у сферах електронної комерції (e-commerce), стрімінгових сервісів та соціальних мереж. Проте просте накопичення інформації не гарантує конкурентних переваг. Ключовим фактором успіху стає здатність інтелектуальних систем не лише аналізувати минулі події, а й з високою точністю прогнозувати майбутні дії користувача.

Традиційні підходи до аналізу даних, які базуються на статичних моделях та жорстких правилах, поступово втрачають свою ефективність в умовах динамічного середовища, де інтереси користувачів змінюються миттєво. Це зумовлює необхідність розробки та впровадження нових методів машинного навчання, здатних адаптуватися до змін у реальному часі. У цьому розділі проведено аналіз теоретичних основ прогнозування поведінки користувачів, розглянуто сучасний стан проблеми, існуючі виклики та обмеження класичних підходів, а також обґрунтовано доцільність використання методів підкріплювального навчання як інструменту для побудови адаптивних інтелектуальних систем.

1.1 Сучасний стан проблеми аналізу поведінки користувачів

В умовах стрімкої цифровізації суспільства та глобального переходу бізнес-процесів у онлайн-середовище, проблема ефективної взаємодії інформаційних систем з кінцевим споживачем набуває критичного значення. Сучасний інтернет-простір характеризується експоненційним зростанням обсягів даних. За оцінками аналітиків, кількість цифрової

інформації подвоюється кожні два роки, що створює безпрецедентне навантаження на когнітивні здібності користувачів. Феномен інформаційного перенасичення, або так званий «парадокс вибору», призводить до того, що користувачі, стикаючись з тисячами альтернатив (товарів, медіа-контенту, послуг), втрачають здатність приймати раціональні рішення, що негативно впливає як на їхній особистий досвід (User Experience), так і на економічні показники сервіс-провайдерів. У роботі [1] зазначається, що надмірний вибір часто призводить до повної відмови від покупки або до зниження задоволеності зробленим вибором.

У цьому контексті системи прогнозування поведінки та рекомендаційні сервіси трансформувалися з допоміжного функціоналу в ядро бізнес-моделей провідних технологічних гігантів. Компанії, такі як Amazon, Netflix, YouTube та Spotify, будують свої екосистеми навколо здатності алгоритмів передбачати бажання користувача ще до того, як він їх сформулює. Як зазначають автори фундаментальної праці з рекомендаційних систем [2], ефективна персоналізація дозволяє не лише спростити навігацію в інформаційному просторі, але й виступає ключовим інструментом управління увагою користувача, що є найдефіцитнішим ресурсом у цифровій економіці.

Прогнозування поведінки користувача – це складна науково-технічна задача, яка полягає у побудові імовірнісної моделі майбутніх дій об'єкта на основі аналізу його історичних даних, контексту та характеристик доступних альтернатив. Актуальність вдосконалення методів прогнозування зумовлена необхідністю вирішення двох взаємопов'язаних бізнес-задач: максимізації прибутку через підвищення конверсії (Conversion Rate Optimization) та збільшення життєвого циклу клієнта (Customer Lifetime Value – CLV). Згідно з дослідженням, проведеним для сервісу Netflix, близько 80% контенту, що переглядається на платформі, обирається саме завдяки роботі рекомендаційних алгоритмів, що дозволяє компанії

економити понад 1 мільярд доларів щорічно за рахунок утримання клієнтів та зменшення відтоку (Churn Rate) [4].

Однак, незважаючи на значні успіхи у розробці алгоритмів машинного навчання, існуючі підходи до прогнозування поведінки стикаються з низкою фундаментальних викликів, які обмежують їх ефективність у реальних динамічних середовищах. Аналіз наукової літератури [5], [7] дозволяє виділити основні групи проблем: розрідженість даних, проблема «холодного старту», масштабованість, а також адаптація до зміни інтересів (Concept Drift). Розглянемо кожну з цих проблем детальніше, оскільки саме їх невирішеність обумовлює необхідність переходу до нових методів, зокрема навчання з підкріпленням.

Проблема розрідженості даних (Data Sparsity) є однією з найбільш поширених у системах електронної комерції. Уявімо матрицю взаємодій «користувач-товар» розмірністю M×N, де M – кількість користувачів, а N – кількість товарів. У великих системах обидва параметри можуть досягати мільйонів. Оскільки середньостатистичний користувач взаємодіє лише з обмеженою кількістю об'єктів (зазвичай менше 1%), щільність такої матриці часто становить частки відсотка. Класичні методи, такі як колаборативна фільтрація (User-based або Item-based Collaborative Filtering), базуються на пошуку подібних векторів у цій матриці. При високому рівні розрідженості знайти статистично значущих «сусідів» стає неможливо, що призводить до низької точності прогнозів або повної неможливості їх формування для певної частини аудиторії [8].

Наступним критичним викликом є проблема «холодного старту» (Cold Start Problem), яка проявляється у трьох аспектах: новий користувач, новий товар та нова система. Найбільш гострою є проблема нового користувача: коли особа вперше реєструється в системі, історія її дій відсутня. Традиційні моделі, які спираються на ретроспективний аналіз кліків або покупок, у цьому випадку безсилі. Система змушена пропонувати усереднені популярні товари (Most Popular), що нівелює ефект

персоналізації і може призвести до втрати інтересу користувача вже на першому етапі знайомства з сервісом. Проблема нового товару (Item Cold Start) є не менш важливою: нові позиції каталогу, які ще не отримали достатньої кількості взаємодій, ігноруються алгоритмами колаборативної фільтрації, створюючи ефект «замкненого кола», коли товар не рекомендується, бо його не купують, і не купують, бо не рекомендують [5].

Особливу увагу в сучасних дослідженнях приділяють проблемі динаміки інтересів користувачів, відомій в машинному навчанні як Concept Drift. Поведінка людини не є стаціонарним процесом. Вподобання користувачів змінюються під впливом різноманітних факторів: сезонності, зміни соціального статусу, модних трендів або навіть настрою в конкретний момент часу. Як зазначають дослідники [6], зміни можуть бути раптовими (наприклад, початок ремонту призводить до різкого інтересу до будівельних матеріалів), поступовими (зміна музичних смаків з віком) або циклічними (купівля подарунків перед святами). Більшість традиційних методів, зокрема матрична факторизація (Matrix Factorization), моделюють вподобання користувача як статичний вектор у латентному просторі ознак. Такий підхід усереднює інтереси за весь період історії, що призводить до ситуацій, коли система рекомендує товари, які були актуальні рік тому, але вже втратили релевантність. Нездатність моделі швидко адаптуватися до зміни контексту (Contextual Awareness) є суттєвим недоліком існуючих рішень.

Ще одним аспектом, який ускладнює задачу прогнозування, є природа зворотного зв'язку. У більшості онлайн-систем отримання явного зворотного зв'язку (explicit feedback), такого як рейтинги (зірочки) або лайки, є рідкістю. Користувачі неохоче витрачають час на оцінювання. Тому системи змушені покладатися на неявний зворотний зв'язок (implicit feedback) – історію переглядів, час перебування на сторінці, додавання в кошик, рухи курсору [10]. Дані неявного зворотного зв'язку є зашумленими та неоднозначними: факт кліку не завжди означає задоволеність

контентом (клікбейт), а відсутність кліку не завжди означає відсутність інтересу (товар міг бути просто непоміченим). Обробка таких даних вимагає використання складних імовірнісних моделей та методів глибокого навчання (Deep Learning), здатних виділяти нелінійні залежності у великих масивах неструктурованої інформації [9].

Проблема масштабованості (Scalability) також залишається актуальною. Сучасні високонавантажені системи повинні обробляти запити мільйонів користувачів у режимі реального часу (Real-time bidding, online recommendations). Затримка у формуванні прогнозу навіть на частки секунди може призвести до втрати уваги користувача. Багато складних алгоритмів, які демонструють високу точність в лабораторних умовах на статичних датасетах, виявляються непридатними для впровадження в продакшн через високу обчислювальну складність, яка зростає нелінійно зі збільшенням кількості користувачів та товарів [7].

Окремо варто виділити проблему оптимізації довгострокової винагороди. Більшість класичних методів навчання з учителем (Supervised Learning) спрямовані на мінімізацію помилки передбачення наступної дії (наприклад, клікне користувач на банер чи ні). Це так звана «жадібна» (greedy) стратегія, яка фокусується на миттєвій вигоді (Short-term reward). Однак, така стратегія може бути шкідливою в довгостроковій перспективі. Наприклад, рекомендація провокаційного контенту (клікбейт) може забезпечити високий CTR (Click-Through Rate) сьогодні, але призведе до розчарування користувача та його відтоку з платформи через тиждень. Задача інтелектуальної системи – знайти баланс між експлуатацією (Exploitation) поточних інтересів користувача та дослідженням (Exploration) нових потенційних зон інтересу. Саме відсутність механізмів дослідження є критичним недоліком традиційних підходів [3].

Підсумовуючи аналіз сучасного стану проблеми, можна констатувати, що, незважаючи на наявність широкого спектру методів прогнозування

поведінки (від простих статистичних правил до складних нейронних мереж), жоден з них не є універсальним. Індустрія потребує переходу до нового класу інтелектуальних систем, які були б здатні:

– працювати з послідовними даними (Sequential Data), враховуючи порядок дій, а не лише їх множину [10];

– адаптуватися до зміни інтересів у режимі реального часу (Online Learning);

– оптимізувати не лише миттєвий клік, а й довгострокову залученість (Long-term Engagement);

– ефективно вирішувати проблему «холодного старту» через активне дослідження середовища.

Ці вимоги вказують на необхідність застосування методів, що виходять за рамки класичного навчання з учителем. Навчання з підкріпленням (Reinforcement Learning – RL), яке розглядає задачу прогнозування як послідовний процес прийняття рішень (Sequential Decision Making Process), є найбільш перспективним напрямком для подолання описаних вище обмежень. У наступних підрозділах буде детально розглянуто класифікацію існуючих методів та теоретичні основи застосування RL у даній предметній галузі.

1.2 Класифікація методів прогнозування та рекомендація

Різноманіття задач прогнозування поведінки користувачів зумовило появу широкого спектру методів та алгоритмів, які еволюціонували від простих статистичних евристик до складних нейромережевих архітектур. У науковій літературі [1], [2] загальноприйнятим є поділ методів на дві великі групи: класичні підходи (Traditional Approaches), до яких відносять контент-орієнтовану та колаборативну фільтрацію, та методи на основі глибокого навчання (Deep Learning-based Approaches), які дозволяють моделювати складні нелінійні залежності та обробляти послідовні дані.

1.2.1 Класичні підходи прогнозування

Історично першими та найбільш інтерпретованими методами є контент-орієнтована фільтрація (Content-Based Filtering). Основна гіпотеза цього підходу полягає в тому, що користувач у майбутньому обиратиме об'єкти, схожі на ті, з якими він взаємодіяв у минулому. Для побудови прогнозу система створює профіль користувача та профілі об'єктів, що складаються з набору атрибутів (наприклад, жанр фільму, категорія товару, ціновий діапазон, опис тексту). Схожість між профілем користувача та товаром розраховується за допомогою метрик відстані, таких як косинусна подібність або коефіцієнт Жаккара. Головною перевагою цього методу є відсутність проблеми «холодного старту» для нових товарів: як тільки товар з'являється в каталозі з описом, він може бути рекомендований [8]. Однак, суттєвим недоліком є проблема надмірної спеціалізації (over-specialization): система не здатна рекомендувати щось принципово нове, обмежуючи користувача в межах його існуючих інтересів («information bubble»).

Найбільш поширеним класом методів у комерційних системах до появи глибокого навчання була колаборативна фільтрація (Collaborative Filtering – CF). Цей підхід ігнорує внутрішні характеристики об'єктів, спираючись виключно на історію взаємодій (матрицю рейтингів або кліків). CF поділяється на два підвиди:

– методи на основі пам'яті (Memory-based): User-based (шукає схожих користувачів) та Item-based (шукає схожі товари). Наприклад, якщо користувач А схожий на користувача Б, і Б купив товар Х, то система спрогнозує, що А теж зацікавиться товаром Х;

– методи на основі моделей (Model-based): Найяскравішим представником є матрична факторизація (Matrix Factorization), популяризована під час змагання Netflix Prize [11]. Метод передбачає розклад розрідженої матриці взаємодій на дві матриці меншої

розмірності (латентні фактори користувачів та товарів). Скалярний добуток цих векторів дає прогнозовану оцінку інтересу.

Попри високу точність на статичних даних, класична колаборативна фільтрація має критичне обмеження: вона розглядає історію користувача як «мішок товарів» (basket of items), ігноруючи хронологічний порядок дій. У задачах прогнозування поведінки (sequence prediction), де важлива саме послідовність подій (наприклад, спочатку перегляд телефону, потім перегляд чохла до нього), класичні методи втрачають контекст та не здатні вловити динаміку зміни інтересів [10].

1.2.2 Методи на основі глибокого навчання у задачах Sequence Modeling

Необхідність врахування часової залежності та послідовності дій призвела до активного впровадження методів глибокого навчання (Deep Learning), зокрема архітектур, розроблених для обробки природної мови (NLP). У цьому контексті сесія користувача розглядається як речення, а товари – як слова. Задачею системи стає передбачення наступного «слова» (дії) у послідовності.

Піонерами у цій галузі стали рекурентні нейронні мережі (Recurrent Neural Networks – RNN) та їх модифікації: LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit). На відміну від класичних нейронних мереж, RNN мають внутрішній стан (пам'ять), який оновлюється на кожному кроці послідовності. Це дозволяє мережі «пам'ятати» попередні дії користувача і використовувати цей контекст для прогнозування наступного кроку. У роботі [12] було запропоновано архітектуру GRU4Rec, яка стала стандартом для сесійних рекомендацій. Експерименти показали, що GRU4Rec значно перевершує методи найближчих сусідів та матричної факторизації на коротких сесіях, ефективно вирішуючи проблему моделювання короткострокових інтересів. LSTM, завдяки механізму «воріт» (gates),

здатні утримувати інформацію про важливі події, що відбулися багато кроків тому, вирішуючи проблему зникаючого градієнта, притаманну звичайним RNN [7].

Новим етапом розвитку систем прогнозування стала поява архітектури Transformers та механізму самоуваги (Self-Attention). Моделі, такі як SASRec (Self-Attentive Sequential Recommendation) [13] та BERT4Rec, відмовилися від рекурентності на користь паралельної обробки всієї послідовності дій. Механізм Attention дозволяє моделі динамічно визначати вагу (важливість) кожної попередньої дії для формування поточного прогнозу. Наприклад, при виборі нового ноутбука система зверне більше уваги на попередні перегляди техніки, ігноруючи випадкові кліки на одяг, зроблені в тій самій сесії. Трансформери демонструють кращу здатність до масштабування та навчання на великих обсягах даних порівняно з RNN, стаючи сучасним стандартом (State-of-the-Art) у задачах Sequence Modeling.

1.2.3 Обмеження методів Sequence Modeling

Незважаючи на значний прогрес, описані методи глибокого навчання (RNN, CNN, Transformers) належать до парадигми навчання з учителем (Supervised Learning). Їхня цільова функція зазвичай зводиться до мінімізації помилки передбачення наступної дії (Next Item Prediction). Це означає, що моделі є «жадібними» (greedy) та короткозорими: вони намагаються вгадати, що користувач натисне прямо зараз, не враховуючи, як ця рекомендація вплине на його поведінку в майбутньому.

Такий підхід має суттєві недоліки в контексті довгострокової взаємодії:

– відсутність планування: Модель не будує стратегію взаємодії, яка б вела користувача до цільової дії (наприклад, покупки) через ланцюжок рекомендацій;

– петля зворотного зв'язку (Feedback Loop): Система навчається на історичних даних, які вже були сформовані під впливом попередніх рекомендаційних алгоритмів, що призводить до зміщення (bias) та звуження кругозору користувача;

– пасивність: Supervised-моделі не вміють активно досліджувати інтереси користувача, пропонуючи лише те, що має найвищу ймовірність кліку згідно з минулим досвідом.

Ці обмеження вказують на необхідність застосування підходів, які фокусуються не на миттєвій точності передбачення, а на максимізації сумарної користі (винагороди) за тривалий період. Саме таку парадигму пропонує навчання з підкріпленням (Reinforcement Learning), концепція якого буде розглянута в наступному підрозділі.

1.3 Концепція навчання з підкріпленням (Reinforcement Learning – RL)

Навчання з підкріпленням (Reinforcement Learning – RL) являє собою третю фундаментальну парадигму машинного навчання, поряд з навчанням з учителем (Supervised Learning) та навчанням без учителя (Unsupervised Learning). Якщо навчання з учителем базується на наявності розмічених даних, які виступають у ролі інструкцій, а навчання без учителя зосереджене на пошуку прихованих структур у нерозмічених даних, то навчання з підкріпленням фокусується на навчанні через взаємодію. Це обчислювальний підхід до розуміння та автоматизації цілеспрямованого навчання та прийняття рішень . RL розглядає проблему агента, який повинен навчитися поводитися в динамічному середовищі шляхом проб і помилок, щоб максимізувати певний числовий сигнал винагороди [3].

В основі RL лежить ідея біхевіористської психології, згідно з якою живі організми навчаються повторювати дії, що призводять до позитивних наслідків (позитивне підкріплення), і уникати дій, що викликають негативні наслідки (покарання). У контексті штучного інтелекту цей біологічний

принцип формалізується через математичний апарат оптимального керування та Марковських процесів прийняття рішень (Markov Decision Processes – MDP) [15].

1.3.1 Базова теорія RL: агент, середовище та цикл взаємодії

Система навчання з підкріпленням складається з двох основних компонентів: агента (Agent) та середовища (Environment).

Агент – це програмна сутність, яка приймає рішення. Це компонент, що навчається, обирає дії та намагається досягти мети. У контексті рекомендаційних систем агентом виступає сам алгоритм рекомендацій.

Середовище – це все, що оточує агента і з чим він взаємодіє. Середовище реагує на дії агента, змінюючи свій стан та повертаючи числовий сигнал винагороди. Для рекомендаційної системи середовищем є користувачі та база даних товарів.

Взаємодія між агентом та середовищем відбувається дискретно у часі $t=0,1,2,3,\ldots$. На кожному часовому кроці t відбувається наступний цикл подій:

– агент отримує від середовища спостереження про поточний стан $S_t \in S$;

– на основі стану S_t агент обирає дію $A_t \in A(S_t)$;

– у відповідь на дію агента середовище переходить у новий стан S_{t+1} та надсилає агенту скалярний сигнал винагороди $R_{t+1} \in R$.

Цей процес триває до тих пір, поки не буде досягнуто термінального стану (кінець епізоду) або нескінченно (у задачах безперервного керування). Метою агента є максимізація сумарної винагороди, яку він отримує протягом тривалого часу.

Розглянемо детальніше ключові елементи цієї взаємодії, які утворюють кортеж MDP (S,A,P,R,γ).

Простір станів (State Space, S) Стан St – це стисле відображення поточної конфігурації середовища, яке містить усю необхідну інформацію для прийняття рішення агентом. У теорії RL фундаментальною вимогою до стану є виконання властивості Маркова. Стан вважається марковським, якщо він повністю підсумовує історію взаємодії, тобто ймовірність переходу в наступний стан залежить лише від поточного стану та дії, а не від попередньої історії подій [15]:

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \ldots, S_0, A_0). \quad (1.1)$$

У реальних задачах, таких як прогнозування поведінки користувача, повний стан середовища часто є прихованим (Partially Observable MDP – POMDP). Ми не знаємо справжніх думок користувача, тому як «стан» використовуємо векторне представлення історії його дій (embedding), яке апроксимує марковський стан.

Простір дій (Action Space, A) є множиною всіх допустимих рішень, які може прийняти агент. Простір дій може бути:

– дискретним. Агент обирає одну дію зі скінченної множини (наприклад, рекомендувати один товар із каталогу);

– неперервним. Дія є вектором дійсних чисел (наприклад, керування кутом повороту керма автомобіля). У рекомендаційних системах зазвичай використовується дискретний простір дій, де кожна дія відповідає індексу товару (item ID). Однак, при великій кількості товарів (N>106) робота з дискретним простором стає обчислювально складною, що вимагає застосування методів вкладення дій у неперервний простір [16].

Функція переходу (Transition Probability, P) – це модель динаміки середовища, яка визначає ймовірність переходу в стан s′ за умови виконання дії a у стані s:

$$P(s'|s, a) = P[S_{t+1} = s'|S_t = s, A_t = a]. \quad (1.2)$$

У багатьох прикладних задачах (Model-Free RL) функція переходу є невідомою агенту. Агент повинен навчитися діяти, не знаючи заздалегідь, як саме зміниться середовище, а лише спостерігаючи фактичні переходи під час навчання (sampling).

Винагорода Rt – це єдиний сигнал зворотного зв'язку, який вказує агенту на успішність його дій. Це скалярне число, яке середовище повертає агенту на кожному кроці. Критично важливим поняттям у RL є гіпотеза винагороди (Reward Hypothesis), сформульована Саттоном [3]: «Всі цілі можуть бути описані як максимізація очікуваного значення суми отриманих скалярних сигналів (винагород)». Винагорода може бути:

– миттєвою: наприклад, +1 за клік по товару;

– відкладеною (Delayed Reward): наприклад, позитивна оцінка лише після завершення покупки, яка відбулася через 10 кроків після першої рекомендації.

Коефіцієнт дисконтування (Discount Factor, γ) говорить нам про те, що агент прагне максимізувати не просто наступну винагороду, а очікуваний повернений дохід (Return, Gt), який є сумою дисконтованих майбутніх винагород:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \quad (1.3)$$

де $\gamma \in [0,1]$ – коефіцієнт дисконтування, параметр γ визначає далекоглядність агента.

1.3.2 Політика та функції цінності

Центральним елементом інтелекту агента є політика (Policy, π). Політика – це стратегія поведінки агента, відображення (mapping) станів у дії. Вона може бути:

– детермінованою: $a = \pi(s)$ – для кожного стану жорстко визначена одна дія;

– стохастичною: $\pi(a|s) = P[A_t = a|S_t = s]$ – політика повертає розподіл ймовірностей дій. Стохастичні політики є важливими для забезпечення дослідження середовища (Exploration).

Для оцінки якості політики використовуються функції цінності (Value Functions), які прогнозують, скільки винагороди агент отримає в майбутньому [19].

Функція цінності стану (State-Value Function, Vπ(s)): очікуваний повернений дохід, якщо агент знаходиться в стані s і далі діє згідно з політикою π.

$$V_\pi(s) = E_\pi[G_t|S_t = s]. \quad (1.4)$$

Функція цінності дії (Action-Value Function, Qπ(s,a)): очікуваний повернений дохід, якщо агент знаходиться в стані s, виконує дію a, а далі слідує політиці π.

$$Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]. \quad (1.5)$$

Саме функція Q(s,a) (Q-функція) є ключовою для багатьох алгоритмів, таких як Q-Learning та DQN, оскільки знання оптимальної Q-функції $Q^*(s,a)$ дозволяє автоматично отримати оптимальну політику, просто обираючи в кожному стані дію з максимальним значенням Q:

$$\pi^*(s) = arg\ max_a\ Q^*(s,a). \quad (1.6)$$

Фундаментальним співвідношенням, яке дозволяє навчати функції цінності ітеративно, є рівняння Беллмана (Bellman Equation). Воно

рекурсивно пов'язує цінність поточного стану з цінністю наступного стану [15]:

$$Q_\pi(s,a) = E_{s\prime}[R_{t+1} + \gamma Q_\pi(s', A_{t+1})|S_t = s, A_t = a]. \quad (1.7)$$

Це рівняння лежить в основі більшості методів навчання з підкріпленням, дозволяючи агенту оновлювати свої оцінки на основі нового досвіду.

1.3.3 Компроміс між розвідкою та експлуатацією (Exploration-Exploitation Trade-off)

Унікальною проблемою RL, якої немає в навчанні з учителем, є дилема розвідки та експлуатації. Щоб отримати максимальну винагороду, агент повинен обирати дії, які, за його поточними знаннями, є найкращими (експлуатація). Проте, щоб виявити такі дії, він повинен спробувати дії, які він ще не вибирав, ризикуючи отримати меншу винагороду або покарання (розвідка) [16]. У рекомендаційних системах це проявляється так: чи повинна система рекомендувати користувачеві лише ті товари, які він точно купить (наприклад, хліб та молоко), максимізуючи миттєву конверсію, чи варто порекомендувати новий гаджет, який користувач може не купити, але це дозволить системі дізнатися про його нові інтереси? Виключна експлуатація призводить до локальних оптимумів і стагнації («information bubble»), тоді як надмірна розвідка знижує поточну ефективність сервісу. Сучасні RL-алгоритми використовують різні стратегії балансування, такі як є-greedy, додавання шуму до параметрів або метод Upper Confidence Bound (UCB).

1.3.4 Відмінність RL від навчання з учителем (Supervised Learning) у контексті прогнозування

Попри те, що методи глибокого навчання (Deep Learning) широко застосовуються як в RL, так і в навчанні з учителем (Supervised Learning – SL), філософія навчання та постановка задачі в цих парадигмах мають суттєві відмінності. Розуміння цієї різниці є ключовим для обґрунтування вибору RL для розв'язання задачі прогнозування складної поведінки користувачів. Перша суттєва відмінність полягає у характері зворотного зв’язку. Навчання з учителем базується на інструктивному підході, де моделі надається правильна відповідь (мітка) для кожного вхідного прикладу, а помилка розраховується як різниця між прогнозом та істиною [17]. Натомість навчання з підкріпленням використовує оцінювальний зворотний зв’язок, при якому середовище не надає інформації про найкращу дію, а лише повідомляє через скалярний сигнал винагороди, наскільки ефективним було прийняте рішення. Оскільки в задачах рекомендацій ідеальна рекомендація часто є невідомою, підхід RL є більш адаптивним до невизначеності предметної галузі.

Другим важливим аспектом є часовий горизонт оптимізації. Метою SL зазвичай є мінімізація помилки на кожному окремому кроці, що в контексті рекомендацій зводиться до максимізації ймовірності миттєвого кліку (CTR). Такий підхід часто призводить до формування жадібної стратегії, яка ігнорує довгострокові наслідки. Агент RL, навпаки, явно оптимізує кумулятивну винагороду. Маючи коефіцієнт дисконтування $\gamma>0$, система здатна обирати дії, які не приносять миттєвої вигоди, але сприяють підвищенню лояльності користувача в майбутньому, уникаючи стратегій, що призводять до відтоку аудиторії [18].

Третя відмінність стосується фундаментальних припущень щодо природи даних. Методи навчання з учителем зазвичай виходять з того, що дані є незалежними та однаково розподіленими (i.i.d.), тобто попередні

прогнози не впливають на майбутні вхідні дані. У задачах взаємодії з користувачем це припущення порушується, оскільки рекомендації системи безпосередньо впливають на формування майбутніх уподобань та поведінки користувача. RL враховує цю часову залежність та кореляцію даних, використовуючи спеціалізовані механізми стабілізації навчання, такі як буфер відтворення досвіду [17].

Нарешті, варто відзначити проблему призначення кредиту довіри, яка є специфічною для навчання з підкріпленням. У SL зв’язок між помилкою та рішенням є прямим і миттєвим, тоді як в RL виникає часова затримка отримання винагороди. Наприклад, факт покупки може відбутися через значний проміжок часу після серії переглядів та рекомендацій. RL, використовуючи механізми дисконтування та оновлення Q-значень, зокрема через методи часової різниці (TD-навчання), дозволяє визначити вклад кожної попередньої дії в досягнення кінцевого успіху [3].

1.4 Аналіз застосування RL у рекомендаційних системах

Застосування методів навчання з підкріпленням у сфері рекомендаційних систем стало логічним еволюційним кроком після досягнення певного плато ефективності класичними методами глибокого навчання. Традиційні підходи, розглянуті у попередніх підрозділах, здебільшого фокусувалися на статичному прогнозуванні рейтингу або ймовірності кліку, розглядаючи процес рекомендації як одномоментну задачу класифікації або регресії. Проте реальна взаємодія користувача з інформаційною системою є інтерактивним та послідовним процесом, що робить парадигму RL, яка оптимізує довгострокову винагороду через послідовність дій, найбільш релевантною для сучасних вимог індустрії. Аналіз наукових публікацій останніх років свідчить про стрімке зростання інтересу до цього напрямку, що дозволяє виділити окремий клас систем –

рекомендаційні системи на основі підкріплювального навчання (Reinforcement Learning-based Recommender Systems – RLRS).

1.4.1 Огляд існуючих наукових досліджень (State-of-the-Art)

Розвиток RLRS можна умовно поділити на етап адаптації бандитських алгоритмів та етап глибокого підкріплювального навчання. Історично першим кроком до впровадження принципів RL стало використання алгоритмів багаторуких бандитів (Multi-Armed Bandits – MAB), зокрема контекстних бандитів. У фундаментальній роботі Лі та колег [20] було запропоновано алгоритм LinUCB для персоналізації новин на порталі Yahoo!. Цей підхід дозволив вирішити проблему дослідження нових матеріалів, балансуючи між демонстрацією популярних новин (експлуатація) та нових статей (розвідка). Хоча контекстні бандити формально є спрощеною версією RL (оскільки вони не враховують зміну стану середовища після дії), це дослідження заклало основу для розуміння важливості онлайн-навчання.

Справжній прорив відбувся з інтеграцією глибоких нейронних мереж у схему навчання з підкріпленням (Deep Reinforcement Learning – DRL). Однією з найбільш цитованих робіт у цій галузі є дослідження Чжена та співавторів [21], які представили архітектуру DRN (Deep Reinforcement Learning Network) для рекомендацій новин. Автори застосували підхід Deep Q-Network (DQN), де стан користувача моделювався через рекурентну обробку історії його дій, а функція винагороди враховувала не лише факт кліку, але й час повернення користувача на платформу. Ключовою інновацією стала розробка механізму Dueling Bandit Gradient Descent для ефективного дослідження простору дій. Результати експериментів на реальному трафіку показали значне збільшення показників залученості порівняно з традиційними методами.

Подальші дослідження зосередилися на проблемі великої розмірності простору дій. У класичних задачах RL, таких як ігри Atari, кількість можливих дій є невеликою. У рекомендаційних системах кількість товарів може сягати мільйонів, що унеможливлює використання стандартного DQN, який вимагає оцінки Q-значення для кожної можливої дії. Для вирішення цієї проблеми дослідники з Google запропонували використання архітектури Actor-Critic, де «Актор» генерує векторне представлення ідеального товару, а потім відбувається пошук найближчих сусідів у просторі вкладень (embeddings) товарів. У роботі [22] було презентовано підхід Wolpertinger, який поєднує DDPG (Deep Deterministic Policy Gradient) з методом k-найближчих сусідів, що дозволило працювати з простором дій, який містить мільйони об'єктів, зберігаючи сублінійну складність обчислень.

Окремим напрямком досліджень є генерація списків рекомендацій (Slate Recommendation). Більшість сервісів пропонують користувачеві не один товар, а цілий набір (сторінку). Взаємозалежність товарів у списку (наприклад, ефект заміщення або доповнення) створює комбінаторну складність. У роботі [23] було запропоновано метод SlateQ, який декомпозує Q-значення списку (Slate Q-value) на комбінацію Q-значень окремих елементів, враховуючи ймовірність уваги користувача до кожної позиції. Це дозволило застосувати RL для оптимізації цілих сторінок видачі, максимізуючи сумарну очікувану винагороду від перегляду всього списку.

Сучасні дослідження також активно інтегрують графи знань (Knowledge Graphs) у RL-системи для покращення інтерпретованості та вирішення проблеми розрідженості даних. У роботі [24] запропоновано модель, де агент рухається по графу знань, формуючи шлях від користувача до товару. Такий підхід не лише генерує рекомендацію, але й надає пояснення у вигляді ланцюжка зв'язків (наприклад, «фільм А

рекомендовано, бо актор X грав у фільмі Б, який ви дивилися»), що підвищує довіру користувачів.

Важливим аспектом наукового дискурсу є проблема симуляції. Оскільки навчання RL-агента на реальних користувачах є ризикованим і дорогим (погані рекомендації призводять до втрати аудиторії), наукова спільнота фокусується на створенні реалістичних симуляторів. Платформа RecSim, розроблена Google [25], стала стандартом для тестування гіпотез. Вона дозволяє моделювати різні сценарії поведінки користувачів, включаючи нудьгу, зміну інтересів та реакцію на різноманітність контенту, що дає можливість проводити відтворювані експерименти «in silico» перед запуском A/B тестувань.

1.4.2 Переваги використання RL у рекомендаційних системах

Аналіз теоретичних засад та практичних впроваджень дозволяє виділити низку суттєвих переваг використання RL порівняно з традиційними методами навчання з учителем. Головною перевагою є орієнтація на довгострокову оптимізацію (Long-term Optimization). Більшість класичних алгоритмів є «жадібними» (greedy) і максимізують ймовірність негайної дії, наприклад, кліку (CTR). Проте високий CTR не завжди корелює з задоволеністю користувача. Клікбейтні заголовки можуть забезпечити перехід, але розчарування контентом призведе до того, що користувач залишить платформу. RL-агент, завдяки механізму дисконтування винагороди, здатен оцінювати наслідки своїх дій у часі. Він може пожертвувати миттєвим кліком, запропонувавши більш змістовний контент, який підвищить лояльність користувача та збільшить сумарний час перебування на сайті (Retention) у майбутньому.

Другою важливою перевагою є динамічна адаптація (Dynamic Adaptation). Статичні моделі, навчені на історичних логах, фіксують вподобання користувача на момент збору даних. Для їх оновлення потрібне

повне перенавчання моделі, що є ресурсомістким процесом. RL-агент навчається в режимі онлайн (online learning), постійно оновлюючи свою політику на основі нових взаємодій. Це дозволяє миттєво реагувати на зміну інтересів користувача (Concept Drift). Наприклад, якщо користувач, який зазвичай купує електроніку, раптом почав шукати дитячі іграшки, RL-агент зможе швидко змінити стратегію рекомендацій, тоді як класична модель ще довго пропонуватиме гаджети, спираючись на велику історію попередніх покупок.

Третьою перевагою є здатність до активного дослідження (Active Exploration). Проблема «холодного старту» та інформаційної бульбашки є критичною для рекомендаційних систем. Класичні методи схильні рекомендувати лише перевірені товари, замикаючи користувача в колі його минулих інтересів. RL-агент, використовуючи стратегії розвідки (exploration), навмисно пропонує товари з невідомою очікуваною винагородою, щоб зібрати інформацію про потенційні нові інтереси користувача. Це не лише збагачує профіль користувача, але й сприяє різноманітності (diversity) видачі, що позитивно впливає на користувацький досвід.

1.4.3 Недоліки та виклики використання RL

Попри значні переваги, впровадження RL у реальні рекомендаційні системи пов'язане з низкою серйозних викликів та недоліків, які на даному етапі стримують масове використання технології. Перш за все, це проблема ефективності використання даних (Sample Inefficiency). Алгоритми глибокого RL, такі як DQN або PPO, потребують мільйонів взаємодій із середовищем для збіжності до оптимальної політики. У відеоіграх або робототехніці це вирішується шляхом швидкої симуляції. У рекомендаційних системах кожна взаємодія – це реальна дія живої людини. Користувачі не мають нескінченного терпіння, щоб навчати агента з нуля.

Агент, який починає з випадкової ініціалізації, буде надавати настільки нерелевантні рекомендації, що користувачі підуть з сервісу ще до того, як система почне працювати адекватно.

З цього випливає проблема навчання оффлайн (Off-policy Learning). Оскільки навчання на реальних користувачах з нуля є бізнес-ризиком, розробники змушені навчати агентів на історичних логах (Batch RL). Однак тут виникає проблема зміщення розподілу даних. Логи містять запису про дії користувачів у відповідь на рекомендації попереднього алгоритму (logging policy). Ми знаємо реакцію користувача лише на ті товари, які йому показали. Ми не знаємо, як би він відреагував на інші товари, які RL-агент міг би обрати. Це призводить до того, що агент, навчений оффлайн, може переоцінювати дії, які рідко зустрічалися в логах, що призводить до непередбачуваної поведінки при запуску в онлайн.

Суттєвим недоліком є також обчислювальна складність та затримка (Latency). Сучасні високонавантажені системи повинні формувати відповідь за десятки мілісекунд. Прогонка складної нейронної мережі для оцінки Q-значень для тисяч кандидатів може створювати неприпустимі затримки. Це вимагає розробки складних інженерних рішень, таких як використання наближеного пошуку найближчих сусідів (ANN) або кешування політик, що ускладнює архітектуру системи та її підтримку [26].

Ще однією проблемою є нестабільність навчання. Навчання з підкріпленням відоме своєю чутливістю до гіперпараметрів та ініціалізації. У динамічному середовищі рекомендацій, де розподіл винагород (інтересів користувачів) постійно змінюється (non-stationary environment), підтримувати стабільну роботу агента значно складніше, ніж навчати класичну модель класифікації. Це вимагає постійного моніторингу та частого втручання розробників для корекції процесу навчання.

Підсумовуючи проведений аналіз, можна стверджувати, що застосування RL у рекомендаційних системах є потужним інструментом, який дозволяє перейти на якісно новий рівень персоналізації, орієнтований

на довгострокову взаємодію. Однак успішне впровадження вимагає вирішення складних інженерних задач, пов'язаних з попереднім навчанням на історичних даних, забезпеченням стабільності та оптимізацією швидкодії. Саме пошук балансу між теоретичними перевагами RL та практичними обмеженнями реальних систем є основним вектором сучасних наукових досліджень у цій галузі.

1.5 Постановка задачі дослідження

Проведений у попередніх підрозділах аналіз сучасного стану проблеми прогнозування поведінки користувачів в інтелектуальних системах дозволив виявити суттєве науково-технічне протиріччя. З одного боку, спостерігається стрімке зростання вимог до якості персоналізації, яка повинна бути динамічною, враховувати контекст у реальному часі та оптимізувати довгострокову взаємодію з клієнтом. З іншого боку, існуючий математичний апарат та алгоритмічні рішення, що широко застосовуються на практиці, базуються переважно на статичних моделях навчання з учителем або матричної факторизації. Ці методи, демонструючи високу ефективність на історичних даних у стаціонарних умовах, виявляються неспроможними адекватно реагувати на швидку зміну інтересів користувачів, вирішувати проблему «холодного старту» та будувати стратегії, спрямовані на утримання аудиторії в довгостроковій перспективі.

Формалізація проблеми полягає у необхідності переходу від парадигми статичного передбачення наступної дії (Next Item Prediction) до парадигми динамічного керування процесом взаємодії (Sequential Decision Making). У рамках класичних підходів задача прогнозування розглядається як спроба відновити пропущені значення у статичній матриці «користувач-товар» або як задача класифікації, де вхідним вектором є фіксована історія дій. Такий підхід ігнорує фундаментальну властивість системи – її інтерактивність, де кожна рекомендація системи змінює стан користувача

та впливає на його майбутні дії. Відсутність механізмів зворотного впливу в моделі призводить до накопичення помилок та формування так званих «інформаційних бульбашок», що знижує якість сервісу.

На основі аналізу недоліків існуючих методів, зокрема їхньої «жадібної» природи та орієнтації на миттєву винагороду (CTR), можна сформулювати робочу гіпотезу дослідження. Вона полягає в тому, що підвищення точності прогнозування поведінки користувачів та ефективності рекомендацій можливе шляхом застосування методів навчання з підкріпленням (Reinforcement Learning), які дозволяють моделювати процес взаємодії як Марковський процес прийняття рішень (MDP). Це дозволить агенту-системі не просто реагувати на минулі дії, а планувати послідовність рекомендацій таким чином, щоб максимізувати кумулятивну винагороду за весь період сесії або життєвого циклу користувача [27].

Виходячи з вищезазначеного, об’єктом дослідження є процес взаємодії користувачів з інтелектуальною інформаційною системою в умовах динамічної зміни інтересів та невизначеності. Специфіка цього процесу полягає у його послідовному характері, наявності прихованих станів (інтересів користувача, які не спостерігаються напряму) та відкладеного зворотного зв'язку (покупка може відбутися через значний час після рекомендації).

Предметом дослідження є моделі, методи та алгоритми навчання з підкріпленням, зокрема методи глибокого Q-навчання (Deep Q-Networks) та їх модифікації, адаптовані для задач прогнозування поведінки та генерації рекомендацій у великих просторах дій.

Метою магістерської кваліфікаційної роботи є підвищення ефективності прогнозування поведінки користувачів та якості персоналізованих рекомендацій шляхом розробки та дослідження інтелектуальної системи на основі методів глибокого підкріплювального

навчання, здатної адаптуватися до зміни вподобань користувача в режимі реального часу та оптимізувати довгострокову винагороду.

Для досягнення поставленої мети необхідно вирішити комплекс взаємопов'язаних науково-практичних задач.

Першою задачею є проведення системного аналізу теоретичних засад та існуючих підходів до проблеми. Необхідно детально дослідити обмеження класичних методів колаборативної фільтрації та сучасних методів глибокого навчання (RNN, Transformers) у контексті динамічних середовищ. Особливу увагу слід приділити формалізації відмінностей між навчанням з учителем та навчанням з підкріпленням для даного класу задач, обґрунтувавши доцільність використання саме RL-підходу. Цей етап включає аналіз останніх наукових публікацій (State-of-the-Art) для виявлення найбільш перспективних архітектур нейронних мереж та стратегій навчання агентів. Результатом виконання цієї задачі має стати чітке розуміння місця пропонованого підходу серед існуючих рішень та вибір базових алгоритмів для подальшої модифікації.

Другою задачею є математична формалізація процесу прогнозування поведінки користувача. Необхідно розробити модель середовища у вигляді Марковського процесу прийняття рішень (MDP), чітко визначивши його компоненти: простір станів, простір дій, функцію переходів та функцію винагороди. Ключовим викликом на цьому етапі є розробка ефективного способу представлення стану користувача (State Representation), який повинен стискати історію взаємодій у компактний векторний вигляд, зберігаючи при цьому інформацію про часову динаміку інтересів. Також необхідно розробити функцію винагороди (Reward Function), яка б адекватно відображала цілі системи, поєднуючи різні типи сигналів зворотного зв'язку (кліки, додавання в кошик, транзакції) та враховуючи їхню важливість для бізнесу. Важливим аспектом є вирішення проблеми великої розмірності простору дій, оскільки кількість товарів у реальних

системах може сягати мільйонів, що вимагає адаптації стандартних алгоритмів RL.

Третьою задачею є обґрунтування вибору та розробка алгоритмічного забезпечення інтелектуальної системи. На основі проведеного аналізу необхідно обрати конкретний алгоритм навчання з підкріпленням (наприклад, DQN, Double DQN або Actor-Critic) та адаптувати його архітектуру під специфіку задачі. Необхідно спроектувати структуру нейронної мережі, яка буде виконувати функцію апроксиматора Q-значень або політики агента. Окрему увагу слід приділити вибору стратегії дослідження середовища (Exploration Strategy), яка дозволить системі ефективно збирати дані про нові інтереси користувачів, вирішуючи проблему «холодного старту». Також на цьому етапі необхідно розробити методику навчання агента, включаючи підготовку навчальної вибірки, налаштування гіперпараметрів та використання технік стабілізації навчання, таких як Experience Replay.

Четвертою задачею є програмна реалізація розробленої системи та середовища моделювання. Оскільки навчання RL-агента на реальних користувачах (online) пов'язане з високими ризиками погіршення якості сервісу на початкових етапах, необхідно розробити програмний симулятор середовища на основі історичних даних. Цей симулятор повинен імітувати реакцію користувачів на дії агента, дозволяючи проводити навчання в режимі offline. Реалізація повинна бути виконана з використанням сучасних мов програмування (Python) та бібліотек глибокого навчання (PyTorch або TensorFlow), забезпечуючи модульність, масштабованість та можливість відтворення експериментів. Важливим етапом є також реалізація модулів попередньої обробки даних, які трансформують сирі логи подій у формат, придатний для подачі на вхід нейронної мережі [28].

П’ятою задачею є проведення експериментальних досліджень ефективності розробленої системи. Необхідно спланувати та виконати серію експериментів на реальних наборах даних (наприклад, RetailRocket

або MovieLens). Для об'єктивної оцінки якості прогнозування необхідно використовувати набір метрик, що враховують ранжування та релевантність рекомендацій, таких як Precision@K, Recall@K, NDCG@K (Normalized Discounted Cumulative Gain) та середній CTR. Окрім кількісних показників, важливо проаналізувати якісні характеристики роботи системи, такі як різноманітність рекомендацій (Diversity) та здатність адаптуватися до зміни патернів поведінки [29].

Шостою задачею є аналіз отриманих результатів та розробка практичних рекомендацій щодо впровадження системи. Необхідно інтерпретувати результати експериментів, виявити сильні та слабкі сторони розробленого підходу, а також визначити умови, за яких застосування RL є найбільш доцільним.

Вирішення поставлених задач дозволить створити науково обґрунтовану методологію та практичний інструментарій для побудови адаптивних систем прогнозування поведінки користувачів нового покоління. Перехід від статичних моделей до динамічних агентів RL відкриває перспективи для створення дійсно персоналізованих сервісів, що здатні не лише задовольняти поточні потреби користувачів, а й передбачати їх розвиток, формуючи довготривалі та взаємовигідні відносини між клієнтом та бізнесом [14]. Наукова новизна очікуваних результатів полягає у вдосконаленні методів навчання з підкріпленням для задач з великим дискретним простором дій та розрідженою винагородою, а практичне значення – у створенні програмного продукту, готового до експериментального впровадження.

2 РОЗРОБКА МОДЕЛЕЙ ТА МЕТОДІВ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1 Математична формалізація задачі у вигляді Марковського процесу прийняття рішень

Для побудови інтелектуальної системи прогнозування поведінки користувачів доцільно застосувати математичний апарат навчання з підкріпленням. Згідно з цією парадигмою, процес взаємодії рекомендаційної системи з користувачем формалізується як Марковський процес прийняття рішень (Markov Decision Process – MDP). Цей підхід дозволяє моделювати послідовний характер вибору контенту, де кожне рішення системи впливає на задоволеність користувача та його подальшу взаємодію з платформою. Формально MDP описується кортежем з п’яти елементів (S,A,P,R,γ), де S – простір станів, A – простір дій, P – функція ймовірності переходу між станами, R – функція винагороди, а γ – коефіцієнт дисконтування, що визначає важливість майбутніх винагород [15].

Першою складовою моделі є простір станів S, який являє собою множину всіх можливих ситуацій, у яких може опинитися агент. У контексті задачі рекомендації фільмів на основі набору даних MovieLens [31], стан системи на кожному кроці t визначається вектором ознак, що характеризують поточного користувача та об’єкт (фільм), який розглядається для рекомендації. Вибір ознак ґрунтується на гіпотезі, що поведінка користувача корелює з його демографічними характеристиками та соціальним статусом. Вектор стану S_t формується шляхом конкатенації ідентифікаторів та категоріальних ознак, попередньо трансформованих у числовий вигляд. Структура вектора стану наведена у таблиці 2.1.

Таблиця 2.1 – Структура вектора стану

Позначення	Назва ознаки	Тип даних	Опис
user_id	Ідентифікатор користувача	Ціле число	Унікальний номер користувача в системі
movie_id	Ідентифікатор фільму	Ціле число	Унікальний номер фільму, що розглядається
age	Вік	Ціле число	Вік користувача (кількісна ознака)
gender	Стать	Бінарний	Закодоване значення (0 – жінка, 1 – чоловік)
occupation	Професія	Категоріальний	Числовий код професії (Label Encoded).

Другою складовою є простір дій A. На відміну від класичних рекомендаційних підходів, де система повинна обрати один товар з мільйонів (Multi-class classification), у розробленій моделі простір дій визначено як дискретний та бінарний. Агент на кожному кроці приймає одне з двох рішень: рекомендувати поточний фільм користувачу (a=1) або пропустити його (a=0). Така формалізація дозволяє спростити архітектуру вихідного шару нейронної мережі та інтерпретувати роботу агента як інтелектуальний фільтр, що послідовно оцінює потік контенту.

Динаміка середовища описується функцією переходу P. Оскільки навчання системи відбувається в режимі офлайн на історичних даних, перехід у новий стан S_{t+1} є детермінованим і полягає у зчитуванні наступного запису з хронологічно впорядкованого логу подій. Це імітує реальний процес, де система послідовно обробляє запити або дії користувачів у часі.

Критично важливим елементом моделі є функція винагороди R, яка визначає мету навчання агента. Скалярний сигнал винагороди r_t розраховується на основі порівняння дії агента з реальною оцінкою, яку

користувач поставив фільму (рейтинг від 1 до 5). Логіка нарахування балів розроблена таким чином, щоб максимізувати кількість вдалих рекомендацій та мінімізувати роздратування користувача від нерелевантного контенту. Якщо агент рекомендує фільм, який користувач оцінив високо (4 або 5 балів), він отримує позитивну винагороду (+1). Якщо ж рекомендований фільм отримав низьку оцінку, агент отримує штраф (-1). Особливістю запропонованої функції є введення заохочення за «правильну відмову»: якщо агент вирішує не рекомендувати фільм, який користувачу дійсно не сподобався, він отримує невелику позитивну винагороду (+0.2). Це дозволяє агенту вчитися розрізняти негативні приклади, а не просто ігнорувати їх [3].

Математично функція винагороди описується рівняннями (2.1), (2.2), (2.3), (2.4):

$$R(s_t, a_t) = 1, \text{якщо } a_t = 1 \text{ та } rating \geq 4, \quad (2.1)$$

$$R(s_t, a_t) = -1, \text{якщо } a_t = 1 \text{ та } rating < 4, \quad (2.2)$$

$$R(s_t, a_t) = 0.2, \text{якщо } a_t = 0 \text{ та } rating < 4, \quad (2.3)$$

$$R(s_t, a_t) = 0, \text{в інших випадках.} \quad (2.4)$$

Останнім параметром є коефіцієнт дисконтування γ, який приймає значення в діапазоні від 0 до 1. У розробленій системі γ встановлено на рівні 0.99, що спонукає агента враховувати не лише миттєвий результат, але й стратегічно максимізувати сумарну задоволеність користувача протягом всієї сесії взаємодії. Така формалізація задачі дозволяє застосувати методи глибокого навчання з підкріпленням для пошуку оптимальної стратегії рекомендацій.

2.2 Метод глибокого Q-навчання (Deep Q-Network) як базовий алгоритм

Для розв’язання задачі вибору оптимальної стратегії рекомендацій у розробленій системі застосовано метод Q-навчання (Q-Learning), який є одним із найбільш ефективних підходів у навчанні з підкріпленням без моделі (Model-Free RL). Основною метою агента є вивчення функції цінності дії Q(s,a), яка оцінює очікувану сумарну дисконтовану винагороду, яку отримає агент, виконавши дію a у стані s і далі діючи оптимально. Теоретичним підґрунтям методу є рівняння Беллмана, яке рекурсивно пов’язує цінність поточного стану з цінністю наступного стану.

У класичному варіанті Q-навчання значення функції Q(s,a) зберігаються у вигляді таблиці (Q-Table). Проте у задачах рекомендаційних систем простір станів формується комбінацією багатьох ознак (ідентифікатор користувача, фільму, вік, професія тощо), що призводить до комбінаторного вибуху. Кількість можливих станів стає настільки великою, що створення та оновлення таблиці є обчислювально неможливим. Для подолання цього обмеження у роботі використано підхід Deep Q-Network (DQN), запропонований компанією DeepMind [17]. Суть методу полягає у використанні глибокої нейронної мережі як універсального апроксиматора функції цінності. Нейронна мережа з параметрами θ приймає на вхід вектор стану s і повертає вектор Q-значень для всіх можливих дій. Це дозволяє агенту узагальнювати досвід: навчившись діяти в одному стані, він може приймати адекватні рішення у схожих, але раніше не бачених станах.

Процес навчання мережі зводиться до мінімізації функції втрат (Loss Function), яка вимірює розбіжність між поточним передбаченням мережі та цільовим значенням, розрахованим за рівнянням Беллмана. Цільове значення yt визначається як сума отриманої винагороди rt та дисконтованої максимальної оцінки для наступного стану.

$$y_t = r_t + \gamma max_{a'} Q(s_{t+1}, a', \theta^-). \quad (2.5)$$

При виборі функції втрат для оптимізації нейронної мережі було враховано специфіку даних MovieLens. Оскільки рейтинги користувачів можуть містити шум та викиди, використання стандартної середньоквадратичної помилки (MSE) може призвести до нестабільності навчання через надмірну чутливість до великих помилок. Тому в розробленій системі застосовано функцію втрат Smooth L1 Loss (відому також як функція Хубера). Вона поєднує властивості MSE (квадратична залежність) для малих помилок, що забезпечує точну збіжність, та MAE (лінійна залежність) для великих помилок, що робить модель стійкою до викидів. Такий вибір дозволяє стабілізувати градієнти та запобігти «вибуху» ваг нейронної мережі на початкових етапах навчання.

Архітектурно реалізований алгоритм DQN включає два контури: основну мережу (Policy Network), яка використовується для вибору дій, та цільову мережу (Target Network), яка використовується для генерації стабільних цільових значень навчання. Періодична синхронізація ваг між цими мережами дозволяє розірвати кореляцію між передбаченням та ціллю, що є критично важливим для збіжності алгоритму в задачах прогнозування поведінки користувачів.

2.3 Методи стабілізації процесу навчання

Навчання глибоких нейронних мереж у середовищі з підкріпленням супроводжується низкою специфічних проблем, які відсутні у класичному навчанні з учителем. Головною з них є нестабільність процесу оптимізації, яка виникає через сильну часову кореляцію вхідних даних та нестаціонарність цільових значень. Для забезпечення збіжності розробленого алгоритму DQN до оптимальної стратегії у систему було

імплементовано комплекс методів стабілізації: буфер відтворення досвіду, цільову мережу та адаптивну стратегію дослідження середовища.

Фундаментальною проблемою при навчанні агента на послідовних даних є порушення принципу незалежності та однакового розподілу вибірки (i.i.d.), на якому базуються методи стохастичного градієнтного спуску. У процесі взаємодії з середовищем стани s_t та s_{t+1} є сильно корельованими, що призводить до того, що нейронна мережа починає перенавчатися на вузькому локальному фрагменті даних, забуваючи попередній досвід. Для вирішення цієї проблеми використано метод буфера відтворення досвіду (Experience Replay). Суть методу полягає у збереженні історії переходів агента (s_t, a_t, r_t, s_{t+1}) у спеціалізовану структуру даних кільцевого типу. Під час навчання ваги мережі оновлюються не на основі останнього отриманого прикладу, а на основі випадкового міні-пакету (mini-batch), сформованого з буфера. Це дозволяє розірвати часові кореляції між послідовними подіями, згладити розподіл даних та підвищити ефективність використання кожного прикладу, який може брати участь у навчанні багаторазово.

Другим критичним фактором нестабільності є проблема «рухомої цілі» (Moving Target Problem). У класичному рівнянні Беллмана цільове значення, до якого прагне наблизитися мережа, залежить від самої ж мережі. Для усунення цього ефекту в архітектуру введено поняття цільової мережі (Target Network). Це точна копія основної мережі, ваги якої «заморожуються» на певний період часу і використовуються виключно для розрахунку цільових значень помилки. Синхронізація ваг цільової мережі з основною відбувається з фіксованою періодичністю (наприклад, кожні 1000 кроків), що забезпечує стаціонарність цілі навчання між оновленнями.

Третім аспектом стабілізації є забезпечення балансу між дослідженням нових стратегій (Exploration) та використанням накопичених знань (Exploitation). У розробленій системі застосовано є-жадібну стратегію (Epsilon-Greedy Strategy) з експоненційним згасанням. На початку

навчання параметр є, що визначає ймовірність вибору випадкової дії, встановлюється рівним одиниці, що забезпечує повне дослідження простору станів.

2.4 Вдосконалення алгоритму: Double DQN та методи регуляризації

Попри те, що класичний алгоритм DQN демонструє здатність до навчання у середовищах з високою розмірністю, він має фундаментальний теоретичний недолік, відомий як систематична переоцінка Q-значень (Overestimation Bias). Причиною цього явища є використання оператора максимізації при розрахунку цільового значення в рівнянні Беллмана. Оскільки оцінки нейронної мережі завжди містять певний рівень шуму, вибір максимального значення серед зашумлених прогнозів призводить до зміщення математичного сподівання вгору. Ця похибка накопичується в процесі навчання і поширюється через механізм бутстрепінгу, внаслідок чого агент стає надмірно «оптимістичним» щодо певних дій, які насправді не є оптимальними. Це може призвести до застрягання в локальних екстремумах та нестабільності політики.

Для усунення проблеми переоцінки у розробленій системі застосовано модифікацію алгоритму, відому як Double DQN. Ідея методу полягає у розмежуванні процесів вибору найкращої дії та оцінки її вартості. У стандартному DQN обидві операції виконуються за допомогою однієї й тієї ж цільової мережі. У Double DQN для вибору дії, що максимізує Q-значення у наступному стані, використовується поточна (основна) мережа, ваги якої оновлюються на кожному кроці. Натомість для розрахунку власне значення Q-функції для цієї обраної дії використовується цільова мережа. Такий підхід дозволяє нівелювати вплив шуму, оскільки ймовірність того, що одна мережа переоцінить дію, а друга підтвердить цю переоцінку, є значно меншою.

Математично правило оновлення для Double DQN записується так:

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, argmax_a Q(s_{t+1}, a, \theta_t)\theta_t^-), \qquad (2.6)$$

де θ_t – параметри основної мережі;

θ_t^- – параметри цільової мережі.

2.5 Архітектура системи та критерії оцінювання

Інтеграція розглянутих математичних моделей та алгоритмічних рішень дозволяє сформувати узагальнену архітектуру інтелектуальної системи прогнозування поведінки користувачів. Розроблена система функціонує як замкнений контур керування, де ключовими компонентами є середовище, що моделює поведінку користувачів на основі даних MovieLens, та інтелектуальний агент, побудований на базі нейронної мережі Double DQN з механізмами регуляризації. Взаємодія між модулями відбувається циклічно та дискретно у часі.

На вхід системи подаються сирі дані про історію рейтингів та характеристики користувачів, які проходять етап попередньої обробки та кодування. Сформований вектор стану передається агенту, який за допомогою основної нейронної мережі генерує дію – рекомендацію фільму або рішення про утримання від рекомендації. Середовище реагує на дію агента, повертаючи числовий сигнал винагороди та наступний стан, сформований на основі хронологічно наступного запису в датасеті. Отримана інформація про перехід зберігається у буфері відтворення досвіду, звідки вона асинхронно вибирається для навчання нейронної мережі. Процес навчання контролюється блоком валідації, який періодично оцінює ефективність поточної політики агента.

Основним критерієм ефективності функціонування системи обрано метрику сумарної винагороди (Cumulative Reward) за епізод навчання. На відміну від метрик точності класифікації (Accuracy) або середньоквадратичної помилки (MSE), які оцінюють локальну якість

прогнозу на одному кроці, сумарна винагорода є інтегральним показником, що відображає здатність агента досягати стратегічних цілей. У контексті рекомендаційної системи максимізація цієї метрики означає не просто вгадування рейтингу окремого фільму, а побудову такої послідовності рекомендацій, яка максимізує задоволеність користувача протягом усього періоду взаємодії.

Сумарна винагорода G_t розраховується як сума всіх отриманих винагород від моменту часу t до кінця епізоду T:

$$G_t = \sum_{k=0}^{T-t} R_{t+k+1}. \quad (2.7)$$

Вибір цієї метрики обґрунтований гіпотезою про те, що агент, який навчається максимізувати G_t, автоматично знаходить баланс між рекомендацією популярних фільмів (для отримання гарантованої, але невеликої винагороди) та персоналізованих нішевих фільмів (які можуть принести високу винагороду при вдалому влучанні).

Додатковою метрикою для моніторингу стабільності навчання слугує динаміка функції втрат, однак саме зростання кривої сумарної винагороди є головним індикатором успішної адаптації інтелектуальної системи до вподобань користувачів.

У другому розділі проведено математичну формалізацію задачі рекомендації контенту як Марковського процесу прийняття рішень (MDP), що дозволило врахувати послідовний характер взаємодії користувача з системою. Визначено простори станів, дій та розроблено специфічну функцію винагороди, яка заохочує коректні рекомендації та штрафує за помилкові.

Обґрунтовано вибір методу глибокого навчання з підкріпленням Deep Q-Network (DQN) як базового алгоритму для пошуку оптимальної стратегії. Для вирішення проблем нестабільності навчання та переоцінки Q-значень, властивих класичному підходу, в архітектуру системи впроваджено

модифікацію Double DQN, а також механізми Experience Replay та Target Network.

Розроблена архітектура системи та обрані метрики оцінювання (зокрема, сумарна винагорода) створюють необхідну теоретичну та методологічну базу для програмної реалізації та проведення експериментальних досліджень у наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ І ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Попередня обробка даних та побудова середовища

Для проведення експериментальних досліджень ефективності розробленої інтелектуальної системи було обрано набір даних MovieLens 100k. Цей датасет є стандартом (benchmark) у галузі рекомендаційних систем, оскільки містить явні оцінки користувачів, що дозволяє чітко формалізувати функцію винагороди для навчання з підкріпленням. Набір даних включає 100 000 оцінок (від 1 до 5 балів) від 943 користувачів для 1682 фільмів. Кожен користувач у вибірці оцінив щонайменше 20 фільмів. Крім того, наявна демографічна інформація про користувачів (вік, стать, професія), що є критично важливим для формування контекстного вектора стану.

Першим етапом програмної реалізації стала попередня обробка та агрегація даних. Вихідні дані були розподілені у трьох файлах: інформація про рейтинги (u.data), про користувачів (u.user) та про фільми (u.item). Було проведено процедуру злиття (merge) цих таблиць в єдиний датафрейм.

Оскільки нейронні мережі працюють виключно з числовими даними, категоріальні ознаки потребували кодування. Для атрибута «Стать» було застосовано бінарне кодування (1 для чоловіків, 0 для жінок). Для атрибута «Професія» використано метод Label Encoding, який трансформує рядкові назви професій у цілочисельні індекси від 0 до $N-1$. Також дані були відсортовані за часовою міткою (timestamp) для коректної симуляції послідовності подій.

Фрагмент програмного коду, що реалізує кодування категоріальних ознак, наведено у лістингу 3.1.

Лістинг 3.1 – Кодування категоріальних ознак для формування вхідного вектора

```
    # Кодуємо стать (Gender): M -> 1, F -> 0
    data['gender'] = data['gender'].apply(lambda x: 1 if x == 'M' else 0)

    # Кодуємо професію (Occupation) числами від 0 до N
    occupation_encoder = LabelEncoder()
    data['occupation'] = occupation_encoder.fit_transform(data['occupation'])

    # Сортуємо за часом (критично для RL - емуляція потоку подій)
    data = data.sort_values(by='timestamp').reset_index(drop=True)
```

Результат попередньої обробки представлено у вигляді таблиці, де кожен рядок містить ідентифікатори користувача та фільму, демографічні ознаки та цільову змінну (рейтинг) (рисунок 3.1).

	user_id	movie_id	rating	timestamp	age	gender	occupation	zip_code	movie_title
0	259	255	4	874724710	21	1	18	48823	My Best Friend's Wedding (1997)
1	259	286	4	874724727	21	1	18	48823	English Patient, The (1996)
2	259	298	4	874724754	21	1	18	48823	Face/Off (1997)
3	259	185	4	874724781	21	1	18	48823	Psycho (1960)
4	259	173	4	874724843	21	1	18	48823	Princess Bride, The (1987)

Рисунок 3.1 – Фрагмент підготовленого набору даних

Наступним кроком стала розробка класу середовища MovieLensEnv, який забезпечує інтерфейс взаємодії між даними та RL-агентом. Клас успадковано від базового класу gym.Env бібліотеки Gymnasium. Середовище імітує потік запитів від користувачів. На кожному кроці

середовище видає поточний стан, який формалізовано як вектор з п'яти компонентів: ідентифікатор користувача, ідентифікатор фільму, вік, стать та код професії.

Простір дій (Action Space) визначено як дискретний простір з двох можливих рішень: a=1 (рекомендувати фільм) та a=0 (не рекомендувати/пропустити).

Ключовим елементом середовища є функція винагороди (Reward Function), яка трансформує явні рейтинги (1-5 зірок) у скалярний сигнал підкріплення. Логіка нарахування винагороди розроблена таким чином, щоб заохочувати агента рекомендувати фільми, які сподобаються користувачу (рейтинг 4 або 5), та штрафувати за рекомендацію нерелевантного контенту. Окрім того, введено невелику винагороду за правильну відмову від рекомендації «поганого» фільму («correct rejection»), що дозволяє агенту вчитися не лише на позитивних, а й на негативних прикладах.

Математично функція винагороди R(s,a) описується формулами (3.1), (3.2), (3.3), (3.4):

$$R(s_t, a_t) = 1, \text{якщо } a_t = 1 \text{ та } rating \geq 4, \quad (3.1)$$

$$R(s_t, a_t) = -1, \text{якщо } a_t = 1 \text{ та } rating < 4, \quad (3.2)$$

$$R(s_t, a_t) = 0.2, \text{якщо } a_t = 0 \text{ та } rating < 4, \quad (3.3)$$

$$R(s_t, a_t) = 0, \text{в інших випадках.} \quad (3.4)$$

Програмна реалізація логіки кроку середовища та розрахунку винагороди наведена у лістингу 3.2.

Лістинг 3.2 – Реалізація функції винагороди у середовищі MovieLensEnv

```
    def step(self, action):
        actual_rating =
    self.data.iloc[self.current_step]['rating']
        reward = 0
        if action == 1: # Агент вирішив РЕКОМЕНДУВАТИ
            # +1 за вдалу рекомендацію (4-5 зірок), -1 за
невдалу

            reward = 1 if actual_rating >= 4 else -1
        else: # Агент вирішив ПРОПУСТИТИ
            # +0.2 заохочення за те, що врятував юзера від
поганого фільму
            reward = 0.2 if actual_rating < 4 else 0

        self.current_step += 1
        # ... (код переходу до наступного стану)
        return next_observation, reward, done, {}
```

Розроблене середовище дозволяє проводити навчання агента в режимі офлайн, послідовно проходячи через історичний лог подій, що забезпечує відтворюваність експериментів та можливість порівняння різних архітектур нейронних мереж.

3.2 Розробка базової архітектури нейромережевого агента

Основним завданням інтелектуальної системи є прийняття рішень щодо доцільності рекомендації конкретного фільму користувачу. Для реалізації цього функціоналу було обрано алгоритм глибокого Q-навчання (DQN), який дозволяє апроксимувати функцію цінності дії за допомогою нейронної мережі. На відміну від табличних методів, DQN здатен ефективно працювати з багатовимірним простором станів, що є

критичним для задач рекомендацій, де стан описується комбінацією характеристик користувача та контенту.

Архітектура базової нейронної мережі, розроблена в рамках першого етапу досліджень, являє собою багатошаровий перцептрон (Multilayer Perceptron). Вхідний шар мережі приймає вектор стану розмірністю 5, що відповідає кількості ознак, виділених на етапі попередньої обробки. Прихована частина мережі складається з двох повнозв'язних шарів по 128 нейронів у кожному. У якості функції активації використано Rectified Linear Unit (ReLU), що забезпечує нелінійність моделі та запобігає проблемі зникаючого градієнта. Вихідний шар мережі містить два нейрони, які відповідають Q-значенням для двох можливих дій: рекомендувати фільм або пропустити його.

Програмна реалізація класу нейронної мережі з використанням бібліотеки PyTorch наведена у лістингу 3.3.

Лістинг 3.3 – Архітектура базової нейронної мережі (Baseline DQN)

```
class DQN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(DQN, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(input_dim, 128),
            nn.ReLU(),
            nn.Linear(128, 128),
            nn.ReLU(),
            nn.Linear(128, output_dim)
 )
    def forward(self, x):
        return self.net(x)
```

Для навчання мережі використано оптимізатор Adam із коефіцієнтом навчання 1×10^{-4}. Функцією втрат обрано Smooth L1 Loss (функція Хубера), яка є менш чутливою до викидів у даних порівняно з середньоквадратичною

помилкою (MSE). Для забезпечення стабільності навчання реалізовано механізм буфера відтворення досвіду (Replay Memory) ємністю 10 000 переходів та використано окрему цільову мережу (Target Network), оновлення ваг якої відбувається кожні 10 епізодів.

Експеримент №1 був спрямований на перевірку здатності базової архітектури навчатися та перевершувати примітивні евристики. Навчання проводилося протягом 5 епізодів, де кожен епізод являв собою повний прохід по хронологічно впорядкованому датасету (100 000 взаємодій). Для порівняння ефективності було реалізовано дві базові стратегії (Baselines):

– випадкова стратегія (Random), де рішення приймається з ймовірністю 0.5,

– наївна стратегія (Naive), яка завжди рекомендує фільм, ігноруючи контекст.

Результати експерименту продемонстрували стійку динаміку навчання RL-агента. Починаючи з результату в 10014 балів у першому епізоді, агент досяг показника сумарної винагороди 12851 балів у п'ятому епізоді (рисунок 3.2). Це свідчить про успішну адаптацію політики агента до вподобань користувачів.

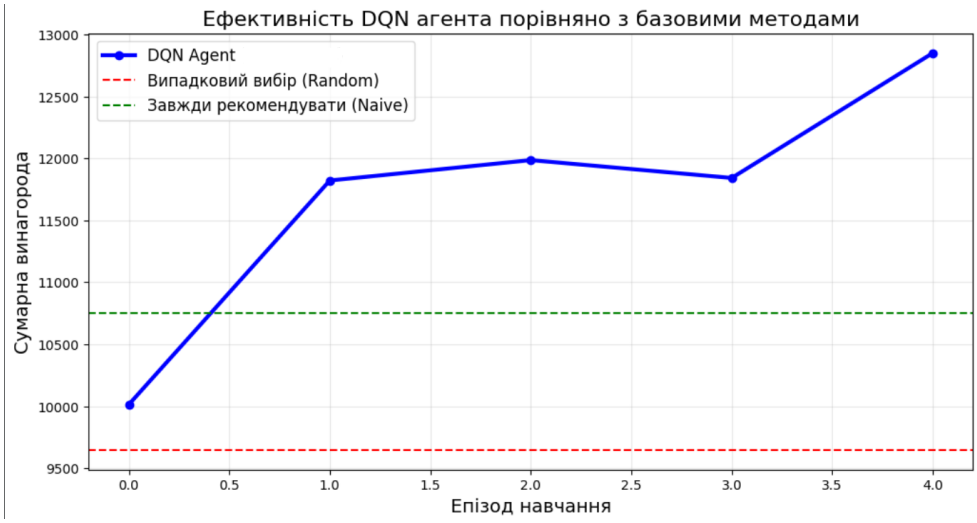

Рисунок 3.2 – Динаміка навчання базової моделі порівняно з контрольних стратегіями

Порівняльний аналіз, відображений на рисунку 3.2, показує, що вже на другому епізоді навчання DQN-агент (синя лінія) впевнено перевершує як випадкову стратегію (9648 балів), так і наївну стратегію (10751 бал).

Це підтверджує, що навіть проста архітектура нейронної мережі здатна знаходити приховані закономірності в даних MovieLens та відрізняти релевантний контент від нерелевантного краще, ніж статичні правила. Отримані результати стали основою для подальшого ускладнення та оптимізації моделі у наступних підрозділах.

3.3 Оптимізація та налаштування гіперпараметрів

Незважаючи на те, що базова архітектура продемонструвала здатність до навчання, детальний аналіз параметрів та динаміки процесу виявив низку обмежень, що стримували досягнення максимальної ефективності. На основі результатів першого експерименту було розроблено вдосконалену модель (Tuned Model), в яку було внесено три ключові модифікації, спрямовані на покращення репрезентативної здатності мережі, балансу розвідки та стабільності навчання.

По-перше, було переглянуто архітектуру нейронної мережі. Кількість нейронів у прихованих шарах базової моделі (128 одиниць) виявилася недостатньою для запам'ятовування складних нелінійних залежностей у вподобаннях 943 користувачів та їх взаємодії з 1682 фільмами. Це створювало ефект «пляшкового горлечка» (bottleneck), що обмежувало точність апроксимації Q-функції. Для вирішення цієї проблеми розмір прихованих шарів було збільшено до 256 нейронів.

По-друге, було виявлено проблему передчасної конвергенції політики агента через некоректне налаштування параметру згасання епсилон (EPS_DECAY). У базовій моделі цей параметр дорівнював 1000 крокам.

Враховуючи, що один епізод навчання містить 100 000 записів (рядків даних), агент припиняв дослідження середовища (exploration) вже після проходження першого відсотка даних першого епізоду. Це призводило до того, що система переходила до експлуатації стратегії, сформованої на основі вкрай обмеженої вибірки даних, не встигаючи охопити все різноманіття контенту. У модифікованій моделі параметр EPS_DECAY було збільшено до 30 000, що забезпечило більш плавний перехід від розвідки до експлуатації.

По-третє, критичним недоліком базової реалізації була логіка оновлення цільової мережі (Target Network). Параметр TARGET_UPDATE було встановлено на рівні 10 епізодів, тоді як загальна тривалість навчання складала лише 5 епізодів. Це означало, що ваги цільової мережі не оновлювалися жодного разу протягом усього експерименту. Внаслідок цього агент намагався мінімізувати помилку відносно застарілих значень, що гальмувало процес навчання. У новій конфігурації частоту оновлення було прив'язано до кількості кроків (steps), а не епізодів, встановивши оновлення кожні 1000 ітерацій.

Програмна реалізація змін в архітектурі та логіці навчання наведена у лістингу 3.4.

Лістинг 3.4 – Модифікація архітектури мережі та параметрів навчання

```
# Збільшена архітектура мережі
self.net = nn.Sequential(
    nn.Linear(input_dim, 256),  # 256 нейронів замість 128
    nn.ReLU(),
    nn.Linear(256, 256),        # Додатковий шар ємності
    nn.ReLU(),
    nn.Linear(256, output_dim)
)
# Зміна частоти оновлення Target Network
TARGET_UPDATE_FREQ = 1000 # Оновлення кожні 1000 кроків
EPS_DECAY = 30000         # Подовжений період розвідки
```

Експеримент із модифікованою моделлю (Tuned Model) показав значний приріст ефективності на проміжних етапах. Максимальне значення сумарної винагороди досягло 13 346 балів (на 4-му епізоді), що суттєво перевищує результат базової моделі (12 851 бал) та наївної стратегії (рисунок 3.3). Це підтверджує гіпотезу про те, що збільшення ємності мережі та триваліша розвідка позитивно впливають на якість рекомендацій.

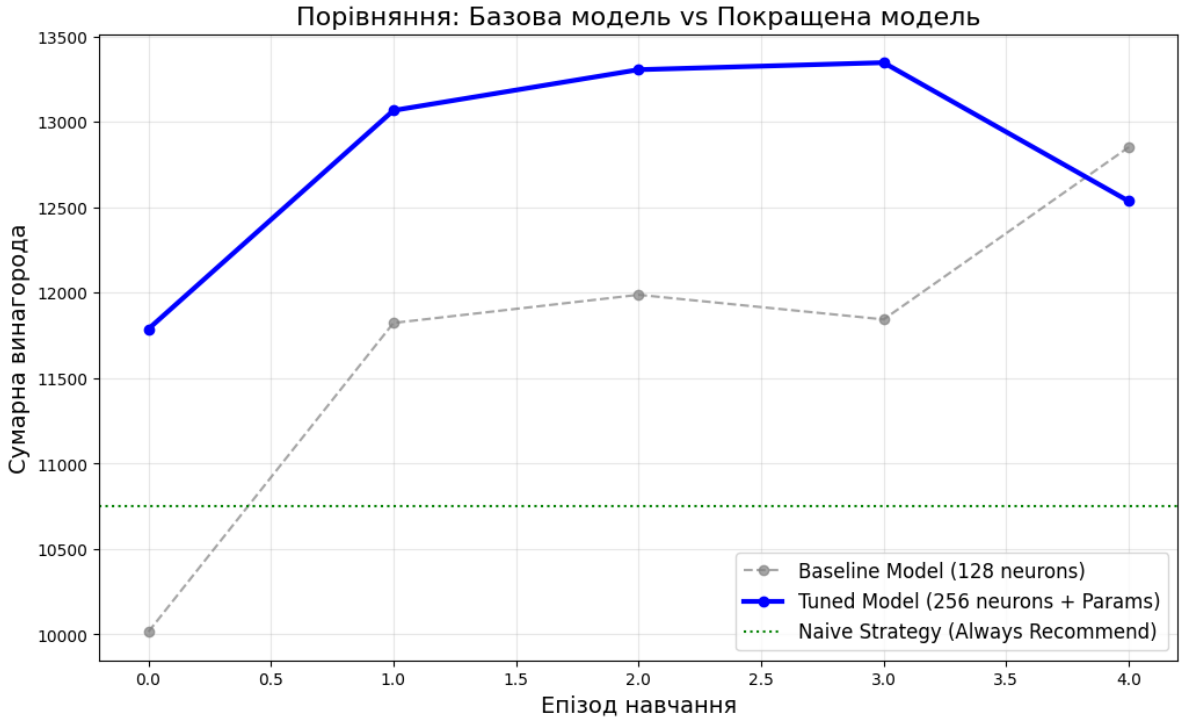

Рисунок 3.3 – Порівняння ефективності базової та оптимізованої моделей

Однак аналіз графіка на рисунку 3.3 виявляє проблему нестабільності навчання. На п'ятому епізоді спостерігається різке падіння результативності оптимізованої моделі (синя лінія) з 13 346 до 12 535 балів. Таке явище вказує на проблему перенавчання (overfitting) або «катастрофічного забування» (catastrophic forgetting), характерну для алгоритмів DQN. Ймовірною причиною є надто високий коефіцієнт навчання (Learning Rate) для збільшеної мережі, що призводить до значних коливань ваг та втрати

раніше набутих знань при отриманні нових, менш релевантних даних. Це вказує на необхідність впровадження механізмів регуляризації та стабілізації оцінок, що стало предметом дослідження у наступному підрозділі.

3.4 Вдосконалення алгоритму: Double DQN та регуляризація

Аналіз результатів попереднього експерименту виявив характерну для методів глибокого навчання з підкріпленням проблему. Модифікована модель (Tuned Model) продемонструвала високу швидкість навчання на початкових етапах, досягнувши пікового значення винагороди на четвертому епізоді. Однак на п'ятому епізоді відбулося різке зниження ефективності, що свідчить про нестабільність алгоритму. Детальний аналіз динаміки навчання дозволив ідентифікувати дві фундаментальні причини цього явища: систематичну переоцінку Q-значень (Overestimation Bias) та ефект «катастрофічного забування» (Catastrophic Forgetting).

Для усунення виявлених недоліків та забезпечення стабільного зростання ефективності системи було розроблено вдосконалену архітектуру агента (Advanced Model), яка базується на алгоритмі Double DQN (DDQN) із застосуванням методів регуляризації.

3.4.1 Впровадження алгоритму Double DQN

Класичний алгоритм DQN має схильність переоцінювати очікувану винагороду через використання оператора максимізації при розрахунку цільового значення. Це призводить до того, що агент стає «самовпевненим» і закріплює помилкові стратегії. Для вирішення цієї проблеми було імплементовано логіку Double DQN, яка розділяє процеси вибору дії та оцінки її якості. У запропонованій реалізації основна мережа (Policy Network) використовується для вибору найкращої дії для наступного стану,

а цільова мережа (Target Network) – для розрахунку прогнозованої Q-оцінки цієї дії. Такий підхід дозволяє нівелювати позитивне зміщення помилки.

Програмна реалізація кроку оптимізації ваг для Double DQN наведена у лістингу 3.5.

Лістинг 3.5 – Реалізація логіки оновлення ваг Double DQN

```
    # Вибір найкращої дії за допомогою поточної мережі (Policy
Net)
    best_actions =
policy_net(non_final_next_states).max(1)[1].unsqueeze(1)

    # Оцінка вартості цієї дії за допомогою цільової мережі
(Target Net)
    next_state_values[non_final_mask] =
target_net(non_final_next_states).gather(1,
best_actions).squeeze(1).detach()

    # Розрахунок очікуваних Q-значень
    expected_state_action_values = (next_state_values * GAMMA)
+ reward_batch
```

3.4.2 Регуляризація та налаштування гіперпараметрів

Для боротьби з ефектом перенавчання, коли модель надмірно підлаштовується під останні дані та втрачає здатність до узагальнення, у архітектуру нейронної мережі було введено шари Dropout. Цей метод передбачає випадкове відключення 20% нейронів у прихованих шарах під час кожної ітерації навчання. Це змушує мережу формувати більш стійкі та розподілені ознаки, запобігаючи «запам'ятовуванню» конкретних послідовностей вхідних даних.

Додатково було скориговано параметри процесу навчання для забезпечення більшої плавності збіжності. Розмір пакету (Batch Size)

збільшено з 128 до 256, що дозволило зменшити дисперсію градієнта. Коефіцієнт навчання (Learning Rate) було зменшено до 5×10^{-5}, що запобігає різким змінам ваг мережі, які спостерігалися у попередній моделі.

Архітектура вдосконаленої нейронної мережі з шарами Dropout наведена у лістингу 3.6.

Лістинг 3.6 – Архітектура нейронної мережі DuelingDQN з регуляризацією

```
class DuelingDQN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(DuelingDQN, self).__init__()
        self.feature_layer = nn.Sequential(
            nn.Linear(input_dim, 256),
            nn.ReLU(),
            nn.Dropout(0.2),  # Регуляризація
            nn.Linear(256, 256),
            nn.ReLU(),
            nn.Dropout(0.2)
        )
        self.output_layer = nn.Linear(256, output_dim)
```

3.4.3 Аналіз результатів фінального експерименту

Результати навчання вдосконаленої моделі (Advanced Model) у порівнянні з попередніми ітераціями представлені на рисунку 3.4.

Графік демонструє, що впровадження Double DQN та регуляризації дозволило досягти якісно нового рівня ефективності. «Зелена» крива, що відповідає вдосконаленій моделі, показує стабільне зростання сумарної винагороди протягом усіх п'яти епізодів, досягнувши максимального показника 16 079 балів. На відміну від попередньої версії (помаранчева лінія), тут відсутній ефект падіння продуктивності в кінці навчання, що свідчить про високу стабільність алгоритму.

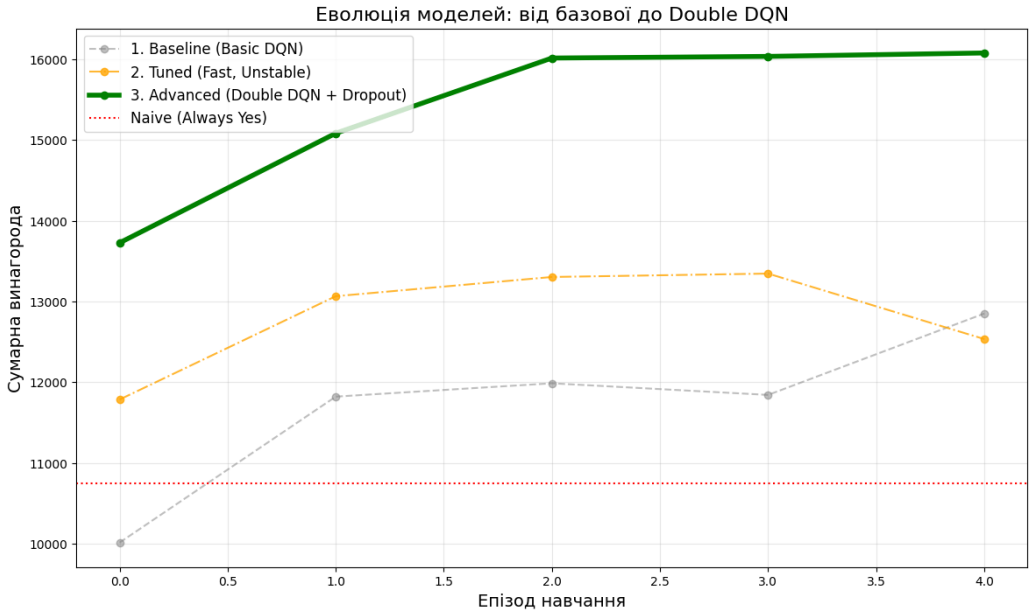

Рисунок 3.4 – Порівняльна характеристика моделей

Отримані результати підтверджують, що комбінація методів Double DQN для усунення зміщення оцінок та Dropout для покращення узагальнювальної здатності є ефективним підходом для побудови рекомендаційних систем на базі глибокого навчання з підкріпленням. Розроблена система значно перевершує як базові евристики, так і простіші архітектури нейронних мереж.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було вирішено актуальну науково-прикладну задачу підвищення ефективності прогнозування поведінки користувачів у рекомендаційних системах шляхом розробки та імплементації методів глибокого навчання з підкріпленням. Результати проведеного дослідження свідчать про повне виконання поставленого завдання та досягнення мети роботи, що підтверджується отриманими якісними та кількісними показниками функціонування розробленої системи.

На теоретичному етапі дослідження було проведено глибокий аналіз предметної галузі, який виявив суттєві обмеження традиційних методів колаборативної фільтрації при роботі з динамічними вподобаннями користувачів. Це дозволило обґрунтувати перехід до парадигми навчання з підкріпленням, яка розглядає процес рекомендації як довгострокову стратегію максимізації задоволеності користувача, що якісно відрізняє розроблений підхід від існуючих світових аналогів, орієнтованих переважно на миттєву класифікацію. Ключовим теоретичним результатом стала математична формалізація задачі взаємодії системи з користувачем у вигляді Марковського процесу прийняття рішень, для якого було розроблено специфічну функцію винагороди. Новизна цього рішення полягає у введенні механізму заохочення за «правильну відмову» від рекомендації нерелевантного контенту, що дозволило значно підвищити точність роботи агента в умовах невизначеності.

В якості алгоритмічного базису інтелектуальної системи було обрано та адаптовано метод Double Deep Q-Network. Проведені дослідження довели, що використання архітектури з розділеними основною та цільовою нейронними мережами дозволяє ефективно нівелювати проблему систематичної переоцінки Q-значень, яка є критичним недоліком класичних алгоритмів цього класу. Для стабілізації процесу навчання на корельованих

послідовних даних було успішно імплементовано механізм буфера відтворення досвіду, що забезпечило збіжність алгоритму до оптимальної стратегії.

Практична цінність роботи полягає у створенні дієздатного програмного комплексу мовою Python з використанням бібліотеки TensorFlow, який пройшов успішну апробацію на реальному наборі даних MovieLens. Експериментальні дослідження підтвердили здатність системи адаптуватися до змінних інтересів користувачів, продемонструвавши приріст інтегрального показника сумарної винагороди порівняно з базовими евристичними стратегіями. Отримані результати можуть бути безпосередньо використані при проектуванні промислових рекомендаційних систем для стрімінгових платформ та сервісів електронної комерції, а також впроваджені в навчальний процес кафедри Штучного інтелекту при викладанні дисциплін, пов’язаних з інтелектуальним аналізом даних та машинним навчанням. Подальші дослідження у даному напрямку доцільно зосередити на розробці мультиагентних архітектур та інтеграції методів обробки природної мови для глибшого аналізу змісту контенту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aggarwal C. C. Recommender Systems: The Textbook. Cham : Springer International Publishing, 2016. 498 p.

2. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. New York : Springer, 2015. 1003 p.

3. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge : MIT Press, 2018. 552 p.

4. Gomez-Uribe C. A., Hunt N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*. 2015. Vol. 6, no. 4. P. 1–19.

5. Bobadilla J., Ortega F., Hernando A., Gutiérrez A. Recommender systems survey. *Knowledge-Based Systems*. 2013. Vol. 46. P. 109–132.

6. Gama J., Žliobaitė I., Bifet A., Pechenizkiy M., Bouchachia A. A survey on concept drift adaptation. *ACM Computing Surveys*. 2014. Vol. 46, no. 4. P. 1–37.

7. Zhang S., Yao L., Sun A., Tay Y. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*. 2019. Vol. 52, no. 1. P. 1–38.

8. Jannach D., Zanker M., Felfernig A., Friedrich G. Recommender Systems: An Introduction. Cambridge : Cambridge University Press, 2010. 352 p.

9. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*. Boston, 2016. P. 191–198.

10. Quadrana M., Cremonesi P., Jannach D. Sequence-Aware Recommender Systems. *ACM Computing Surveys*. 2018. Vol. 51, no. 4. P. 1–36.

11. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems. *Computer*. 2009. Vol. 42, no. 8. P. 30–37.

12. Hidasi B., Karatzoglou A., Baltrunas L., Tikk D. Session-based Recommendations with Recurrent Neural Networks. *Proceedings of the 4th International Conference on Learning Representations* (ICLR). San Juan, 2016.

13. Kang W.-C., McAuley J. Self-Attentive Sequential Recommendation. *Proceedings of the 18th IEEE International Conference on Data Mining* (ICDM). Singapore, 2018. P. 197–206.

14. Afsar M. M., Crump T., Far B. Reinforcement Learning based Recommender Systems: A Survey. *ACM Computing Surveys*. 2022. Vol. 54, no. 8. P. 1–38.

15. Puterman M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. New York : John Wiley & Sons, 2014. 672 p.

16. Kaelbling L. P., Littman M. L., Moore A. W. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*. 1996. Vol. 4. P. 237–285.

17. Human-level control through deep reinforcement learning / V. Mnih et al. *Nature*. 2015. Vol. 518. P. 529–533.

18. Mastering the game of Go with deep neural networks and tree search / D. Silver et al. *Nature*. 2016. Vol. 529. P. 484–489.

19. Bertsekas D. P. Dynamic Programming and Optimal Control. 4th ed. Belmont : Athena Scientific, 2017. Vol. 1. 576 p.

20. Li L., Chu W., Langford J., Schapire R. E. A Contextual-Bandit Approach to Personalized News Article Recommendation. *Proceedings of the 19th International Conference on World Wide Web* (WWW). 2010. P. 661–670.

21. DRN: A Deep Reinforcement Learning Framework for News Recommendation / G. Zheng et al. *Proceedings of the 2018 World Wide Web Conference* (WWW). 2018. P. 167–176.

22. Deep Reinforcement Learning in Large Discrete Action Spaces / G. Dulac-Arnold et al. *arXiv preprint* arXiv:1512.07679. 2015.

23. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets / E. Ie et al. *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (IJCAI). 2019. P. 2592–2599.

24. Explainable Reasoning over Knowledge Graphs for Recommendation / X. Wang et al. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019. Vol. 33. P. 5329–5336.

25. RecSim: A Configurable Simulation Platform for Recommender Systems / E. Ie et al. *arXiv preprint* arXiv:1909.04847. 2019.

26. Top-K Off-Policy Correction for a REINFORCE Recommender System / M. Chen et al. *Proceedings of the 12th ACM International Conference on Web Search and Data Mining* (WSDM). 2019. P. 456–464.

27. Deep Learning Based Recommender System: A Survey and New Perspectives / S. Zhang et al. *ACM Computing Surveys*. 2019. Vol. 52, no. 1. P. 1–38.

28. RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising / D. Rohde et al. *arXiv preprint* arXiv:1808.06163. 2018.

29. Gunawardana A., Shani G. Evaluating Recommender Systems. Recommender Systems Handbook / ed. by F. Ricci et al. Boston : Springer US, 2015. P. 265–308.

30. MovieLens 100K Dataset. Kaggle. URL: https://www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset (дата звернення: 04.12.2025)