МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Застосування статистичних методів та методів штучного інтелекту для оптимізації процесу електронного навчання**

Виконала: студентка 2 курсу, групи 8.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
А.В. Пилипенко
(ініціали та прізвище)
Керівник доцент, к. т. н. Н. П. Полякова
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Інженерний навчально-науковий інститут
Кафедра програмного забезпечення автоматизованих систем
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ В.Г. Вербицький
"_01_" __вересня_____2020_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Пилипенко Анні Василівні
(прізвище, ім'я, по батькові)

1.Тема роботи Застосування статистичних методів та методів штучного інтелекту для оптимізації процесу електронного навчання

керівник роботи доцент, канд. техн. наук Полякова Наталія Петрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від _"25"_травня_2020_року № 600-с_____

2. Строк подання студентом кваліфікаційної роботи 30.11.2020
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми використання методів статистики та машинного навчання в оптимізації електронної освіти;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
13 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09 – 07.09.2020	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	07.09 – 10.09.2020	виконано
3	Аналіз існуючих методів рішення	10.09 – 16.09.2020	виконано
4	Дослідження області використання рекомендаційних систем	17.09 – 20.09.2020	виконано
5	Дослідження використання рекомендаційних систем у сфері електронного навчання	20.09 – 29.09.2020	виконано
6	Узгодження подальших дій з науковим керівником	29.09 – 30.09.2020	виконано
7	Аналіз теоретичних відомостей	30.09 – 03.10.2020	виконано
8	Проектування модулів системи рекомендацій	03.10 – 07.10.2020	виконано
9	Узгодження архітектури з науковим керівником	07.10 – 10.10.2020	виконано
10	Реалізація функціоналу рекомендаційної системи	10.10 – 07.11.2020	виконано
11	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	07.11 – 09.11.2020	виконано
12	Проведення аналізу можливостей системи	09.11 – 20.11.2020	виконано
13	Оформлення звіту	10.11 – 30.11.2020	виконано

Студент _____ Пилипенко А.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Полякова Н.П.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Застосування статистичних методів та методів штучного інтелекту для оптимізації процесу електронного навчання.

Кваліфікаційна робота магістра складається із введення, 4 розділів і висновків, списку використаних джерел з 30 найменувань. Робота містить 109 сторінок тексту, 45 рисунків, 3 таблиці, 8 лістингів коду.

Кваліфікаційна робота для здобуття ступеня вищої освіти магістра за спеціальністю 121 — Інженерія програмного забезпечення, науковий керівник Н.П. Полякова. Інженерний інститут ЗНУ. Інститут енергетики, електроніки та інформаційних технологій, 2020.

Мета роботи полягає у дослідженні та вивченні методів статистики та штучного інтелекту у сфері електронного навчання, порівняння їх особливостей, перевірка можливостей застосування і аналіз модуля рекомендаційної системи як інструменту для використання в навчальних платформах.

Досліджено методи і конкуруючі сучасні системи прогнозування та створення рекомендацій курсів, фільмів, відео, новин, товарів і можливості розробки і використання системи у розрізі платформи для електронної освіти. Порівняно методи штучного інтелекту для рекомендації навчальних матеріалів. Спроектовано та реалізовано два застосунки на мові програмування C#, кожен з яких виконує функцію прогнозування (рекомендації).

Ключові слова: *ПРОГРАМНА ПЛАТФОРМА, ЕЛЕКТРОННЕ НАВЧАННЯ, СТАТИСТИЧНІ МЕТОДИ, ПРОГНОЗУВАННЯ, РЕКОМЕНДАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, C#, .NET CORE, КОМП'ЮТЕРНИЙ ЗАСТОСУНОК.*

SUMMARY

Application of statistical methods and artificial intelligence methods to optimize the e-learning process.

Qualification work for higher master's degree in specialty 121 - Software Engineering, supervisor N.P. Polyakova. Engineering Institute ZNU. Faculty of Energy, Electronics and Information Technology, 2020.

The purpose of the work is to research and study the methods of statistics and artificial intelligence in the field of e-learning, compare their features, test the applicability and analyze the recommendation system module as a tool for use in educational platforms.

Methods and competing modern systems for predictions and creating recommendations for courses, films, videos, news, goods and the possibilities of developing and using the system in the context of a platform for e-education are investigated. Comparison of artificial intelligence methods for recommendation of e-learning materials. Designed and implemented two applications in the C # programming language, each of which performs the function of prediction (recommendation).

Keywords: SOFTWARE PLATFORM, E-LEARNING, STATISTICAL METHODS, PREDICTION, RECOMMENDATION, ARTIFICIAL INTELLIGANCE, MACHINE LEARNING, C#, .NET CORE, COMPUTER APPLICATION.

ЗМІСТ

ВСТУП

Актуальність теми

Електронне навчання — це реалізація освітніх програм з використанням інформаційно-освітніх ресурсів та за допомогою інформаційних та електронних технологій. Такий вид навчання не є новим на сьогоднішній день і вже пройшов ряд модифікацій та оптимізацій, що дозволило збільшити кількість переваг такого формату та позбутися ряду його недоліків. Але не зважаючи на це досі залишається невирішним ряд питань, таких як:

1) неможливість швидкого зворотнього зв’язку від викладача;

2) дуже велика кількість претендентів на онлайн класи, які розраховані на обмежену кількість людей;

3) неоптимізованість перевірки деяких видів тестових завдань (перевірка здійснюється особисто викладачем, що унеможливлює роботу з великою кількістю учнів;

4) мотивація учнів до такого формату навчання, боротьба зі зменшенням мотивації впродовж проходження курсу;

5) об’єктивність оцінювання учня у форматі електронної освітньої платформи.

На даний момент існують платформи, які реалізували деякі способи боротьби з недоліками даного формату. Популярними методами мотивації є гейміфікація процесу навчання, перехід до формату відео лекцій. Також дані платформи використовують в більшості автоматизовану перевірку тестів. Існують платформи, які також використовують системи штучного інтелекту, наприклад, для рекомендації нових курсів, які можуть бути цікаві конкретному учневі, виконують збір різного роду статистики про активність учнів, їх результати, тощо. Але головною проблемою у галузі залишається те, що всі ці методи додаються до вже давно спроектованих та працюючих систем. Це унеможливлює або значно подовжує і погіршує додання такого роду оптимізації.

Мета і завдання дослідження

Мета дослідження полягає у вивченні та дослідженні методів статистики та штучного інтелекту в оптимізації процесу електронного навчання, а саме:

1) аналіз початкової вибірки та зниження розмірності;

2) впровадження статистичних методів для оцінки впливу змін в курсах на активність учнів та їх результативність;

3) дослідження і порівняння різних реалізацій рекомендаційних систем.

Головною відмінністю розроблюваної системи від більшості вже існуючих є комбінація зазначених підходів і забезпечення мінімальної зв’язності між системою та модулями.

Об’єкт дослідження

Концепції опису сутностей та інші дані про користувачів та матеріали для побудови і дослідження ефективних рекомендаційних систем у сфері електронного навчання.

Предмет дослідження

Методи опису сутностей та рекомендаційні системи, проблема прогнозування вибору та вподобань користувача в галузі освіти. Також предметом дослідження є методи та алгоритми статистики та машинного навчання, на базі дослідження яких стає можливим розробити програмні застосунки та реалізувати їх взаємодію.

Методи дослідження

Для розв’язання представлених завдань використовуються такі методи дослідження:

1) аналіз джерел про попередню обробку даних вибірки, зниження розмірності даних та приклади використання зазначених методів;

2) аналіз джерел про види рекомендаційних систем, їх відмінності та сфери застосування;

3) проведення аналогії з-поміж існуючих на ринку платформ електронного навчання;

4) аналіз джерел про тенденції у розвитку сучасних платформ для електронного навчання а також основних проблем даної сфери;

5) порівняльний аналіз програмних продуктів;

6) пошук існуючих статистичних методів та методів штучного інтелекту, здатних допомогти у вирішенні поставлених проблем.

Наукова новизна одержаних результатів

Одержані результати є наочним відображення переваг та недоліків використання статистичних методів та рекомендаційних систем у сфері електронного навчання. На основі аналізу цих даних у майбутньому можлива розробка платформи для електронної освіти з використанням найкращих и найефективніших методів взаємодії з користувачем, передбачення його поведінки та надання якісних і актуальних для нього знань.

Розроблені модулі прогнозування мови можна інтегрувати в будь-яку систему та пристосувати до предметної області.

Нажаль, відкритих систем прогнозування та рекомендацій для платформ електронної освіти майже немає, а ті які функціонують, мають дуже великий відсоток помилок. Тому створений застосунок покликаний полегшити інтеграцію машинного навчання у веб системи для електронного навчання.

Практичне значення одержаних результатів

На базі отриманих результатів можна зрозуміти які засоби реалізації рекомендаційних систем є найбільш актуальними для конкретної галузі. Також проаналізувавши дану роботу можна зробити висновки щодо найбільш вдалої реалізації системи рекомендацій та попередньої обробки вибірки для такого роду систем. Ознайомитися з найпоширенішими проблемами шо виникають у процесі створення такого програмного забезпечення, та методами їх вирішення.

Також у результаті було отримано два модулі для інтеграції у платформи для електронного навчання.

Апробація результатів кваліфікаційної роботи магістра

Результати роботи було представлено на науково-технічних конференціях студентів, магістрантів, аспірантів, молодих вчених і опубліковані в збірнику наукових праць студентів, аспірантів і молодих вчених «Молода наука — 2020» [30], а також в збірнику XXV науково-технічної конференції студентів, магістрантів, аспірантів, молодих вчених та викладачів 2020 року [31].

Глосарій

Виділення ознак — це різновид абстрагування, процес зниження розмірності, в якому вихідний набір вихідних змінних скорочується до більш керованих груп (ознак) для подальшої обробки, залишаючись при цьому достатнім набором для точного і повного опису вихідного набору даних. Виділення ознак використовується в машинному навчанні, розпізнаванні образів і при обробці зображень. Виділення ознак починає з вихідного набору даних, виводить вторинні значення (ознаки), для яких передбачається, що вони повинні бути інформативними і не бути надмірними, що сприяє подальшому процесу навчання машини і узагальнення кроків, а в деяких випадках веде і кращої і легшої інтерпретаціх даних людиною.

Відбір ознак (відбір змінних, відбір атрибутів або генералізація) — це різновид абстрагування, процес відбору підмножини значущих ознак (змінних залежних і незалежних) для використання в побудові моделі. Техніки відбору ознак використовуються для: спрощення моделей для того, щоб зробити їх простіше для інтерпретації дослідниками / користувачами; коротший час тренування; щоб уникнути прокляття розмірності; покращене узагальнення шляхом скорочення перенавчання.

Дистрибутивна семантика — це область лінгвістики, яка займається обчисленням ступеня семантичної близькості між лінгвістичними одиницями

на підставі їх розподілу (дистрибуції) у великих масивах лінгвістичних даних (текстових корпусах).

Електронне навчання (англ. E-learning, скорочення від англ. Electronic Learning) — система навчання, за допомогою інформаційних, електронних технологій. Часто тлумачиться, як синонім таких понять: дистанційне навчання, навчання з застосуванням комп'ютерів, мережеве навчання, віртуальне навчання, мультимедійне навчання, мобільне навчання.

Зниження розмірності — це перетворення даних, що складається в зменшенні числа змінних шляхом отримання головних змінних. Перетворення може бути розділене на відбір ознак та виділення ознак.

Інтервал довіри біноміальної пропорції — це довірчий інтервал для ймовірності успіху, обчислений за результатами серії експериментів «успіх-невдача» (випробування Бернуллі). Іншими словами, довірчий інтервал біноміальної пропорції — це інтервальна оцінка ймовірності успіху p, коли відомо лише кількість експериментів n та кількість успіхів nS.

Існує кілька формул для біноміального довірчого інтервалу, але всі вони опираються на припущення про біноміальний розподіл. Взагалі, біноміальний розподіл застосовується, коли експеримент повторюється фіксовану кількість разів, кожне випробування експерименту має два можливі результати (успіх і невдача), ймовірність успіху однакова для кожного випробування, а випробування статистично незалежні.

Кластерний аналіз (англ. Cluster analysis) — багатовимірна статистична процедура, яка виконує збір даних, що містять інформацію про вибірку об'єктів, і потім впорядковує об'єкти в порівняно однорідні групи. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу задач навчання без учителя.

Коллаборативна фільтрація, сумісна фільтрація — це один з методів побудови прогнозів (рекомендацій) у рекомендованих системах, використовуючи відомі переважні оцінки групи користувачів, що прогнозують невідомі припущення інших користувачів. Основне припущення даного методу: ті, хто

одинаково оцінив якісь або будь-які предмети в минулому, схильні дати подібні оцінки інших предметів і в майбутньому. Наприклад, з підтримкою коллаборативної фільтрації музичний додаток здатний прогнозувати, яка музика сподобається користувачу, маючи неповний список його симпатій та антипатій. Прогнози складаються індивідуально для кожного користувача, хоча використовувана інформація зібрана від багатьох учасників. Тим самим коллаборативна фільтрація відрізняється від більш простого підходу, що дає усереднену оцінку для кожного об'єкта інтересу, наприклад, що базується на кількості поданих за нього голосів.

Корпус текстів — підібрана і оброблена за певними правилами сукупність текстів, які використовуються в якості бази для дослідження мови. Вони використовуються для статистичного аналізу і перевірки статистичних гіпотез, підтвердження лінгвістичних правил в даній мові. Корпус текстів є предметом дослідження корпусної лінгвістики.

Навчання без вчителя (самонавчання, спонтанне навчання, англ. Unsupervised learning) — один із способів машинного навчання, при якому випробувана система спонтанно навчається виконувати поставлене завдання без втручання з боку експериментатора. З точки зору кібернетики, це є одним з видів кібернетичного експерименту. Як правило, це придатне тільки для завдань, в яких відомі описи безлічі об'єктів (навчальної вибірки), і потрібно виявити внутрішні взаємозв'язки, залежності, закономірності, що існують між об'єктами. Навчання без вчителя часто протиставляється навчанню з учителем, коли для кожного навчального об'єкта примусово задається «правильна відповідь», і потрібно знайти залежність між стимулами і реакціями системи.

Навчання з учителем (англ. Supervised learning) — один із способів машинного навчання, в ході якого випробувана система примусово навчається за допомогою прикладів «стимул-реакція». З точки зору кібернетики, є одним з видів кібернетичного експерименту. Між входами і еталонними виходами (стимул-реакція) може існувати деяка залежність, але вона невідома. Відома

тільки кінцева сукупність прецедентів-пар «стимул-реакція», звана навчальної вибіркою. На основі цих даних потрібно відновити залежність (побудувати модель відносин стимул-реакція, придатних для прогнозування), тобто побудувати алгоритм, здатний для будь-якого об'єкта видати досить точну відповідь.

Прокляття розмірності (ПР) — термін, який використовується щодо ряду властивостей багатовимірних просторів і комбінаторних задач. В першу чергу це стосується експоненціального зростання необхідних експериментальних даних в залежності від розмірності простору при вирішенні задач ймовірносно-статистичного розпізнавання образів, машинного навчання, класифікації.

Рейтинг кліків або Клікабельність (англ. Click-through rate, CTR) — відношення числа кліків на оголошення до числа його показів. Вимірюється у відсотках. Використовується для виміру ефективності онлайн-реклами для певного веб-сайту чи email-кампаній. Формула розрахунку: CTR = (число кліків / число показів) х 100 %. Наприклад, якщо рекламне оголошення було показано 100 раз, а клікнули по ньому лише один раз, то показник CTR дорівнюватиме 1 %. Слід зазначити, що показник більший за 2 % вважається успішним, і загалом рекламну кампанію з таким показником можна вважати успішною.

Рекомендаційні системи — програми, які намагаються передбачити, які об'єкти (фільми, музика, книги, новини, веб-сайти) будуть цікаві користувачеві, маючи певну інформацію про його профілі. В процесі роботи рекомендаційні системи збирають дані про користувачів, використовуючи поєднання явних і неявних методів. Рекомендаційні системи порівнюють однотипні дані від різних людей і обчислюють список рекомендацій для конкретного користувача. Для обчислення рекомендацій використовується граф інтересів. Рекомендаційні системи — зручна альтернатива пошуковим алгоритмам, так як вони можуть виявитиоб’єкти, які не будуть знайдені через звичайний пошук даних.

Факторизація або розкладання на множники — це декомпозиція об'єкта (наприклад, числа, многочлена або матриці) у добуток інших об'єктів, або множників, які після перемноження дадуть вихідний об'єкт. Наприклад, число 15 розкладається на прості множники як 3 × 5, многочлен x2 − 4 розкладається на множники як (x − 2)(x + 2). У всіх випадках, отримано добуток простіших об'єктів. Метою факторизації є зазвичай звести щось до «базових будівельних блоків», наприклад, цілі числа до простих чисел чи многочлени до незвідних многочленів. Факторизація цілих чисел забезпечується основною теоремою арифметики і факторизація многочленів — основною теоремою алгебри. Теорема Вієта пов'язує коефіцієнти многочлена з його коренями. Згідно з основною теоремою арифметики, кожне додатне ціле число більше одиниці має єдиний розклад на прості множники. За допомогою алгоритмів факторизації цілих чисел, можна розкласти будь-яке ціле число на прості множники за допомогою повторного застосування цих алгоритмів. Проте для дуже великих чисел невідомо ефективних алгоритмів.

Штучний інтелект (англ. Artificial intelligence, AI) — розділ комп'ютерної лінгвістики та інформатики, що опікується формалізацією проблем та завдань, які подібні до дій, що виконує людина.

Штучний нейрон (Математичний нейрон Маккалоха — Піттса, Формальний нейрон) — вузол штучної нейронної мережі, що є спрощеною моделлю природного нейрона. Математично, штучний нейрон зазвичай представляють як деяку нелінійну функцію від єдиного аргументу — лінійної комбінації всіх вхідних сигналів. Цю функцію називають функцією активації або функцією спрацьовування, передавальною функцією. Отриманий результат посилається на єдиний вихід. Такі штучні нейрони об'єднують в мережі — з'єднують виходи одних нейронів з входами інших.

Штучні нейронні мережі (скорочено ШНМ) — це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального про-

грамування під задачу. Наприклад, у розпізнаванні зображень вони можуть навчатися ідентифікувати зображення, які містять котів, аналізуючи приклади зображень, мічені як «кіт» і «не кіт», і використовуючи результати для ідентифікування котів в інших зображеннях. Вони роблять це без жодного апріорного знання про котів, наприклад, що вони мають хутро, хвости, вуса та котоподібні писки. Натомість, вони розвивають свій власний набір доречних характеристик з навчального матеріалу, який вони оброблюють. ШНМ ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами (аналогічно до біологічних нейронів у головному мозку тварин). Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, й потім сигналізувати штучним нейронам, приєднаним до нього.

e-Learning (електронне навчання, електронна освіта) — навчання за допомогою Інтернет і мультимедіа.

ML.NET — це бібліотека машинного навчання для мов програмування C# і F#. Вона також підтримує моделі Python при використанні разом з NimbusML. Попередній випуск ML.NET включав перетворення для функціональної інженерії, наприклад створення n-gram, і навчання, яке керувало двійковою класифікацією, багатокласовою класифікацією та регресійними завданнями. З тих пір були додані додаткові завдання штучного інтелекту, такі як системи виявлення аномалій та рекомендацій, а інші підходи, такі як глибоке навчання, будуть включені в майбутні версії.

Word embedding — загальна назва для різних підходів до моделювання мови та навчання уявлень в обробці природної мови, спрямованих на зіставлення словами (і, можливо, фразам) з деякого словника векторів з Rn для n, значно меншої кількості слів в словнику. Теоретичною базою для векторних уявлень є дистрибутивна семантика.

РОЗДІЛ 1 АНАЛІЗ ЗАСОБІВ ЕЛЕКТРОННОГО НАВЧАННЯ, МЕТОДІВ СТАТИСТИКИ ТА РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Електронне навчання

Електронне навчання — це система навчання за допомогою інформаційних та електронних технологій.

Існує визначення, яке дали фахівці ЮНЕСКО: «e-Learning — навчання за допомогою інтернету і мультимедіа» [13].

До електронного навчання відноситься:

1) самостійна робота з електронними матеріалами, з використанням персонального комп'ютера, КПК, мобільного телефону, телевізора і інших електронних матеріалів;
2) отримання консультацій, порад, оцінок у віддаленого (територіально) експерта (викладача), можливість дистанційної взаємодії;
3) створення розподіленої спільноти користувачів (соціальних мереж) для проведення загальної віртуальної навчальної діяльності;
4) своєчасна цілодобова доставка електронних навчальних матеріалів; стандарти і специфікації на електронні навчальні матеріали та технології, дистанційні засоби навчання;
5) формування та підвищення інформаційної культури у всіх керівників підприємств і підрозділів групи і оволодіння ними сучасними інформаційними технологіями, підвищення ефективності своєї звичайної діяльності;
6) освоєння і популяризація інноваційних педагогічних технологій, передача їх викладачам;
7) можливість розвивати навчальні веб-ресурси;
8) можливість в будь-який час і в будь-якому місці отримати сучасні знання, що знаходяться в будь-якій доступній точці світу;

9) доступність вищої освіти особам з особливостями психофізичного розвитку.

До електронної освіти відносяться електронні підручники, освітні послуги та технології.

Сучасні студенти та школярі — в більшості Мережеве покоління, для кого електронний спосіб отримання інформації (у цьому випадку саме навчальної інформації) є нормальною складовою життя. У цілому Інтернет технології в освіті вітаються студентами, знання, вміння, навички знадобляться у самовдосконаленні та кар'єрному зростанні. Інформаційні технології стали їх сучасним робочим інструментом.

Стрімкість сучасного світу вимагає розвитку, застосування і оптимізації швидких и дешевих засобів генерації и передачі знань. Електронне навчання є одним з інструментів, які дозволяють вирішувати цю проблему сучасності.

Широкий спектр методів дистанційного навчання дозволяє обирати метод з урахуванням індивідуальних вимог та уподобань студента. А також електронне навчання не виключає спілкування з викладачем особисто:

Було проведено дослідження на тему «Що чекає e-learning надалі?» [11]. На думку респондентів, для успішного застосування e-learning в корпоративному секторі найбільшу увагу необхідно приділяти таким напрямкам:

1) виробництво і доставка електронного навчального контенту (25%);
2) розвиток корпоративної стратегії e-learning (17%);
3) впровадження і використання технологій та інструментів (17%);
4) управління й оцінка ефективності ініціатив у сфері електронного навчання (14%);
5) врахування потреб і побажань учнів (12%);
6) збільшення технологічних потужностей для підтримки e-learning (11%);
7) інші (4%).

1.1.1 Підходи та особливості електронного навчання

До електронного навчання відносяться електронні підручники, освітні послуги та технології. Фактично електронне навчання почалося з використанням комп'ютерів в освіті. Спочатку навчання з використанням комп'ютерів чергувалось зі звичайними, класичними практичними заняттями. Але електронне навчання і зараз не виключає спілкування з викладачем один на один.

Найвідоміші «системи» в даній сфері — це системи дистанційного навчання, або скорочено СДН. Термін став настільки широко вживаним, що часто використовується як повний синонім, тобто «Впровадити e-learning» прирівнюється до «придбати і налаштувати СДН».

У становленні електронного навчання можна виділити три етапи:

1) курси на носіях CD-ROM;
2) дистанційне навчання у живих викладачів;
3) електронне навчання з використанням спеціальних інтерактивних програм, нерідко на спеціальних носіях (електронні підручники).

У 2010 році з'явилася ще одна форма навчання — масові відкриті онлайн-курси, які дозволяють одночасно навчати сотні тисяч студентів.

Концепція електронного навчання сучасного зразка розвинулася разом з комп’ютерними технологіями і включає в себе можливість практично з будь-якого місця завантажити додаткові матеріали, що підкріплюють отриману за допомогою електронних посібників теорію, передати виконане завдання, порадитися з викладачем. Головне, щоб всі ці функції підтримував носій електронних програм. Зараз розвиток електронного навчання безпосередньо залежить від розвитку носіїв.

Електронна освіта має низку переваг, які варто взяти до уваги при аналізі даної форми навчання:

1. Свобода доступу — учень може займатися практично в будь-якому місці. Далеко не всі функції електронної освіти реалізуються через

інтернет. Дорослий учень може навчатися без відриву від основної роботи.

2. Зниження витрат на навчання — учень несе витрати на носій інформації, але не на методичну літературу. Крім того, економія зростає за рахунок зарплат, утримання навчальних закладів і так далі. Виробництво електронних навчальних матеріалів також не провокує вирубку лісів.
3. Гнучкість навчання — тривалість і послідовність вивчення матеріалів слухач обирає сам, повністю адаптуючи весь процес навчання під свої можливості і потреби.
4. Можливість розвиватись в ногу з часом — користувачі електронних курсів: і викладачі, і студенти розвивають свої навички і знання відповідно до новітніх сучасних технологій і стандартів. Електронні курси також дозволяють своєчасно і оперативно оновлювати навчальні матеріали.
5. Потенційно рівні можливості навчання — навчання стає незалежним від якості викладання в конкретному навчальному закладі.
6. Можливість визначати критерії оцінки знань — в електронному навчанні є можливість виставляти чіткі критерії, за якими оцінюються знання, отримані студентом в процесі навчання.

1.1.2 Види платформ електронної освіти

За даними SPD Load платформи онлайн-навчання розділені на дві основні групи: з синхронним та асинхронним процесом навчання [11]. Окрім цих двох основних типів, нам потрібно визначити інші підтипи платформ (Рис. 1).

Рис. *1 Приклади найпопулярніших платформ електронного навчання*

Learning Destination Sites

Найближчий по сенсу переклад — навчальні сайти цільового призначення. Ці веб-сайти для електронного навчання схожі на інтернет-магазин. Різні користувачі (викладачі, компанії, тощо) можуть публікувати свої курси та навчальні матеріали. Вони також можуть створити курс на веб-сайті за допомогою системи управління курсами.

Користувачі можуть вибрати необхідний курс і отримати до нього доступ. Ці курси можуть мати посилання для переадресації або це може бути комплексний курс, доступний на веб-сайті.

Даний вид навчальних веб-сайтів є найбільш популярним серед студентів, бо є більш доступним та містить переважно короткі курси, які задовольняють поточні проблеми та питання студента.

Приклади платформ такого виду:

- Coursera;
- Udacity;
- Udemy;
- edX.

Традиційні системи управління навчанням (Traditional Learning Management Systems — скорочено TLMS)

Це веб-сайт, який пропонує інструменти для творців курсів та навчальних матеріалів. Окрім цього, користувачі можуть виконувати різні інші завдання. Це включає створення звітів, відстеження успіху студента та створення профілів.

Компанії або установи частіше обирають онлайн-платформи навчання на базі TLMS для внутрішнього навчання. Не так часто це використовується для розваги зовнішньої аудиторії. Дані платформи зазвичай допомагають працівникам підвищувати кваліфікацію, освоювати різні види програмного забезпечення, тощо. Це може бути що завгодно, від оптимального управління щоденними завданнями до подання податкової декларації.

Приклади платформ даного виду**:**

- Abara;
- CD2 Learning;
- Easy LMS;
- Learn Dach.

Системи управління навчанням з відкритим кодом (Open Source-learning Management System)

З відкритим кодом означає, що ці платформи можуть вікористовуватися користувачами безкоштовно. Ви можете створити власний веб-сайт електронного навчання за допомогою такої системи. Перевага полягає в тому, що ви можете дозволити іншим компаніям створювати свої курси та розміщувати їх на вашій платформі.

Приклади платформ такого виду:

- Moodle;
- Sakai.

Сучасні рішення для управління навчанням (Modern Learning Management Solutions)

Сучасна система LMS означає, що ви створюєте веб-сайт для електронного навчання, присвячений певним темам та педагогіці. Ці системи мають кращі можливості для таких цілей, ніж інші. У такі платформи часто

вбудовано багато інструментів для певних сфер діяльності. Наприклад у систему LMS може бути вбудований компілятор, аналізатор якості коду, графічний емулятор, тощо. Вбудовані інструменти зазвичай залежать від предметної області та часто використовуваних у даній області програмних засобів.

Приклади**:**

- Lessonly;
- People Fluent.

Екосистеми управління навчанням (Learning Management Ecosystems)

Хорошим прикладом такої системи є платформа NeXus, якою керує університет Нотр-Дам.

Заснована система LMS із відкритим кодом NeXus інтегрує всі інші функції. Це включає створення звітів, перевірку, магазини e-commerce та інші.

Створення цієї всеосяжної платформи також є складним. Такий веб-сайт для електронного навчання потребуватиме потужного сервера та постійного моніторингу. Даний вид платформ потребує більшої лояльності клієнта.

Приклади:

- Abara;
- NeXus.

Індивідуальна навчальна платформа (Tailor-Made-learning Platform)

У даний вид платформ зазвичай вбудована певна унікальна комбінація функцій для вузької цільової аудиторії або сфери освіти.

Цей веб-сайт електронного навчання може включати педагогічне навчання та системи звітування. Крім того, ви також можете включити такі аспекти, як LMS, створення курсів та багато іншого.

Окрім найбільш активних функцій у таких платформах велика увага має приділятися аналізу поведінки та досвіду користувача. Відповідно до цього аналізу буде адаптуватися та оптимізуватися дизайн сайту.

Приклади:

- HBX;
- Moodle.

1.1.3 Ринок електронної освіти

Світова індустрія електронного навчання ще у 2000 році становила 48 млрд доларів [10]. Електронне навчання виникло завдяки розвитку інтернету і мультимедіа, ключовими моментами є консалтинг, контент, технології, сервіси та підтримка.

Стрімкість сучасного світу вимагає застосування найбільш швидких і дешевих способів процесів генерації і передачі знань. Електронне навчання як інструмент відповідає цим цілям.

За даними Babson Survey Research Group в 2012 році в США в онлайн-навчання у вищих навчальних закладах було залучено 6,7 мільйона студентів [12]. Онлайн-освіта швидко розвивається і в провідних дослідницьких інститутах навіть розроблені докторські програми, представлені онлайн [12].

Багато вищих навчальних закладів, інститутів на комерційній основі пропонують зараз навчання в онлайн-класах. Кількість таких навчальних закладів зростає в міру розвитку і здешевлення технологій електронного навчання.

Також потрібно враховувати, що для роботи зі студентами в режимі онлайн навчальним закладам потрібен кваліфікований персонал, який володіє навичками роботи з комп'ютером та інтернет-технологіями. У зв'язку з режимом самоізоляції під час пандемії Covid-19 усім ВНЗ України було рекомендовано організувати навчання студентів із застосуванням дистанційних технологій [14].

Основні проблеми електронної освіти

Електронне навчання відносно нове явище, тому воно стикається з низкою проблем у розвитку, не тільки в технічній частині, а й зі сторони законодавчої бази, стандартизації та ін. До основних проблем розвитку електронного навчання можна віднести:

1) відсутність критеріїв і загальних стандартів якості електронних навчальних матеріалів, мало розкритий потенціал можливості передачі інформації через Інтернет: в більшості використовуються такі форми як текст і проста графіка;
2) правові проблеми, пов'язані як з нормативно-правовим забезпеченням електронного навчання, так і з питаннями щодо захисту авторських прав;
3) питання фінансування (витрати на розробку, зберігання даних, створення веб-ресурсів, їх підтримку їх діяльності та оновлення);
4) кадрові проблеми (брак кваліфікованого персоналу, складність його навчання, так як одночасно потрібно охопити і предметну область, і застосування IT-технологій, і художнього оформлення матеріалів).

1.2 Сукупність даних та вибірки

Вибірка або вибіркова сукупність — частина генеральної сукупності елементів, яка охоплюється експериментом (спостереженням, опитуванням). В контексті машинного навчання та оптимізації програмного забезпечення доречнішим буде наступне визначення:

Вибірка (sample, set) — кінцевий набір прецедентів (об'єктів, випадків, подій, експериментів, зразків, і т.п.), деяким способом обраних з безлічі всіх можливих прецедентів, званого генеральною сукупністю.

Якщо дослідник не має можливості управляти вибором прецедентів, то зазвичай передбачається, що вибір прецедентів випадковий. Якщо ж вибором

прецедентів можна управляти, то виникають завдання оптимального формування вибірки.

По кожному прецеденту збираються (вимірюються) деякі дані (data), що утворюють опис прецеденту. Сукупність описів всіх прецедентів вибірки є вхідною інформацією для статистичного аналізу даних, інтелектуального аналізу даних, машинного навчання.

Терміни вибірка (sample, set) і дані (data) взаємозамінні; іноді вони вживаються разом як один термін вибірка даних (data set). Тому аналіз даних можна розуміти також як аналіз кінцевих вибірок.

Основні цілі аналізу даних:

1) перевірка гіпотез щодо наявної вибірки даних;
2) емпірична індукція — виявлення загальних закономірностей, властивих всій генеральної сукупності, за наявними даними;
3) прогнозування — формування статистично обґрунтованих прогнозів щодо нових даних, які ще не спостерігалися.

Навчальна вибірка (training sample) — вибірка, з якої відбувається налаштування (оптимізація параметрів) моделі (наприклад нейронної мережі).

Якщо модель залежності побудована за навчальною вибіркою Xm, то оцінка якості цієї моделі, зроблена по тій же вибірці Xm виявляється, як правило, оптимістично зміщеною. Це небажане явище називають перенавчанням. На практиці воно зустрічається дуже часто. Хорошу емпіричну оцінку якості побудованої моделі дає її перевірка на незалежних даних, які не використовувалися для навчання.

Тестова (або контрольна) вибірка (test sample) — вибірка, за якої оцінюється якість побудованої моделі. Якщо навчальна і тестова вибірки незалежні, то оцінка, зроблена за тестовою вибіркою, є незміщеною.

Оцінку якості, зроблену за тестовою вибіркою, можна застосувати для вибору найкращої моделі. Однак тоді вона знову опиниться оптимістично зміщеною. Для отримання незміщеної оцінки оберненої моделі доводиться виділяти третю вибірку.

Перевірочна вибірка (validation sample) — вибірка, за якою здійснюється вибір найкращої моделі з безлічі моделей, побудованих за навчальною вибіркою.

Характеристики вибірки:

1) якісна характеристика вибірки - що саме ми вибираємо і які способи побудови вибірки ми для цього використовуємо;
2) кількісна характеристика вибірки - скільки випадків вибираємо, іншими словами обсяг вибірки.

Необхідність вибірки:

1) об'єкт дослідження дуже великий (наприклад, споживачі продукції глобальної компанії — величезна кількість територіально розкиданих компаній);
2) існує необхідність в зборі вторинної інформації.

Деякі навчальні вибірки можуть містити тільки кілька сотень спостережень, інші можуть включати в себе мільйони точок даних. У нас є доступ до навчальних множин з мільйонами, або навіть мільярдами прикладів.

Проте, алгоритми машинного навчання потребують очищення даних від шуму. Без цього етапу більшість алгоритмів значно погіршуються у своїй ефективності. Тому у наступних пунктах будуть проаналізовані основні методи попередньої обробки вибірки, її нормалізації, дискретизації та очищення даних від шуму.

Багато з навчальних множин готуються вручну, або ж з використанням напівавтоматичних процесів. Створення великих колекцій даних для навчання по прецедентах може бути досить витратним процесом в деяких областях.

1.2.1 Попередня обробка вибірки

У завданнях машинного навчання якість моделей дуже сильно залежить від даних. Але самі дані в реальних задачах рідко бувають ідеальними. Як правило, самих даних небагато, кількість доступних для аналізу парамет-

рів обмежена, в даних шуми і пропуски. Але вирішувати завдання якось потрібно.

Статистика — потужний інструмент в машинному навчанні. Вона дозволяє отримати інформацію з даних, дізнатися їх структуру і на основі отриманої інформації провести подальший аналіз.

Наприклад, XGBoost може дати поліпшення якості моделі близько 5% в порівнянні з випадковим набором, нейронна мережа до 3% в порівнянні з XGBoost. Оптимізації, регуляризація і підбір гіперпараметрів може ще додати 1-5%.

Але просто додавши інформаційні ознаки, витягнуті з тих же даних, які вже є, можна відразу отримати до 15% приросту якості моделі.

1.2.2 Зниження розмірності

Метод відбору ознак намагається знайти підмножину вихідних змінних (які називаються ознаками або атрибутами).

Є три стратегії:

1) стратегія фільтра (наприклад, накопичення ознак);
2) стратегія обгортання (наприклад, пошук згідно точності);
3) стратегія вкладення (вибираються ознаки для додавання або видалення в міру побудови моделі, заснованої на помилках прогнозування).

У деяких випадках аналіз даних, такий як регресія або класифікація, може бути здійснений в результуючому просторі більш точно, ніж у вхідному просторі.

Метод головних компонент (МГК)

Основна лінійна техніка для зниження розмірності, метод головних компонент, здійснює лінійне відображення даних в простір меншої розмірності таким чином, що дисперсія даних максимізується. На практиці будується

матриця ко-варіації (а іноді кореляції) даних і обчислюються власні вектори цієї матриці.

Частина перша: ініціалізація алгоритму. На вхід алгоритму подається масив даних, а так само розмірність простору, до якої необхідно зменшити дані.

1) Обчислюється ко-варіаційна матриця (вона симетрична, це важливо у даному алгоритмі).
2) Обчислюються власні вектори матриці.
3) Вибираються перші N власних векторів, де N — це розмірність, до якої потрібно зменшити розмірність простору ознак; вибрані вектори можна записати вертикально і зібрати в матрицю $\mathbf{U}_{\mathbf{reduced}}$.

Частина друга: зменшення розмірності вхідного вектора. На вхід подається вектор початкової розмірності, використаний на першому кроці; на виході вектор меншої розмірності (або проекція вхідного вектора на ортонормований базис утворений вибіркою з власних векторів).

Помноживши скалярно вхідний вектор на всі вектори з вибірки власних векторів, виходить зменшений вектор (Рис. 2)

$$\vec{x}_{reduced} = \vec{x}^T U_{reduced}$$

Рис. *2 Зменшений вектор*

Частина третя: відновлення розмірності вектору (з втратою інформації). На вхід подається вектор розмірності, до якої ми зменшували вектори; на виході вектор вихідної розмірності.

Якщо транспонувати матрицю $\mathbf{U}_{\mathbf{reduced}}$ то в ній виявиться стільки ж рядків як і розмірність вхідного вектора на попередньому кроці, а шуканий вектор відновлюється за формулою (Рис. 3).

$$\vec{x} = \vec{x}^T_{reduced} U^T_{reduced}$$

Рис. *3 Відновлення вектору*

Діаграма важливості ознак

Говорячи про інтерпретацію алгоритмів машинного навчання, зазвичай обговорюють лінійні регресії (що дозволяють проаналізувати значимість ознак за допомогою p-значень) і дерева рішень (буквально показують важливість ознак у формі дерева, а заодно і їх ієрархію). З іншого боку, в таких алгоритмах, як Random Forest, LightGBM і XG Boost, часто використовується діаграма значущості ознак, тобто будується діаграма змінних і «кількості їх важливості». Це особливо корисно, коли потрібно надати структуроване обґрунтування важливості ознак з точки зору їх впливу на бізнес.

Невід'ємне матричне розкладання (NMP)

Невід'ємне матричне розкладання розкладає невід'ємну матрицю на добуток двох невід'ємних матриць. Часто використовується у астрономії.

Лінійний дискримінантний аналіз (ЛДА)

Лінійний дискримінантний аналіз (ЛДА) є узагальненням лінійного дискримінанту Фішера, методу, застосовуваного в статистиці, розпізнаванні образів і машинному навчанні для пошуку лінійної комбінації ознак, які описують або розділяють два і більше класи об'єктів або подій.

Автокодування

Може бути використано для вивчення функцій нелінійного зниження розмірності і кодування разом зі зворотною функцією з кодованого до вихідного представлення.

1.2.3 Оверсемплінг і андерсемплінг

Оверсемплінг і андерсемплінг використовуються в задачах класифікації. Часом набір даних для класифікації сильно зрушений в одну сторону. Наприклад, для класу 1 може бути 2000 прикладів, а для класу 2 — всього

200. Це негативно вплине на багато методів машинного навчання, які використовуються для моделювання даних і складання прогнозів. Овер- і андерсемплінг потрібні якраз для таких випадків (Рис. 4).

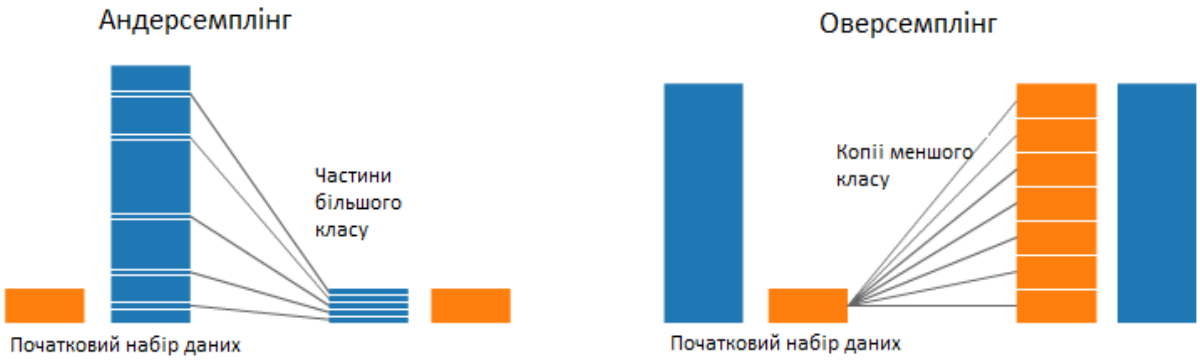

Рис. *4 Приклади овер- і андерсемплінгу*

На обох сторонах цієї картинки синій клас містить набагато більше даних, ніж помаранчевий. У такій ситуації можна зробити один з двох кроків для препроцессінгу даних перед їх використанням для навчання моделей.

Андерсемплінг означає, що використовується тільки частина даних більшого класу. Це необхідно для підтримки розподілу ймовірності класу.

Оверсемплінг означає, що створюються копії меншого класу, щоб зрівняти його розмір з великим класом. Копії робляться таким чином, щоб підтримувати розподіл меншого класу.

1.3 Загальні відомості про концепцію вкладень

Word Embedding — це зіставлення довільній суті (наприклад, вузлу в графі або шматочку картинки) деякого вектору (Рис. 5) [1].

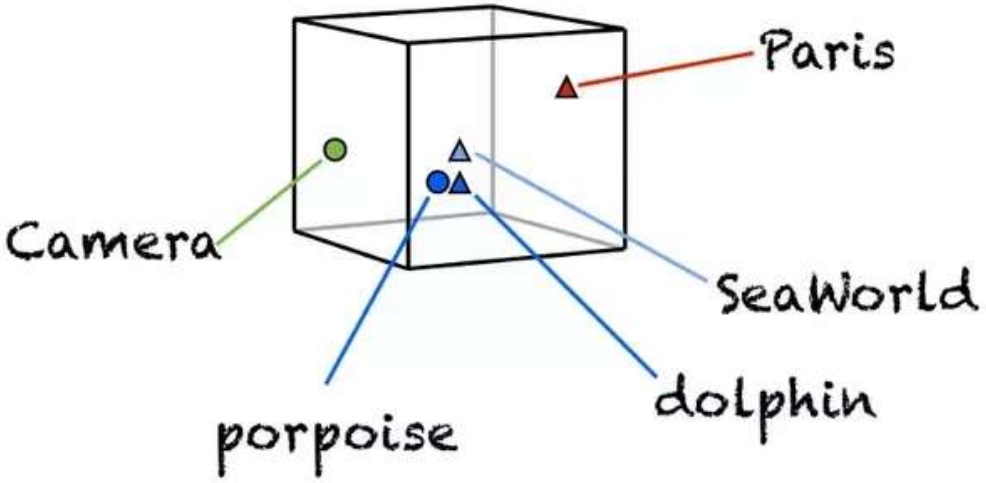

Рис. 5 *Word Embedding. Зіставлення слова вектору*

Далі розглянемо декілька методів зіставлення слова і вектору, їх особливості та сфери застосування.

One-hot encoding (OHE).

Який найпростіший спосіб отримати вектор з слова? Здається, що природно буде взяти вектор довжини нашого словника і поставити тільки одну одиницю в позиції, що відповідає номеру слова в словнику.

One-hot encoding не має властивості семантичної близькості а тому не використовується у сучасних системах. Даний алгоритм як правило реалізовується виключно з метою освоєння концепції вкладень в процесі навчання.

Bag of words (BoW).

Значення одного слова нам може бути і не так важливо, тому що мова (і усна, і письмова) складається з наборів слів, які ми називаємо текстами. Так що якщо ми захочемо якось уявити тексти, то ми візьмемо OHE-вектор кожного слова в тексті і складемо разом. Тобто на виході отримаємо просто підрахунок кількості різних слів у тексті в одному векторі (Рис.6). Такий підхід називається "мішок слів" (bag of words), тому що ми втрачаємо всю інформацію про взаємне розташування слів у тексті [1]. Але незважаючи на втрату цієї інформації так тексти вже можна порівнювати.

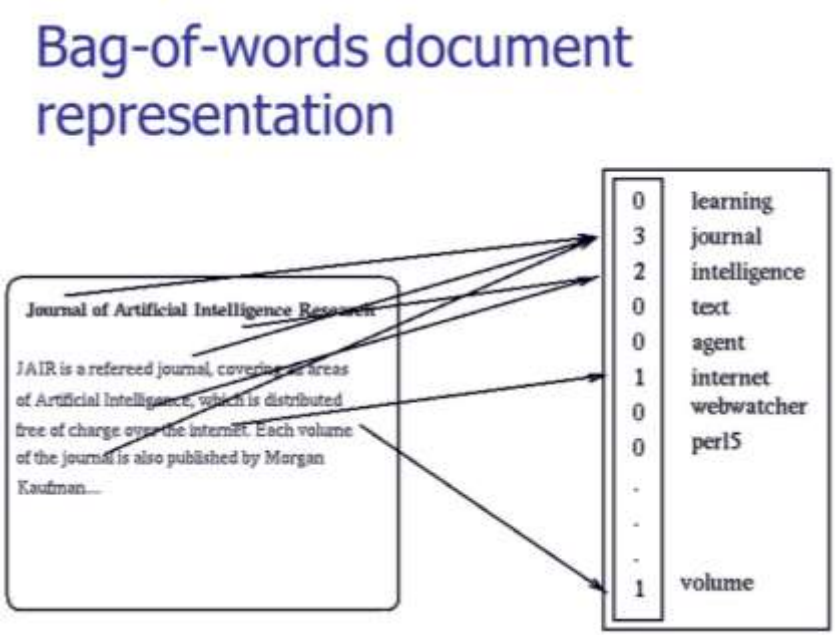

Рис. *6 Bag of words (BoW) text representation*

Матриця term-document

Ми можемо піти далі і представити наш корпус (набір текстів) у вигляді матриці "слово-документ" (term-document). Варто зазначити, що в області інформаційного пошуку (information retrieval) ця матриця має назву "зворотного індексу" (inverted index), в тому сенсі, що звичайний / прямий індекс виглядає як "документ-слово" і дуже незручний для швидкого пошуку [2].

Формула TF-IDF

Ця абревіатура означає "term frequency — inverse document frequency" (Рис. 7).

$$TF - IDF(w, d, C) = \frac{count(w, d)}{count(d)} * log(\frac{\sum_{d' \in C} 1(w, d')}{|C|})$$

Рис. *7 Формула TF-IDF*

TF — це частота слова в тексті. А IDF — це логарифм зворотної частоти поширеності слова в корпусі. Поширеністю називається відношення числа текстів, в яких зустрілося шукане слово, до загальної кількості текстів в корпусі. За допомогою TF-IDF тексти також можна порівнювати, і робити це можна впевненіше, ніж при використанні звичайних частот.

Основна проблема базових концепцій Word Embeddings

Кількість слів в англійській мові при першому наближенні становить мільйон — матриця спільних тільки пар слів буде 10 ^ 6 x 10 ^ 6. Така матриця навіть зараз є великою для пам'яті комп'ютерів, а, скажімо, 10 років тому робота з таким об’ємом була неможливою [3].

Були розроблені способи, що спрощують або розпаралелюють обробку таких матриць, але вони також не вирішували проблему повністю.

Word2Vec

У 2013 році чеський аспірант Томаш Міколов запропонував свій підхід до word embedding, який він назвав **word2vec** [3].

Його підхід заснований на іншій важливій гіпотезі, яку в науці прийнято називати гіпотезою локальності — "слова, які зустрічаються в однакових оточеннях, мають близькі значення" [4]. Близькість в даному випадку розуміється дуже широко, як те, що поруч можуть стояти тільки слова, що поєднуються одне з одним. Наприклад, для нас звичне словосполучення "заводний будильник". А сказати "заводний апельсин" ми не можемо — ці слова не поєднуються.

Ґрунтуючись на цій гіпотезі Томаш Міколов запропонував новий підхід, який не страждав від великих обсягів інформації, а навпаки вигравав.

Word Embeddings у контексті оптимізаціх електронного навчання

У даній роботі концепція Word Embeddings буде розглядатися з метою вирішення наступних категорій задач:

1) Завдання класифікації. Наприклад, необхідно по послідовності відвіданих сторінок сайту (курсів) визначити користувача.
2) Завдання регресії. Наприклад, необхідно по тексту (змісту) курсу/статті визначити його рейтинг на платформі.
3) Передбачення. (Наприклад, передбачення рейтингу нового курсу. Або передбачення успішності студента).

1.4 Загальні відомості про рекомендаційні системи

Рекомендаційні системи — системи, які намагаються передбачити, які об'єкти (курси, фільми, книги, новини, статті, тощо) будуть цікаві користувачеві, маючи певну інформацію про нього [5].

Частини рекомендаційної системи**:**

4) предмет рекомендації (курс, стаття, товар, тощо);
5) мета рекомендації (покупка, інформування, навчання, створення контактів);
6) контекст рекомендації — що користувач в цей момент робить;
7) джерело рекомендації;
8) ступінь персоналізації;
9) принцип рекомендації;
10) формат;
11) алгоритм.

У даній роботі рекомендаційні системи будуть розглядатися з метою вирішення наступних задач:

1) отримання вибірки курсів для рекомендації студенту;
2) отримання вибірки студентів для розсилки певного типу рекомендацій;
3) визначення умовного рейтингу кожного курсу для відображання у списках.

1.4.1 Матриця вподобань

Це матриця, по одній з осей якої відкладені всі клієнти сервісу (Users), а по інший — об'єкти рекомендації (Items). На перетині деяких пар (user, item) дана матриця заповнена оцінками (Ratings) — це відомий нам показник зацікавленості користувача в даному товарі, виражений за заданою шкалою (наприклад від 1 до 5) (Рис. 8).

	Товар 1	Товар 2	Товар 3	Товар 4	Товар 5
Клиент 1		3		5	
Клиент 2	1		1	1	
Клиент 3	2			3	2
Клиент 4		4			5
Клиент 5	5		2	3	4

Рис. *8 Приклад матриці вподобань*

Користувачі зазвичай оцінюють лише невелику частину товарів, що є в каталозі, і завдання рекомендаційної системи — узагальнити цю інформацію і передбачити ставлення клієнта до інших товарів, про які нічого не відомо. Іншими словами потрібно заповнити всі незаповнені клітинки на зображенні вище.

Вподобання (оцінки) користувача можна отримати двома способами:

1) явно (explicit ratings) — користувач сам ставить рейтинг товару, залишає відгук, лайкать сторінку;
2) неявно (implicit ratings) — користувач явно своє ставлення не виражає, але можна зробити непрямий висновок з його дій: купив товар — значить він йому подобається, довго читав опис — значить цікаво і т.п.

Звичайно, явні переваги краще - користувач сам говорить про те, що йому сподобалося. Однак на практиці далеко не всі сайти надають можливість явно виражати свій інтерес, та й не всі користувачі мають бажання це робити. Найчастіше використовуються відразу обидва типи оцінок, які добре доповнюють один одного [6].

1.4.2 Неперсоналізовані рекомендації

Неперсоналізовані рекомендації найпростіші в реалізації. В них потенційний інтерес користувача визначається просто середнім рейтингом товару: «Усім подобається — значить сподобається і вам». За цим принципом працює більшість сервісів, коли користувач не авторизується в системі.

Середній рейтинг товару також може зображуватися різними способами. Це можуть бути зірочки поруч з товаром, кількість лайків, різниця позитивних і негативних голосів (як зазвичай роблять на форумах), частка високих оцінок або взагалі гістограма оцінок.

Гістограми — найбільш інформативний спосіб, але у них є один мінус — їх складно порівнювати між собою або сортувати, коли потрібно вивести товари списком.

1.4.3 Проблема холодного старту

Холодний старт — це типова ситуація, коли ще не накопичено достатню кількість даних для коректної роботи рекомендаційної системи (наприклад, коли товар новий або просто його дуже рідко купують) [7]. Якщо середній рейтинг пораховано за оцінками всього трьох користувачів, така оцінка явно не буде достовірною, і користувачі це розуміють. Часто в таких ситуаціях рейтинги штучно корегують.

Перший спосіб — показувати не середнє значення, а згладжені середнє (**Damped Mean)** [6]. Сенс такий: при малій кількості оцінок рейтинг більше тяжіє до безпечного «середнього» показнику, а як тільки набирається достатня кількість нових оцінок, «усереднюється» і коригування перестає діяти.

Другий підхід — розраховувати по кожному рейтингу інтервали достовірності (**Confidence Intervals)** [6]. Математично, чим більше оцінок, тим менше варіація середнього і, отже, більше впевненість в його правильності. А в якості рейтингу можна виводити, наприклад, нижню межу інтервалу (Low CI Bound). При цьому зрозуміло, що така система буде досить консервативною, з тенденцією до заниження оцінок новим товарам.

Оскільки оцінки обмежені певною шкалою (наприклад від 0 до 1), звичайний спосіб розрахунку інтервалу достовірності тут погано підходить: через хвости розподілу, що йдуть у нескінченність і через симетричність самого інтервалу. Є альтернативний і більш точний спосіб його порахувати — Wilson Confidence Interval (інтервал довіри біномальної пропорції) (Рис. 9). При цьому виходять несиметричні інтервали приблизно такого вигляду.

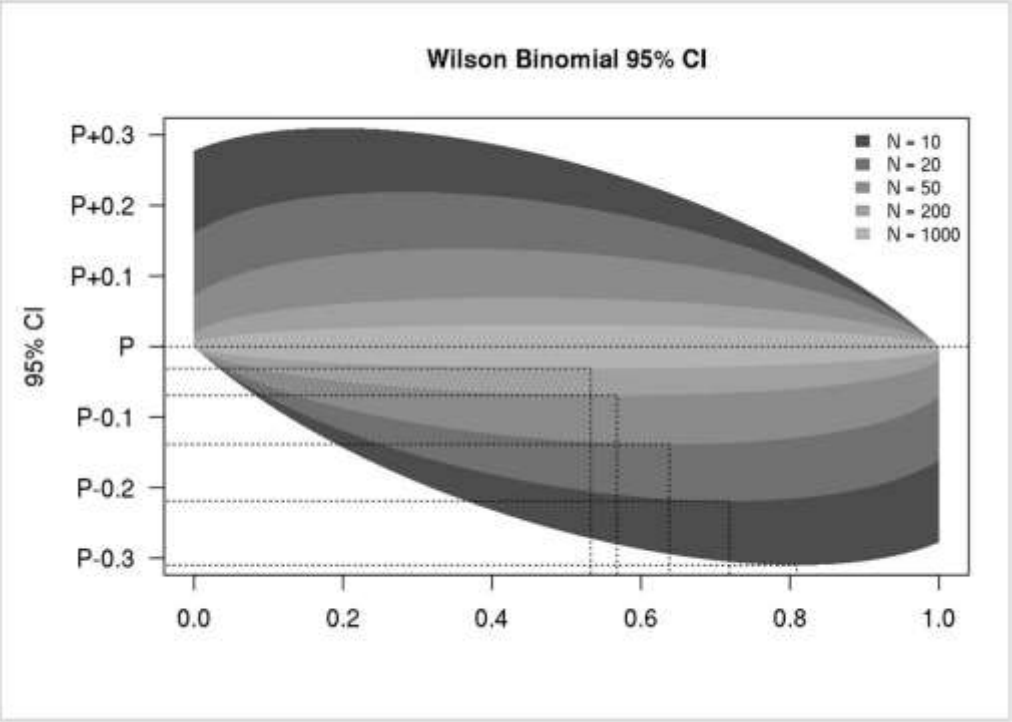

Рис. 9 *Інтервали довіри біномальної пропорції*

На Рис.9 по горизонталі відмічена оцінка середнього рівня значень рейтинга, по вертикалі — розрив навколо середніх значень. Кольором (від світлого до темного) виділені різні розміри виборок (чим вибірка більше, тим менший інтервалом достовірності).

Проблема холодного старту також актуальна і для неперсоналізованих рекомендацій.

Загальний спосіб тут — визначити те, що в даному моменті не може бути прораховано, різними евристиками (наприклад, замінити середнім рейтингом, або не використовувати оцінку поки не збереться достатня кількість даних).

1.4.4 Актуальність рекомендацій

У деяких випадках також важливо враховувати «свіжість» рекомендацій (Рис. 10, Рис. 11). Це особливо актуально для статей. Свіжі записи мають потрапляти в рекомендації частіше. Для цього використовуються коригуючі коефіцієнти (коефіцієнти демпфування). Зазначені нижче приклади були опубліковані ресурсами Reddit та Hacker News як приклади визначення актуальності новин на їх веб-сайтах [6].

$$Rank = \frac{(U-D-1)^{0.8} * P}{T^{1.8}}$$

Рис. 10 Приклад розрахунку рейтингу в журналі Hacker news

У прикладі Hacker news:

U = upvotes, D = downvotes, а P (Penalty) — додаткова коригування для імплементації інших бізнес-правил.

$$Rank = log_{10}(max(1, U - D)) - \frac{|U-D|T}{const}$$

Рис. 11 Розрахунок рейтингу в Reddit

У прикладі Reddit:

U = число голосів «за», D = число голосів «проти», T = час запису. Перший доданок оцінює «якість запису», а другий робить поправку на час.

1.4.5 Content-based рекомендації

Персональні рекомендації передбачають максимальне використання інформації про самого користувача, в першу чергу про його попередні покупки. Одним з перших з'явився підхід content-based filtering. В рамках даного підходу опис товару (content) зіставляється з інтересами користувача, отриманими з його попередніх оцінок. Чим більше товар цим інтересам відпові-

дає, тим вище оцінюється потенційна зацікавленість користувача. Очевидна вимога тут — у всіх товарів в каталозі має бути опис.

Предметом Content-based рекомендацій частіше є товари з неструктурованих описом: фільми, книги, статті. Такими ознаками можуть бути, наприклад, текстові описи, рецензії, склад акторів та інше. Однак ніщо не заважає використовувати і звичайні числові або категоріальні ознаки.

Неструктуровані ознаки описуються типовим для тексту способом — векторами в просторі слів (Vector-Space model). Кожен елемент такого вектора — ознака, потенційно що характеризує інтерес користувача. Аналогічно, продукт — вектор в тому ж просторі [6].

У міру взаємодії користувача з системою (скажімо, він купує фільми), векторні описи придбаних ним товарів об'єднуються (підсумовуються і нормалізуються) в єдиний вектор і, таким чином, формується вектор його інтересів. Далі досить знайти товар, опис якого найближче до вектору інтересів, тобто вирішити задачу пошуку n найближчих сусідів.

Дану задачу вирішує добре відоме в Text Mining перетворення TF-IDF.

В якості запобіжного близькості двох векторів найчастіше використовується косинусна відстань (Рис. 12).

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

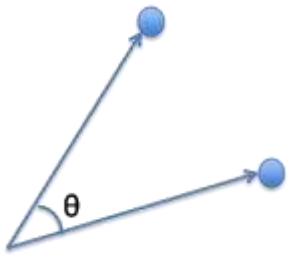

Рис. *12 Косинусна відстань*

Content-based фільтрація майже повністю повторює механізм query-document matching, який використовується в пошукових системах типу Яндекс і Google. Відмінність лише в формі пошукового запиту — тут це вектор, що описує інтереси користувача, а там — ключові слова запитуваного доку-

мента. Коли пошукові системи стали додавати персоналізацію, відмінність стерлася ще більше.

1.4.6 Коллаборативна фільтрація (User-based варіант)

Даний клас систем почав активно розвиватися в 90-і роки. В рамках підходу рекомендації генеруються на підставі інтересів інших схожих користувачів. Такі рекомендації є результатом «колаборації» безлічі користувачів. Звідси і назва методу.

Класична реалізація алгоритму заснована на принципі k найближчих сусідів. Для кожного користувача шукаємо k найбільш схожих на нього (в термінах переваг) і доповнюємо інформацію про користувача відомими даними по його сусідам (Рис. 13) [6].

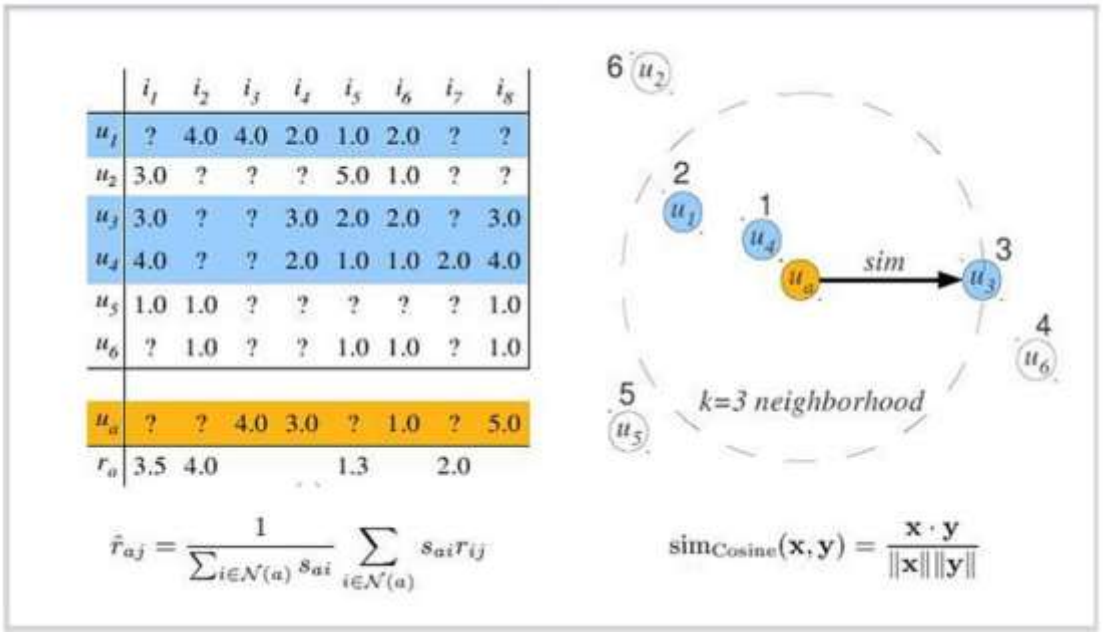

Рис. *13 Принцип роботи метода коллаборативної фільтрації (User-Based)*

У матриці переваг жовтим кольором виділено користувача, для якого ми хочемо визначити оцінки з нових товарів (знаки питання). Синім кольором виділені три його найближчих сусіда.

«Схожість» — в даному випадку синонім «кореляції» інтересів і може виражатися безліччю способів (крім кореляції Пірсона, є ще косинусна відстань, тобто відстань Жаккара, відстань Хеммінга та ін.).

Недолік класичної реалізації алгоритму.

Він не часто застосовується на практиці через квадратичну складність. Як будь-який метод найближчого сусіда, він вимагає розрахунку всіх попарних відстаней між користувачами (а користувачів можуть бути мільйони). Неважко порахувати, що складність розрахунку матриці відстаней буде

$$O(n^2 m)$$

де n — число користувачів, а m — число товарів. При мільйон користувачів для зберігання матриці відстаней в сирому вигляді, потрібно мінімум 4TB.

Дана проблема частково може бути вирішена покупкою високопродуктивного заліза.

Але краще ввести коригування в алгоритм:

1) оновлювати відстані не після кожної покупки, а за розкладом (наприклад, раз в день);
2) не перераховувати матрицю відстаней повністю, а оновлювати її інкрементами;
3) зробити вибір на користь ітеративних і наближених алгоритмів (наприклад ALS).

Для того щоб алгоритм був ефективний, важливо щоб виконувалося кілька припущень:

1) смаки людей не змінюються часом (або змінюються, але для всіх однаково);
2) якщо смаки людей збігаються, то вони збігаються у всьому.

В якості оптимальної кількості сусідів наводяться цифри в 30-50 сусідів для фільмів і 25-100 для довільних рекомендацій. Тут зрозуміло, що якщо візьмемо занадто багато сусідів, то отримаємо більше ймовірність випадко-

вого шуму. І навпаки, якщо візьмемо занадто мало, то отримаємо більш точні рекомендації, але меншу кількість товарів можна рекомендувати.

Важливий етап підготовки даних — нормалізація оцінок.

1.4.7 Коллаборативна фільтрація (Item-based варіант)

Підхід Item-based є природною альтернативою класичному підходу User-based і майже повністю його повторює, за винятком одного моменту — застосовується він до транспонованої матриці переваг. Тобто шукає близькі товари, а не користувачів.

Переваги Item-based перед User-based

Коли користувачів багато (майже завжди), пошук найближчого сусіда стає погано обчислювальним. Наприклад, для 1 млн користувачів потрібно розрахувати і зберігати ~ 500 млрд відстаней. Якщо відстань кодувати 8 байтами, це виходить 4TB для однієї тільки матриці відстаней. Якщо ми робимо Item-based, то складність обчислень знижується з $\boldsymbol{O(n^2m)}$ до $\boldsymbol{O(n^2N)}$, а матриця відстаней має розмірність вже не (1 млн на 1 млн) а, наприклад, (100 на 100) за кількістю товарів.

Оцінка близькості товарів набагато точніша, ніж оцінка близькості користувачів. Це прямий наслідок того, що користувачів зазвичай набагато більше, ніж товарів і отже стандартна помилка при розрахунку кореляції товарів там істотно менше.

У user-based варіанті описи користувачів, як правило, сильно розріджені (товарів багато, оцінок мало). З одного боку це допомагає оптимізувати розрахунок — ми перемножуємо тільки ті елементи, де є перетин. Але з іншого боку — скільки сусідів не бери, список товарів, які в підсумку можна порекомендувати, виходить дуже невеликим.

Уподобання користувача можуть змінюватися з часом, але опис товарів штука набагато більш стійка.

1.4.8 Інші підходи до реалізації рекомендаційних систем

Асоціативні правила (Association Rules)

Асоціативні правила зазвичай використовуються при аналізі продуктових кореляцій (Market Basket Analysis) і виглядають приблизно так «якщо в чеку клієнта є молоко, то в 80% випадків там буде і хліб». Тобто якщо ми бачимо, що молоко в кошик клієнт вже поклав, саме час нагадати про хліб.

Це не те ж саме, що аналіз рознесених в часі покупок, але якщо ми будемо вважати всю історію одним великим кошиком, то цілком можемо застосувати цей принцип і тут [7].

RBM (restricted Bolzman Machines)

Обмежені машини Больцмана — відносно старий підхід, заснований на стохастичних рекурентних нейронних мережах. Він являє собою модель з латентними змінними і в цьому схожий на SVD-розкладання. Тут також шукається найбільш компактний опис користувацьких вподобань, які кодуються за допомогою латентних змінних. Метод не був розроблений для пошуку рекомендацій, але він успішно використовувався в рішеннях Netflix Prize і до сих пір застосовується в деяких завданнях [7].

Автоенкодери (autoencoders)

В основі лежить все той же принцип спектрального розкладання, тому такі мережі ще називають denoising auto-encoders. Мережа спочатку згортає відомі їй дані про користувача в деяке компактне уявлення, намагаючись залишити тільки значиму інформацію, а потім відновлює дані у вихідній розмірності. У підсумку виходить якийсь усереднений, очищений від шуму шаблон, за яким можна оцінити інтерес до будь-якого продукту [7].

DSSM (deep sematic similiarity models)

Один з нових підходів. Все той же принцип, але в ролі латентних змінних тут внутрішні тензорні описи вхідних даних (embeddings). Спочатку модель створювалася для зіставлення запиту з документами (як і content-based рекомендації), але вона легко трансформується в завдання зіставлення користувачів і товарів [7].

Різноманіття архітектур глибоких мереж безмежно, тому Deep Learning надає дійсно широке поле для експериментів в області рекомендаційних систем.

1.5 Результати аналізу існуючих підходів

Проведено дослідження проблеми використання методів статистики та систем штучного інтелекту для вирішення проблем електронних освітніх платформ. Також були описані переваги і недоліки даних методів та засоби їх оптимізації.

Було оглянуто методи вирішення поставленої задачі та було обрано найбільш оптимальні:

1) Коллаборативна фільтрація на основі даних про користувачів
2) Коллаборативна фільтрація на основі даних про товари
3) Матрична факторизація
4) Найпростіша модель гібридної рекомендаційної системи, яка буде включати в себе три попередні пункти та одну довільну рекомендацію

Доведено актуальність розвитку та реалізації теми роботи в подальшому.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ ПОПЕРЕДНЬОЇ ОБРОБКИ ДАНИХ ТА РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Попередня обробка і очищення даних

Попередня обробка і очищення даних — це важливі завдання, які необхідно виконати, перш ніж набір даних можна буде використовувати для навчання моделі. Необроблені дані часто спотворені і ненадійні, і в них можуть бути пропущені значення. Використання таких даних при моделюванні може призводити до невірних результатів. Ці завдання є частиною процесу обробки і аналізу даних групи і зазвичай мають на увазі початкове вивчення набору даних, що використовується для визначення і планування необхідної попередньої обробки.

Реальні дані збираються для подальшої обробки з різних джерел і процесів. Вони можуть містити помилки і пошкодження, негативно впливають на якість набору даних. Ось якими можуть бути типові проблеми з якістю даних:

1) неповнота: дані не містять атрибутів, або в них пропущені значення;
2) шум: дані містять помилкові записи або викиди;
3) неузгодженість: дані містять конфліктуючі між собою записи або розбіжності.

Ось що ми будемо оцінювати, щоб перевірити якість даних:

1) число записів;
2) кількість атрибутів (або компонентів);
3) типи даних атрибута (номінальний, порядковий номер або безперервний);
4) число відсутніх значень;
5) правильно сформовані дані.

Якщо дані мають формат TSV або CSV, ми маємо перевірити правильність поділу стовпців і рядків відповідними роздільниками.

Якщо дані мають формат HTML або XML — переконатися, що формат даних відповідає належним стандартам.

Для вилучення структурованої інформації з частково структурованих або неструктурованих даних також може знадобитися синтаксичний аналіз.

У випадку непослідовних записів даних — перевірити допустимість діапазону значень. Наприклад, якщо дані містять GPA-запис учня (середнє значення точки), перевірити, чи знаходиться GPA в зазначеному діапазоні.

Основні задачі попередньої обробки даних:

1) очищення даних (заповнення відсутніх значень, виявлення і видалення шуму даних і викидів);
2) перетворення даних (нормалізація для зменшення розмірів і шуму);
3) ущільнення даних (створення вибірки даних або атрибутів для спрощення обробки даних);
4) дискретизація даних — (перетворення безперервних атрибутів в категоріальні, щоб простіше було використовувати деякі методи машинного навчання);
5) очищення тексту (видалення специфічних символів для вирівнювання і форматування).

2.1.1 Обробка пропущених значень

При роботі з пропущеними значеннями краще спочатку визначити причину їх появи в даних, що допоможе вирішити проблему. Ось які бувають методи обробки пропущених значень:

1) вилучення (видалення) записів з пропущеними значеннями;
2) фіктивна підстановка — заміна пропущених значень фіктивними, наприклад підстановка значення unknown (невідомо) або значення 0;
3) підстановка середнього значення;
4) підстановка медіанного значення;
5) підстановка по регресії (використання регресійного методу для заміни пропущених значень).

У даній роботі перевага буде надаватися заміні медіанним значенням як для категоріальних значень так і для числових.

2.1.2 Нормалізація даних

Нормалізація даних дозволяє масштабувати числові значення в зазначений діапазон.

Методи нормалізації даних:

1) нормалізація за методом мінімаксу (Рис. 14): лінійне перетворення даних в діапазоні, наприклад, від 0 до 1, де мінімальне і максимальне значення масштабовані і відповідають 0 і 1 відповідно;

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

Рис. *14 Нормалізація методом мінімаксу*

Це найбільш популярний спосіб нормалізації. Для того, щоб застосувати цей метод, має бути відомо максимальне і мінімальне значення ознаки. Проблема в тому, що ці значення відомі не завжди.

2) нормалізація по Z-показником: масштабування даних на основі середнього значення і стандартного відхилення: розподіл різниці між даними і середнім значенням на стандартне відхилення;
3) десяткове масштабування: масштабування даних шляхом видалення десяткового роздільника значення атрибута.

2.1.3 Дискретизація даних

Дані можна дискретизувати, перетворивши безперервні значення в номінальні атрибути або інтервали. Це можна зробити декількома способами.

1) групування рівної ширини: поділ діапазону всіх можливих значень атрибута в групи (N) однакового розміру з подальшим присвоєнням значень, що відносяться до осередку з відповідним номером;

2) групування рівної висоти: поділ усіх можливих значень атрибута в групи (N), що містять однакову кількість примірників, з подальшим присвоєнням значень, що відносяться до осередку з відповідним номером.

2.1.4 Корекція різко відмінних значень

Викид (англ. outlier) — у статистиці результат вимірювання, який виділяється із загальної вибірки.

Увибірці можуть бути присутні значення, що є викидами. Це, як правило, не помилки. Однак, своїми значеннями вони "шокують" модель. Хоча при цьому викиди часто є нормальною частиною розподілу — так, в нормальному розподілі кожне 22-е вимірювання буде виходити з «двох сигм», і кожне 370-е — з трьох.

Для того, щоб визначити, чи є значення викидом, користуються характеристикою вибірки, яка називається інтерквартільним розмахом. Визначається він у такий спосіб:

IQR = Q3 – Q1

де Q1 — перша квартіль, — таке значення ознаки, менше якого рівно 25% всіх значень. Q3 — третя квартіль, значення, менше якого рівно 75% всіх значень.

Для того, щоб зрозуміти, чи є значення викидом, можна скористатися евристикою: викиди лежать за межами наступного інтервалу:

[Q1 – 1.5IQR, Q3 + 1.5IQR]

Найчастіше від викидів в навчальній вибірці краще позбуватися.

2.1.5 One-hot encoding

Це спосіб попередньої обробки категоріальних ознак. Багато моделей погано працюють з категоріальним ознаками взагалі. Справа в тому, що слово "Україна" не можна просто взяти і помножити на яке-небудь число. Але багато моделей працюють саме так: береться коефіцієнт і на нього множить-

ся значення ознаки. Аналогічна операція виконується з іншими ознаками. Всі результати сумуються. На основі значення суми робиться висновок про належність об'єкта до того чи іншого класу (такі моделі називаються лінійними).

Однак, як поступати з ознаками, значення яких не можна виразити чисельно? Можна замінити їх значення чисельним ідентифікатором. Наприклад, замість значення "Україна" використовувати значення 1, а замість "Франція" — 2. Тоді лінійна модель буде працювати. Але, якщо ми так вчинимо, ми втратимо властивість категоріальності ознаки. Тобто модель буде намагатися порівнювати ідентифікатори ознак між собою. Але вони не порівнянні за значенням.

Щоб боротися з цією проблемою, був придуманий спосіб перетворити вхідну ознаку в кілька нових, бінарних ознак. Наприклад, ми можемо ознаку "Країна" перетворити в 4 нових бінарних ознаки (Рис. 15).

Id	Name	Country
1	Anna	Ukraine
2	Margo	Spain
3	Christian	France

Id	Name	Ukraine	Spain	France
1	Anna	1	0	0
2	Margo	0	1	0
3	Christian	0	0	1

Рис. *15 Перетворення категоріальної ознаки в бінарну*

2.2 Теорія рекомендаційних систем та персоналізації

Стрічка, яку людина бачить на Facebook або Youtube — тільки її, іншої такої немає ні у кого. Чому деякі сайти виглядають однаково для кожної лю-

дини, а деякі навпаки різні? Відповідь в тому, що деякі ресурси дуже добре **персоналізовані**.

Персоналізація інтернет-сайту — це його автоматичне підстроювання під поточні потреби конкретного користувача.

Персоналізацію можна розділити на два протилежні підходи:

1. Перший — **персоналізація, що розширюється**.

   Коли на основі деякого знання про користувача йому пропонується додаткова інформація, імовірно йому корисна. Характерним прикладом є товарні рекомендації, найчастіше зустрічаються в інтернет-магазинах, наприклад, з написом «З цим товаром купують…» [6].
2. Другий — **персоналізація, що звужується**. Характерним прикладом її є алгоритм стрічки Facebook, який з потоку постів друзів відбирає і показує користувачеві лише ті пости, які імовірно найбільше його зацікавлять [6].

Два ці підходи вирішують протилежні завдання. Перший — пропонує раніше невідому інформацію. Другий — навпаки, рятує від перевантаження зайвою інформацією.

Що таке рекомендаційна система?

Рекомендаційна система — це програмний комплекс, який визначає інтереси і переваги відвідувача і дає рекомендації відповідно до них (Рис. 16) [6].

Існують різні підходи до розробки рекомендаційних систем, які можуть застосовуватися в залежності від:

1. Доступних даних про користувачів і рекомендованих сутності.
2. Видів явного і неявного фідбеку користувачів.
3. Предметної області.
4. Обмежень на швидкість оновлення рекомендацій.

Рис. *16 Теорія рекомендаційної системи*

При цьому метою класичної рекомендаційною системи є рекомендація товарів, раніше невідомих користувачеві, але йому корисних або цікавих в поточному контексті.

На цю основну мету рекомендаційних систем може бути кілька різних точок зору, які обумовлюють різні критерії оцінки успішності рекомендаційної системи:

1. З точки зору відвідувача — наскільки йому подобаються рекомендовані товари. Метрики цього — якість апроксимації оцінки або CTR рекомендаційної системи і частка відвідувачів, що купили рекомендований товар [6].
2. З точки зору інтернет-магазину або інтернет-видання — максимізація очікуваного доходу з користувача або глибини перегляду [6].

2.2.1 Коллаборативна фільтрація у рекомендаційних системах

Спільна фільтрація (або коллаборативна фільтрація, скорочено КФ) — це техніка, що використовується в системах рекомендування. Спільна фільтрація має два підходи — вузький та більш загальний.

У новому, вужчому сенсі спільна фільтрація — це метод автоматичного прогнозування (фільтрування) інтересів користувача шляхом збору переваг або смакової інформації від багатьох користувачів (коллаборація).

User-Based Collaborative Filtering

Спочатку нам потрібно знайти список користувачів, подібних до цільового користувача. Ми робимо це, порівнюючи рейтинги товарів цільового користувача з рейтингами товарів кожного іншого користувача в таблиці (Рис. 17).

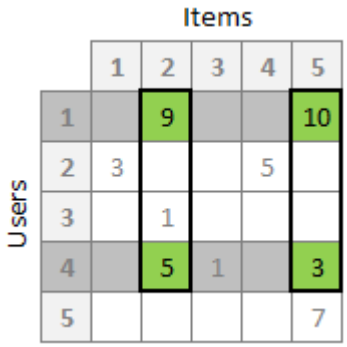

Рис. *17 Пошук схожих користувачів у таблиці*

Наприклад, оскільки користувачі 1 та 4 мають оцінки 2 та 5, ми можемо обчислити оцінку подібності між двома користувачами на основі цих 4 оцінок (Рис. 14). Потрібно додатково звернути увагу, що ми не порівнюємо весь рядок користувача; ми порівнюємо лише елементи, які обидва користувачі оцінили.

Існує кілька способів обчислення оцінок схожості між користувачами, і всі вони беруть два користувацькі рядки як вхідні дані і дають одне числове значення, що представляє подібність як вихідний результат. До часто використовуваних для цього алгоритмів належать подібність косинусів та кореляція Пірсона, обидва з яких реалізовані в цьому проекті.

Після того, як ми обчислимо оцінку схожості між цільовим користувачем та кожним іншим користувачем у таблиці, ми можемо відсортувати цих

подібних користувачів (або сусідів) і зберегти лише найбільш подібних (або найближчих). Цей алгоритм належним чином називається алгоритмом K-Найближчих сусідів (K-Nearest Neighbors — KNN).

Тепер, коли у нас є список користувачів, найбільш схожих на цільового користувача (щодо яких ми отримуємо рекомендацію), ми можемо захопити елементи, прочитані цими сусідами, і рекомендувати їх цільовому користувачеві (Рис. 18).

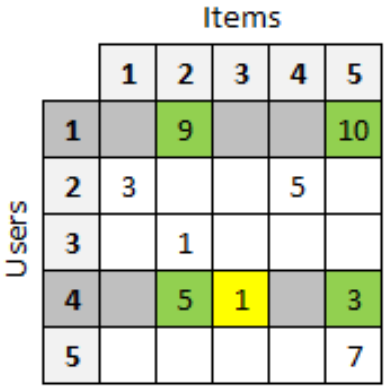

Рис. *18 Рекомендація на основі схожості між користувачами*

Наприклад, давайте знайдемо рекомендацію для користувача 1. Спочатку ми шукаємо його найближчого сусіда, який виявляється користувачем 4 (на основі подібності між пунктами 2 і 5). З таблиці (Рис. 15) ми бачимо, що наш найближчий сусід (користувач 4) любить пункт 3, тому ми можемо рекомендувати пункт 3 користувачеві 1, який ще не розглянув цей елемент.

Основне припущення коллаборативного фільтруючого підходу полягає в тому, що якщо людина A має таку саму думку, як і людина B щодо певного питання, то A буде схильна до думки B щодо іншого питання, ніж до думки випадково вибраної людини. Наприклад, система спільних рекомендацій щодо фільтрації для смаків телебачення може передбачити, який телевізійний показ користувачеві слід сподобатися, отримавши частковий список смаків цього користувача (подобається чи не подобається).

Ці прогнози є специфічними для користувача, але використовують інформацію, отриману від багатьох користувачів. Це відрізняється від більш простого підходу — дати середній (неспецифічний) бал за кожний предмет, наприклад, виходячи з його рейтингу.

Зростання Інтернету значно ускладнило ефективне отримання корисної інформації з усіх доступних даних в Інтернеті. Переважна кількість даних потребує механізмів ефективної фільтрації інформації. Спільна фільтрація — одна з методик, що застосовуються для вирішення цієї проблеми.

Мотивація спільної фільтрації походить від ідеї, що люди часто отримують найкращі рекомендації від когось із смаком, схожим на себе.

Спільна фільтрація охоплює методи відповідності людей з подібними інтересами та надання рекомендацій на цій основі.

Алгоритми спільної фільтрації часто вимагають:

1) активної участі користувачів;
2) простого способу представлення інтересів користувачів;
3) алгоритмів, здатних відповідати людям з подібними інтересами.

Зазвичай робочий процес спільної системи фільтрації такий:

1) користувач висловлює свої переваги рейтинговими елементами (наприклад, книгами, фільмами чи компакт-дисками) системи. Ці рейтинги можна розглядати як приблизне відображення інтересу користувача до відповідного домену;
2) система відповідає рейтингу цього користувача порівняно з іншими користувачами і знаходить людей з найбільш «схожими» смаками;
3) система рекомендує предмети, які подібні користувачі оцінили високо, але ще не оцінені цим користувачем (імовірно, відсутність рейтингу часто розглядають як незнайомість товару).

Ключова проблема спільної фільтрації — як поєднувати та зважувати переваги сусідів користувачів. Іноді користувачі можуть негайно оцінити рекомендовані елементи. В результаті система отримує все більш точне представлення вподобань користувачів з часом.

Item-Based Collaborative Filtering

Item-based системи працюють дуже подібно до систем на базі схожості користувачів. Але при спробі знайти пропозицію для даного користувача, вони знаходять предмети, подібні до елементів, які користувач вже бачив.

Основна відмінність полягає в тому, що замість пошуку найближчих користувачів ми знаходимо найближчі предмети (Рис. 19).

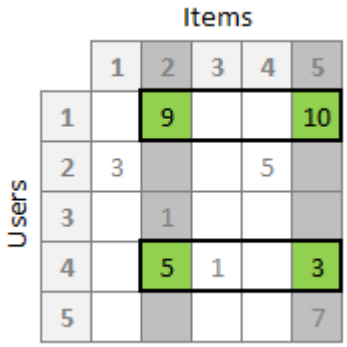

Рис. *19 Item-based таблиця схожості*

Наприклад, оскільки користувачеві 4 сподобався елемент 2, ми можемо знайти елементи, подібні до елемента 2 (наприклад, елемент 5), і запропонувати його користувачеві 4 (Рис. 19). Для порівняння елементів між собою використовуються такі самі методи, що і для порівняння користувачів.

2.2.2 Матрична факторизація в системах рекомендацій

Матрична факторизація — це клас алгоритмів, що використовуються у рекомендаційних системах. Алгоритми матричної факторизації працюють шляхом декомпозиції матриці взаємодії між елементами користувача на добуток двох прямокутних матриць нижчої розмірності [9].

Це сімейство методів стало широко відомим під час Netflix prize challenge завдяки його ефективності, про що повідомляв Саймон Функ у своїй публікації в блозі 2006 року [8], де він поділився своїми висновками з дослідницькою спільнотою.

Ідея матричної факторизації полягає у представленні користувачів та елементів у латентному просторі з меншими розмірами. З моменту першої роботи Funk у 2006 році для систем рекомендування було запропоновано безліч підходів до матричної факторизації.

Модельні системи працюють інакше, ніж спільні системи. Замість того, щоб намагатися робити пропозиції, знаходячи подібних користувачів чи предмети, вони будують математичну модель для прогнозування.

На вибір є багато різних алгоритмів, заснованих на моделях. Одним з найпоширеніших підходів називається декомпозиція сингулярних значень (SVD) і працює шляхом розбиття матриці елемента користувача на менші матриці (Рис. 20), які потім можна використовувати для заповнення відсутніх оцінок.

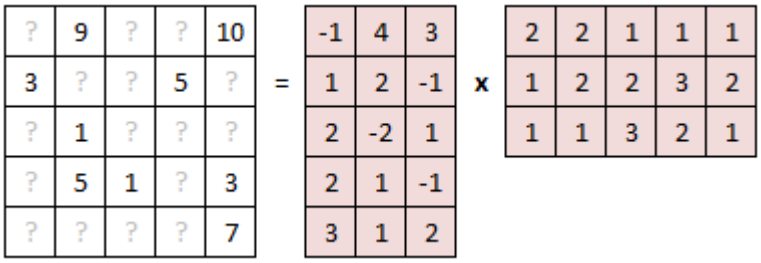

Рис. *20 Матрична факторизація*

Насправді, чиста математична форма SVD не працює надто добре, коли у вас відсутні значення, тому нам доведеться використовувати варіацію SVD ++, яка оптимізована для матриць з відсутніми даними.

Факторизація матриць, як і звичайна числова факторизація (наприклад, 20 = 4 x 5), є процесом пошуку окремих матриць, які при множенні разом дають вихідну матрицю. Теорія полягає в тому, що якщо ми можемо знайти дві менші матриці, які точно апроксимують нашу матрицю Users-Items , то ми, сподіваємось, можемо також наблизити відсутні значення.

Коли ми вираховуємо нашу матрицю, ми можемо вибрати кількість функцій, які ми хочемо використовувати. У підсумку ця кількість функцій

стає одним із двох вимірів кожної матриці. Наприклад, наша матриця функцій користувача матиме таку ж кількість рядків, що і матриця статті користувача, але матиме кількість стовпців, рівну кількості вибраних функцій.

Вибір кількості функцій не має конкретних рекомендацій та залежить від вибірки. Це число зазвичай визначається емпірично. Якщо їх замало — ми не зможемо вивчити наявні дані. Якщо їх занадто багато, замість цього ми можемо запам’ятати (або перенавчити) алгоритм.

Feature матриці містять всю інформацію, яка нам потрібна для апроксимації будь-якого значення у вихідній матриці (Рис. 21). Нам просто потрібно обчислити точковий добуток між відповідним стовпцем user-feature матриці та рядком матриці item-feature.

Items

Users	1	2	3	4	5	3	4	5
1		9			10	-1	4	3
2	3			5		1	2	-1
3		1				2	-2	1
4		5	1		3	2	1	-1
5					7	3	1	2
A	2	2	1	1	1			
B	1	2	2	3	2			
C	1	1	3	2	1			

Рис. *21 Users-Items Feature матриця*

Точковий добуток визначається як сума внутрішнього добутку (Рис. 22), тому в наведеному нижче прикладі ми можемо розрахувати рейтинг користувача 2 для елементу 4 так: (1 x 1) + (3 x 2) + (2 x – 1) = 1 + 6 – 2 = 5

Users \ Items	1	2	3	4	5	X	Y	Z
1		9			10	-1	4	3
2	3			5		1	2	-1
3		1				2	-2	1
4		5	1		3	2	1	-1
5					7	3	1	2
A	2	2	1	1	1			
B	1	2	2	3	2			
C	1	1	3	2	1			

Рис. *22 Визначення точкового добутку користувача 2 для елементу 4*

Після створення цих feature матриць ми можемо використовувати їх для обчислення вихідних значень, а також заповнення відсутніх значень. Ось як ми будемо генерувати пропозиції та прогнозувати рейтинги (Рис. 23).

5	9	16	17	10	=	-1	4	3	x	2	2	1	1	1
3	5	2	5	4		1	2	-1		1	2	2	3	2
3	1	1	-2	-1		2	-2	1		1	1	3	2	1
4	5	1	3	3		2	1	-1						
9	10	11	10	7		3	1	2						

Рис. *23 Матрична факторизація. Прогнозування рейтингів*

Єдина проблема полягає в тому, що ми не можемо використовувати чисто математичну форму факторизації матриць, або ми навчимося передбачати нулі для відсутніх даних. Нам потрібен інший спосіб знайти ці матриці, щоб ми тренувалися лише на ненульових значеннях.

Щоб знайти оптимальні матриці, ми випадково спробуємо різні значення, побачимо, наскільки добре вони виконуються, внесемо невеликі коригування та спробуємо ще раз. Ми будемо повторювати цей процес, поки наші матриці функцій не зможуть точно передбачити відомі значення.

Алгоритм, який ми будемо використовувати, називається стохастичним градієнтним спуском. Тобто ми збираємося ітеративно коригувати наші вхідні параметри, поки не отримаємо результат, який нас задовольнить.

2.2.3 Funk MF

Оригінальний алгоритм, запропонований Саймоном Функ у своїй публікації в блозі [8], визначив матрицю рейтингу користувача-елемента як добуток двох матриць нижнього розміру, перша — рядок для кожного користувача, а друга — стовпець для кожного елемента.

Рядок або стовпець, пов’язаний з певним користувачем або елементом, називають прихованими факторами. Це SVD-подібна модель машинного навчання.

Передбачувані рейтинги можна обчислити як

$$\tilde{R} = HW$$

де $\tilde{R} \in R^{users*items}$ - матриця оцінювання елементів користувача,

$H \in R^{users*latentFactors}$ містить латентні фактори користувача та

$W \in R^{latentFactors*items}$приховані фактори.

Передбачуваний рейтинг, який користувач **u** надасть позиції **i**, обчислюється як:

$$\tilde{r}_{ui} = \sum_{f=0}^{n\,factors} H_{u,f} W_{f,i}$$

Збільшення кількості прихованих факторів поліпшить персоналізацію і якість рекомендацій, поки кількість факторів не стане занадто високою, в цей момент якість рекомендацій знизиться. Загальна стратегія уникнення перевитрати — додати терміни регуляризації до цільової функції [9].

Funk MF був розроблений як проблема прогнозування рейтингу, тому він використовує явні числові оцінки.

2.3 Результати дослідження методів обробки вибірки та реалізації рекомендаційних систем

В ході дослідження методів попередньої обробки даних та засобів реалізації рекомендаційних систем були обрані методи для реалізації у проекті. Після аналізу даних про курси і користувачів було вирішено реалізувати наступні методи попередньої обробки даних:

1. Нормалізація вибірки (методом мінімаксу).
2. Дискретизація даних.

Для реалізації рекомендаційної системи було обрані:

1. Коллаборативна фільтрація на основі даних про користувачів.
2. Коллаборативна фільтрація на основі даних про курси.
3. Матрична факторизація

Також було вирішено створити гібридну рекомендаційну систему, що буде повертати рекомендації на основі зазначених методів, а також додавати відсоток випадкових рекомендацій.

Така комбінація методів дасть можливість зробити детальніше порівняння засобів реалізації рекомендаційних систем та їх ефективність у використанні для платформ електронної освіти.

РОЗДІЛ 3 ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ РЕКОМЕНДАЦІЙ НАВЧАЛЬНИХ МАТЕРІАЛІВ

3.1 Архітектура системи

Рекомендаційна система для платформ електронної освіти складається з двох основних частик: бібліотеки, яка безпосередньо реалізовує можливості різних методів рекомендаційних систем та клієнтського застосунку (наприклад, освітньої платформи), до якого буде підключатися модуль рекомендаційної системи (Рис. 24). У данному проекті в якості клієнтського застосунку був створений десктопний застосунок на Windows Forms.

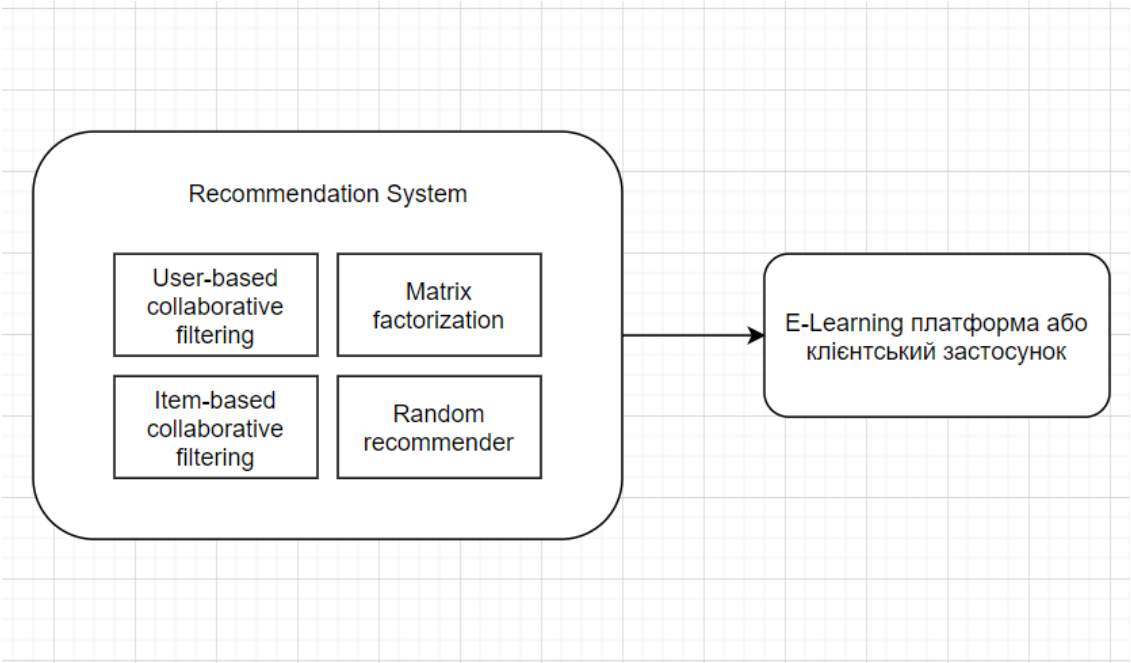

Рис. *24 Архітектура системи*

Архітектура модуля рекомендаційної системи

На Рис. 25 зображена частина архітектури бібліотеки, що стосується ієрархії рекомендаційних систем. Користувач (тобто клієнтський застосунок) взаємодіє з модулем лише через програмний інтерфейс. У даній системі це інтерфейс рекомендаційної системи — Interface Recommender (Рис. 25).

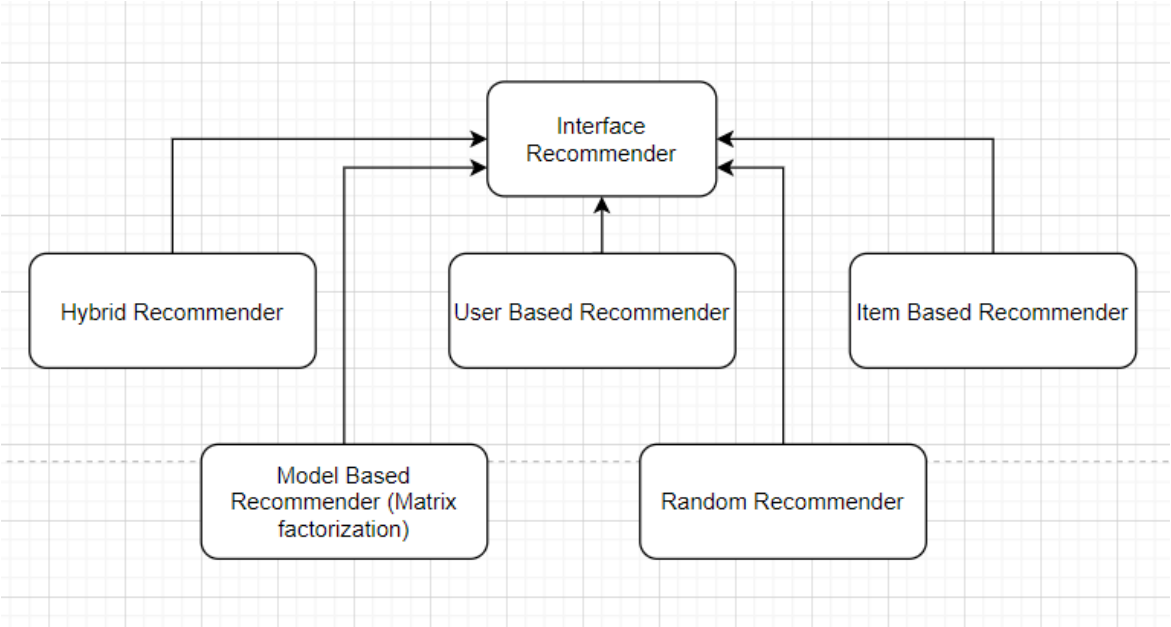

Рис. *25 Ієрархія реалізацій рекомендаційних систем*

Дана система може обробляти дані, які мають структуру User-Items-Tags.

3.2 Засоби реалізації

Під час розробки програмної системи, через низку переваг, було вирішено використовувати наступні засоби:

1. Середовище розробки Visual Studio та Visual Studio Code.
2. Мова програмування C#.
3. Платформа .NET Core.
4. CSV формат зберігання даних.
5. Фреймворк для машинного навчання ML.NET.

Основними параметрами при виборі технологій були: кросплатформність, популярність (велика спільнота користувачів даної технології полегшує процес виявлення помилок та надає більше надійності створеній системі), зручність використання, орієнтованість на кращі підходи у галузі роботи зі статистикою та машинним навчанням з використанням C#.

Microsoft Visual Studio

Дана IDE була обрана через ряд переваг, таких як зручна робота з пакетами та розширеннями, зручність і простота у використанні, має багато інструментів для аналізу коду та роботи з помилками, зручний механізм відладки застосунків, можливість відстежування даних про performance, проста і зручна робота з базовими можливостями GIT, тощо.

Також окремо треба відзначити user-friendly інтерфейс та технологію автодоповнення IntelliSense, що значно пришвидшує процес написання коду та робить його зручнішим.

Microsoft Visual Studio Code

Перевагами даного середовища є кросплатформність (доступні версії для платформ Windows, Linux і OS X), ліцензія, що дозволяє безкоштовне використання в будь-яких цілях (в тому числі для комерційного використання), проста і зручна робота з базовими можливостями GIT, багато інструментів для аналізу коду та роботи з помилками

Також важливим критерієм, що може значно полегшувати процес розробки є велика кількість розширень для даного редактора коду. На момент написання кваліфікаційної роботи можна завантажити і встановити кілька тисяч розширень тільки в категорії «programming languages» (мови програмування) [17].

Мова програмування C#

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з системою типізації, має синтаксис близький до C++ і Java.

Основними перевагами у виборі мови були: можливість розробки кросплатформних застосунків, зручність використання, C-подібний синтаксис, статична типізація та задовільний performance у роботі з математичними операціями та великими обсягами даних, можливість компіляції в native code (остання можливість зображена на Рис. 26) та наявність garbage collection.

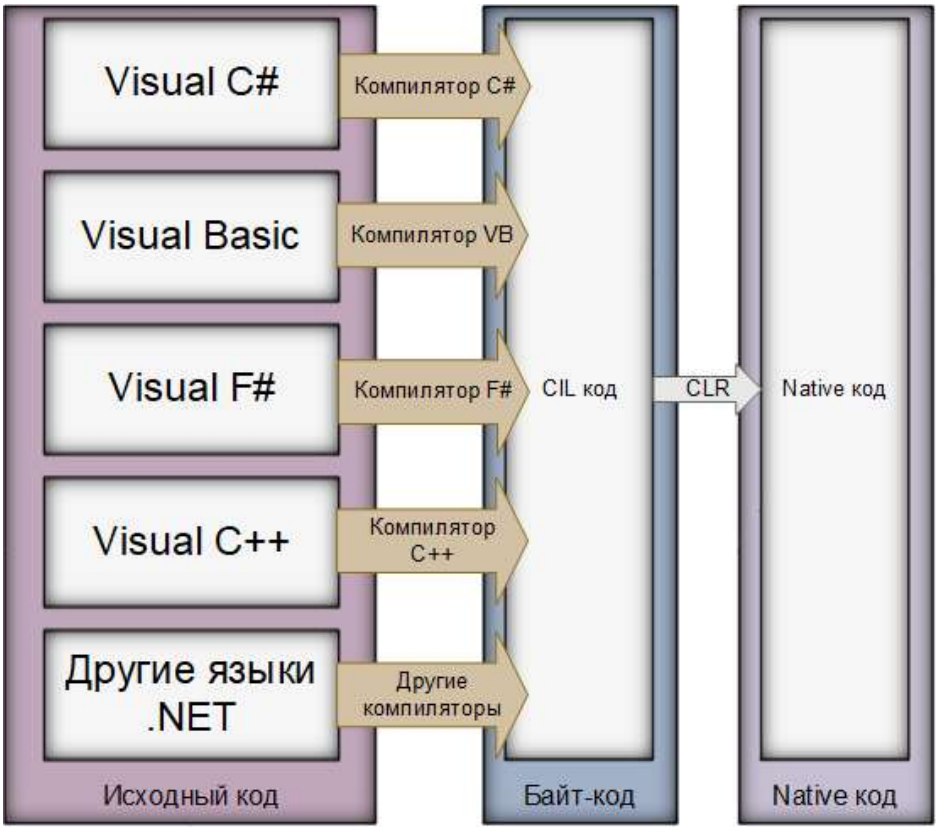

Рис. *26 CLR*

CLR також забезпечує такі важливі служби як:

1) управління пам'яттю;
2) управління потоками;
3) обробка винятків;
4) збірка сміття;
5) безпека виконання.

Також важливо відзначити, що C# є найшвидшою у виконанні CLI мовою [21, 22].

Платформа .NET Core

Основні переваги даної платформи, що були розглянуті в контексті реалізації даної кваліфікаційної роботи: кросплатформність (сумісність з такими операційними системами як Windows, Linux і macOS), велике community, можливість застосування хмарних технологій в майбутньому.

Остання .NET Core 3.1 була випущена 3 грудня 2019 [23]. Також Microsoft оголосили, що наступна версія буде називатися .NET 5 (без викорис-

тання слова "Core" в назві). Тобто вже майже повністю усунутий розрив у можливостях з .NET Framework 4.8 і нова версія стане єдиною платформою для .NET розробки [24].

3.2.1 Comma-Separated Values формат зберігання даних

Файл Comma-Separated Values (CSV) — це текстовий файл, розділений комами, який використовує символ «,» для розділення значень. Кожен рядок файлу є записом даних. Кожен запис складається з одного або декількох полів, розділених комами.

Використання коми як роздільника поля є джерелом імені для цього формату файлу. CSV-файл зазвичай зберігає табличні дані (цифри та текст) у вигляді простого тексту, і в цьому випадку кожен рядок матиме однакову кількість полів.

Формат файлу CSV не повністю стандартизований. Основна ідея розділення полів комою зрозуміла, але ситуація ускладняється, коли дані полів містять також коми або вбудовані розриви рядків. Реалізації CSV можуть не обробляти такі дані полів, або вони можуть використовувати лапки, щоб оточити поле. Але деякі поля можуть потребувати вбудованих лапок, тому реалізація CSV може включати символи екранування або екрановані послідовності.

Крапка з комою часто використовується замість коми у багатьох європейських регіонах, щоб використовувати кому як десятковий роздільник і, можливо, крапку як десятковий символ групування.

CSV — це загальний формат обміну даними, який широко підтримується споживачами, бізнесом та науковими додатками. Серед найпоширеніших видів використання — переміщення табличних даних між програмами, які спочатку працюють у несумісних (часто недокументованих) форматах [25]. Це працює, незважаючи на відсутність дотримання стандарту RFC 4180 (або будь-якого іншого стандарту), оскільки багато програм підтримують зміни у форматі CSV для імпорту даних.

Наприклад, користувачеві може знадобитися перенести інформацію із програми бази даних, яка зберігає дані у власному форматі, до електронної таблиці, яка використовує зовсім інший формат. Програма бази даних, швидше за все, може експортувати свої дані як CSV. Потім експортований файл CSV може імпортувати програма електронних таблиць.

Формат файлу CSV підтримується майже всіма електронними таблицями та системами управління базами даних, включаючи Apple Numbers, LibreOffice Calc та Apache OpenOffice Calc, Microsoft Excel.

Формат CSV підтримується бібліотеками, доступними для багатьох мов програмування. Більшість із них надають певний спосіб вказати розділь-ник поля, десятковий роздільник, кодування символів, правила цитування, формат дати тощо.

Багато утиліт в системах Unix можуть можуть обробляти прості файли CSV. Однак цей метод неправильно обробляє коми у рядках із цитуваннями.

3.2.2 Фреймворк для машинного навчання ML.NET

ML.NET — це безкоштовна програмна бібліотека машинного навчання для мов програмування C # та F #. Вона також підтримує моделі Python при використанні разом із NimbusML. Попередній випуск ML.NET включав трансформації для розробки функцій, таких як створення n-грамів, створення моделей, що навчаються обробляти двійкову класифікацію, багатокласову класифікацію та завдання регресії [26]. З тих пір були додані додаткові завдання ML, такі як системи виявлення аномалій та рекомендації. Глибоке навчання, буде включено в наступні версії [27].

ML.NET надає можливість аналізувати можливості машинного навчання на основі моделей існуючим розробникам .NET.

Фреймворк побудований на .NET Core та .NET Standard, що успадковують можливість запуску кросплатформних систем на Linux, Windows та macOS. Хоча фреймворк ML.NET є новим, його витоки розпочались у 2002 році як проект Microsoft Research під назвою TMSN (text mining search and

navigation — пошук та навігація тексту) для внутрішнього використання в продуктах Microsoft. ML.NET походить з бібліотеки TMSN і в значній мірі перевершив своїх батьків, пише д-р Джеймс Маккаффрі, Microsoft Research [28].

Розробники можуть навчити модель або повторно використати існуючу модель та запустити її в будь-якому середовищі в автономному режимі. Це означає, що розробникам не потрібно мати досвід роботи в Data Science, щоб використовувати фреймворк. Підтримка формату моделі глибокого навчання з відкритим кодом Open Neural Network Exchange (ONNX) була представлена зі збірки 0.3 у ML.NET. Випуск включав інші помітні вдосконалення, такі як Factorization Machines, LightGBM, Ensembles, LightLDA transform. Інтеграція ML.NET TensorFlow увімкнена з випуску 0.5. Підтримка додатків x86 та x64 була додана до збірки 0.7, включаючи розширені можливості рекомендацій із розділенням матриць [28].

Звіт Microsoft про машинне навчання за допомогою ML.NET продемонстрував, що він здатний навчати моделі аналізу настроїв, використовуючи великі набори даних, одночасно досягаючи високої точності. Їх результати показали 95% точність набору даних огляду Amazon розміром 9 ГБ [29].

Зараз фреймворк ML.NET активно використовується у наступних задачах:

1) аналіз настрою користувачів;
2) створення рекомендаційних систем;
3) передбачення змін у цінах;
4) сегментація клієнтів (кластеризація);
5) розпізнавання образів;
6) виявлення шахрайства з кредитними картками (бінарна класифікація);
7) виявлення точок зміни продажів продукції;
8) класифікація зображень;
9) передбачення змін у продажах.

3.3 Модулі і алгоритми

На Рис. 27 зображені модулі, які використовувались для тестування і порівняння між собою. Перший модуль — це гібридна система рекомендацій, що повністю складається з реалізованих протягом виконання кваліфікаційної роботи алгоритмів. Дана система отримує відповіді з чотирьох алгоритмів рекомендацій у різних пропорціях (відсоток кількості рекомендацій від кожної системи можна налаштовувати). Потім вона сортує отримані результати та повертає перелік рекомендацій для даного користувача. Другий модуль — рекомендаційна система, реалізована за допомогою ML.NET. В основі даної системи також лежить алгоритм матричної факторизації.

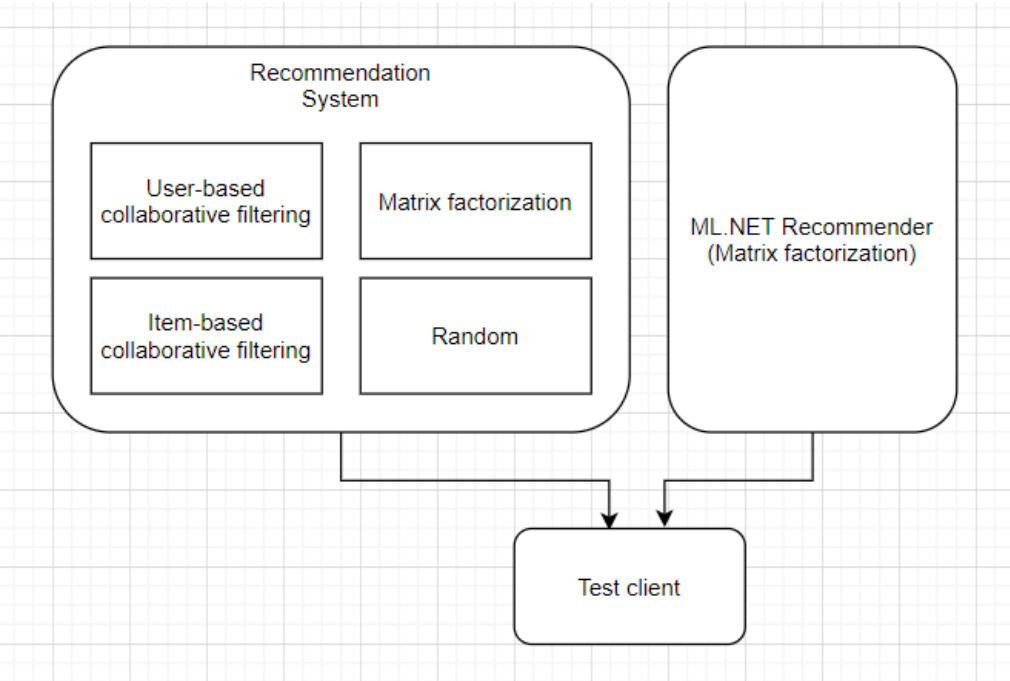

Рис. *27 Модулі рекомендаційної системи*

Також реалізована рекомендаційна система складається з менших частин, що відповідають за абстракцію та відокремлення різних складових алгоритмів для зменшення зв’язності між класами та для надання можливості легко розширювати систему в майбутньому. Ці складові частини зображені на Рис. 28.

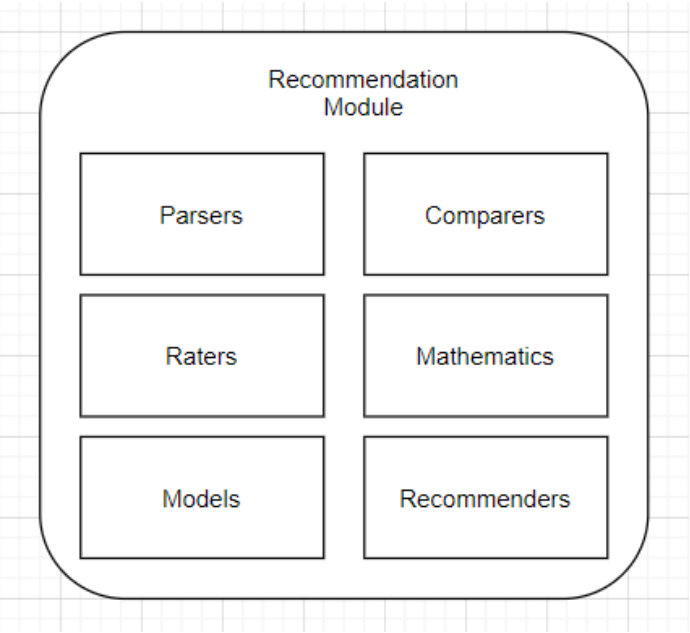

Рис. *28 Модуль рекомендаційної системи*

Модуль Parsers відповідає за зчитування та парсинг даних з CSV файлу у класи Models, що належать конкретній бізнес сфері (в даній кваліфікаційній роботі — сфері електронної освіти). На Рис. 29 зображена діаграма класів даного модуля.

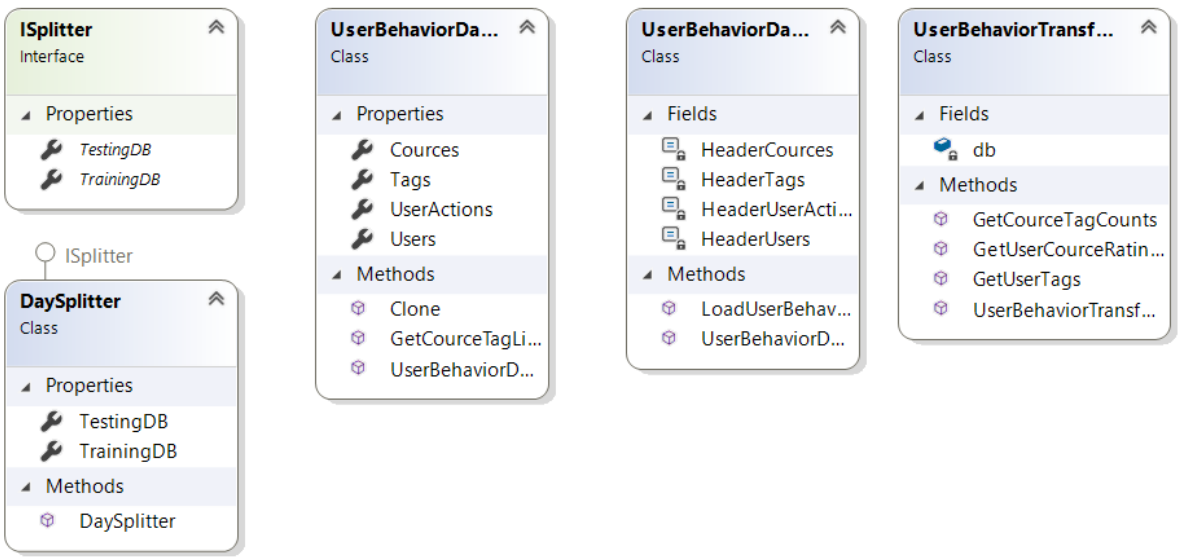

Рис. *29 Діаграма класів модулю Parsers*

На Рис. 30 представлена діаграма класів предметної області. Дана діаграма відповідає структурі даних, яка може оброблюватися реалізованими рекомендаційними системами. Також тут присутні моделі, що використовуються модулем для тестування та для висування припущень щодо рейтингів (оцінок) курсів.

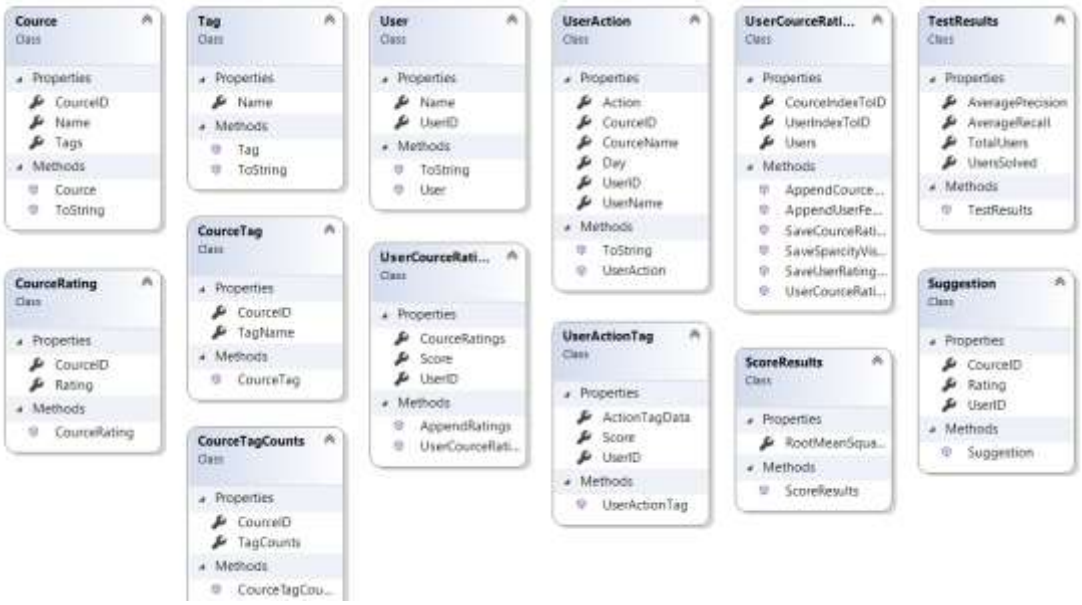

Рис. *30 Моделі предметної області*

Модуль Mathematics включає класи для роботи з математичними сутностями — з матрицями та матричною факторизацією (Рис. 31).

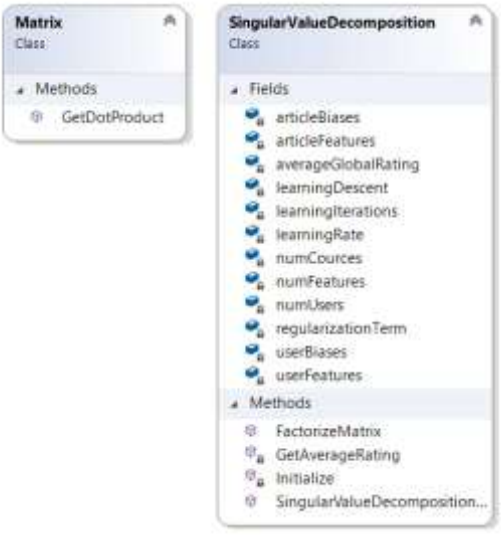

Рис. *31 Модуль для роботи з математичними сутностями*

Модуль Comparers реалізує різні алгоритми для вирахування різниці між двома векторами (Рис. 32).

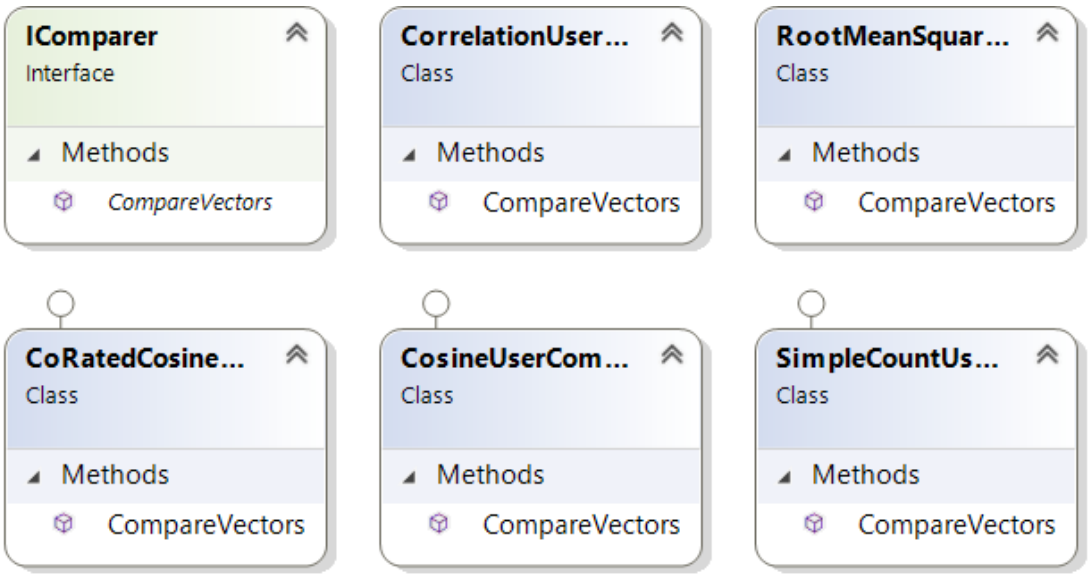

Рис. *32 Модуль Comparers*

Основний модуль даної системи — модуль, що реалізує основні алгоритми рекомендаційних систем. Діаграма класів модуля Recommenders зображена на Рис. 33.

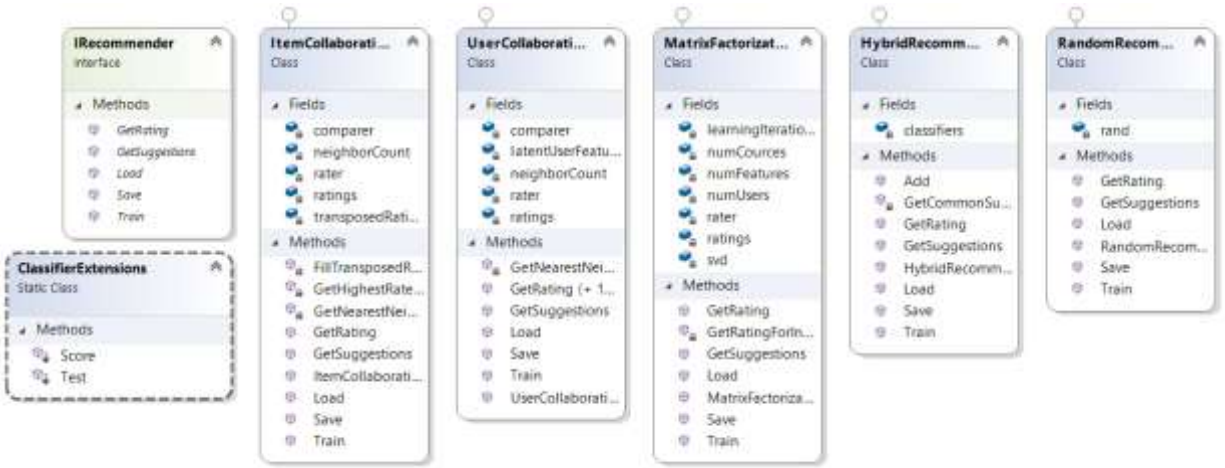

Рис. *33 Модуль рекомендаторів*

IRecommender представляє собою інтерфейс для роботи з будь-якою рекомендаційною системою і узагальнює основні функцій рекомендаторів (Лістинг 1).

Лістинг *1 Узагальнення основних функцій рекомендаторів*

```
public interface IRecommender
    {
        void Train(UserBehaviorDatabase db);

        List<Suggestion> GetSuggestions(int userId, int
numSuggestions);

        double GetRating(int userId, int articleId);

        void Save(string file);

        void Load(string file);
    }
```

Першою рекомендаційною системою була реалізована коллаборативна фільтрація на основі користувачів. Ця форма рекомендатора ґрунтується на припущенні, що користувачі, які мали схожу думку в минулому, ймовірно, можуть мати схожу думку і в майбутньому щодо деяких предметів оцінки.

За допомогою таблиці Користувач-Курс спочатку знаходиться перелік користувачів, подібних до цільового користувача. Це відбувається завдяки порівнянню рейтингів курсів цільового користувача з рейтингами курсів кожного іншого користувача в таблиці (Лістинг 2).

Важливо, що весь рядок користувача не порівнюється, порівнюються лише елементи, які обидва користувачі оцінили.

Лістинг *2 Пошук N найближчих сусідів користувача*

```
private List<UserCourceRatings>
GetNearestNeighbors(UserCourceRatings user, int numUsers)
{
    List<UserCourceRatings> neighbors = new
List<UserCourceRatings>();

    for (int i = 0; i < ratings.Users.Count; i++)
    {
        if (ratings.Users[i].UserID == user.UserID)
        {
            ratings.Users[i].Score =
double.NegativeInfinity;
        }
        else
        {
            ratings.Users[i].Score =

     comparer.CompareVectors(ratings.Users[i].CourceRatings,
user.CourceRatings);
        }
    }

    var similarUsers = ratings.Users.OrderByDescending(x =>
x.Score);

    return similarUsers.Take(numUsers).ToList();
}
```

В даній кваліфікаційній роботі було реалізовано декілька алгоритмів, що порівнюють два вектори між собою: косинусна відстань та кореляція Пір-

сона. Порівняння за допомогою косинусної відстані представлено у Лістингу 3.

Лістинг *3 Косинусна відстань*

```
public double CompareVectors(double[] userFeaturesOne, double[] userFeaturesTwo)
{
    double sumProduct = 0.0;
    double sumOneSquared = 0.0;
    double sumTwoSquared = 0.0;

    for (int i = 0; i < userFeaturesOne.Length; i++)
    {
        // Only compare Cources rated by both users
        if (userFeaturesOne[i] != 0 && userFeaturesTwo[i] != 0)
        {
            sumProduct += userFeaturesOne[i] * userFeaturesTwo[i];
            sumOneSquared += Math.Pow(userFeaturesOne[i], 2);
            sumTwoSquared += Math.Pow(userFeaturesTwo[i], 2);
        }
    }

    if (sumOneSquared > 0 && sumTwoSquared > 0)
    {
        return sumProduct / (Math.Sqrt(sumOneSquared) * Math.Sqrt(sumTwoSquared));
    }
    else
```

```
    {
        return double.NegativeInfinity;
    }
}
```

Алгоритм вирахування різниці між двома векторами (користувачами) за допомогою кореляції Пірсона наведено у Лістингу 4.

Лістинг *4 Кореляція Пірсона*

```
public double CompareVectors(double[] userFeaturesOne,
double[] userFeaturesTwo)
        {
            double average1 = 0.0;
            double average2 = 0.0;
            int count = 0;
            for (int i = 0; i < userFeaturesOne.Length;
i++)
            {
                if (userFeaturesOne[i] != 0 &&
userFeaturesTwo[i] != 0)
                {
                    average1 += userFeaturesOne[i];
                    average2 += userFeaturesTwo[i];
                    count++;
                }
            }
            average1 /= count;
            average2 /= count;
            double sum = 0.0;
            double squares1 = 0.0;
            double squares2 = 0.0;
```

```
            for (int i = 0; i < userFeaturesOne.Length;
i++)
            {
                if (userFeaturesOne[i] != 0 &&
userFeaturesTwo[i] != 0)
                {
                    sum += (userFeaturesOne[i] - average1)
* (userFeaturesTwo[i] - average2);
                    squares1 += Math.Pow(userFeaturesOne[i]
- average1, 2);
                    squares2 += Math.Pow(userFeaturesTwo[i]
- average2, 2);
                }
            }
            return sum / Math.Sqrt(squares1 * squares2);
        }
```

Отримавши список найбільш подібних сусідів цільового користувача, ми можемо обробити всі можливі курси та отримати думку найближчих сусідів щодо кожного з них (Лістинг 5). Якщо рейтинг сусідів для даного курсу досить великий, то алгоритм запропонує його цільовому користувачеві.

Лістинг *5 Припущення системи на основі схожості користувачів*

```
for (int CourceIndex = 0; CourceIndex <
ratings.CourceIndexToID.Count; CourceIndex++)
{
    // If the user in question hasn't rated the given
Cource yet
    if (user.CourceRatings[CourceIndex] == 0)
    {
        double score = 0.0;
        int count = 0;
```

```
        for (int u = 0; u < neighbors.Count; u++)
        {
            if (neighbors[u].CourceRatings[CourceIndex] !=
0)
            {
                score +=
neighbors[u].CourceRatings[CourceIndex];
                count++;
            }
        }
        if (count > 0)
        {
            score /= count;
        }
        suggestions.Add(new Suggestion(userId,
ratings.CourceIndexToID[CourceIndex], score));
    }
}
```

Наступною рекомендаційною системою була реалізована коллаборативна фільтрація на основі даних про предмети рекомендації (в нашому випадку це курси). Алгоритм майже ідентичний попередньому, але в ньому відбувається порівняння не користувачів, а курсів. В основі цього алгоритму лежить припущення щодо схожості предметів рекомендацій між собою а значить і схожості їх оцінок.

Останнім алгоритмом є матрична факторизація. Основна проблема алгоритму полягає в тому, що не можливо використовувати чисто математичну форму факторизації матриць, або система навчиться передбачати нулі для відсутніх даних. Щоб знайти оптимальні матриці, реалізований алгоритм випадково пробує різні значення, перевіряє, наскільки добре вони працюють (вираховує помилку), потім вносить коригування та цикл повторюється.

Даний підхід називається стохастичним градієнтним спуском, тобто алгоритм ітеративно коригує вхідні параметри, поки не отримує задовільний результат (Лістинг 6).

Лістинг *6 Градієнтний спуск для матричної факторизації*

```
averageGlobalRating = GetAverageRating(ratings);

for (int i = 0; i < learningIterations; i++)
{
    squaredError = 0.0;
    count = 0;
    for (int userIndex = 0; userIndex < numUsers;
userIndex++)
    {
        for (int courceIndex = 0; courceIndex < numCources;
courceIndex++)
        {
            if
(ratings.Users[userIndex].CourceRatings[courceIndex] != 0)
            {
                double estimatedRating =
averageGlobalRating +
                        userBiases[userIndex] +
courceBiases[courceIndex] +

    Matrix.GetDotProduct(userFeatures[userIndex],
courceFeatures[courceIndex]);

                double error =
ratings.Users[userIndex].CourceRatings[courceIndex] -
estimatedRating;
```

```
                squaredError += Math.Pow(error, 2);
                count++;
                averageGlobalRating += learningRate *
                        (error - regularizationTerm *
averageGlobalRating);
                userBiases[userIndex] += learningRate *
                        (error - regularizationTerm *
userBiases[userIndex]);
                courceBiases[courceIndex] += learningRate *
                        (error - regularizationTerm *
courceBiases[courceIndex]);
                for (int featureIndex = 0; featureIndex <
numFeatures; featureIndex++)
                {
                    userFeatures[userIndex][featureIndex]
+= learningRate *
                                (error *
courceFeatures[courceIndex][featureIndex] -
                                regularizationTerm *
userFeatures[userIndex][featureIndex]);

courceFeatures[courceIndex][featureIndex] += learningRate *
                                (error *
userFeatures[userIndex][featureIndex] -
                                regularizationTerm *
courceFeatures[courceIndex][featureIndex]);
                }
            }
        }
    }
    squaredError = Math.Sqrt(squaredError / count);
    rmseAll.Add(squaredError);
```

```
    learningRate *= learningDescent;
}
```

Гібридна система в даній роботі є рекомендатором, який повертає дані з переліку описаних рекомендаційних систем (User-Based, Item-Based та Matrix Factorization). Також до гібридного рекомендатора був доданий відсоток випадкових рекомендацій (Лістинг 7).

Лістинг *7 Гібридна рекомендаційна система*

```
public double GetRating(int userId, int courceId)
    {
        return classifiers.Select(classifier => classifier.GetRating(userId,
courceId)).Average();
    }
public List<Suggestion> GetSuggestions(int userId, int numSuggestions)
    {
        List<Suggestion> suggestions = new List<Suggestion>();
        int numSuggestionsEach = (int)Math.Ceiling((double)numSuggestions /
classifiers.Count);
        foreach (IRecommender classifier in classifiers)
        {
            suggestions.AddRange(classifier.GetSuggestions(userId,
numSuggestionsEach));
        }
        suggestions.Sort((c, n) => n.Rating.CompareTo(c.Rating));
        return suggestions.Take(numSuggestions).ToList();
    }
private List<Suggestion> GetCommonSuggestions(int userId, int numSuggestions)
    {
        int internalSuggestions = 100;
```

```
        List<List<Suggestion>> suggestions = new List<List<Suggestion>>();
        foreach (IRecommender classifier in classifiers)
        {
            suggestions.Add(classifier.GetSuggestions(userId, internalSuggestions));
        }
        List<Tuple<Suggestion, int>> final = new List<Tuple<Suggestion, int>>();
        foreach (List<Suggestion> list in suggestions)
        {
            foreach (Suggestion suggestion in list)
            {
                int existingIndex = final.FindIndex(x => x.Item1.CourceID ==
suggestion.CourceID);
                if (existingIndex >= 0)
                {
                    Suggestion highestRated = final[existingIndex].Item1.Rating >
suggestion.Rating ? final[existingIndex].Item1 : suggestion;
                    final[existingIndex] = new Tuple<Suggestion, int>(highestRated,
final[existingIndex].Item2 + 1);
                }
                else
                {
                    final.Add(new Tuple<Suggestion, int>(suggestion, 1));
                }
            }
        }
        final = final.OrderByDescending(x => x.Item2).ThenByDescending(x =>
x.Item1.Rating).ToList();
        return final.Select(x => x.Item1).Take(numSuggestions).ToList();
    }
```

3.4 Структури даних

У контексті роботи з рекомендаційними системами структура даних може значно залежати від предметної області, для якої створюється дана система.

Система, розроблена у даній кваліфікаційній роботі, дозволяє опрацьовувати дані, що задані у форматі, зображеному на Рис. 34. Замість курсів можуть розглядатися буд-які предмети рекомендацій, взаємодія користувача з якими схожа на взаємодію студента за курсами. Наприклад, це можуть бути статті, новини, тощо. Але важливо зазначити, що у випадках, коли для рекомендаційної системи є важливою актуальність даних (наприклад у випадку з новинами) потрібно реалізувати свою модель Rater, яка буде враховувати дату чи час створення предмету рекомендації у наданні оцінки.

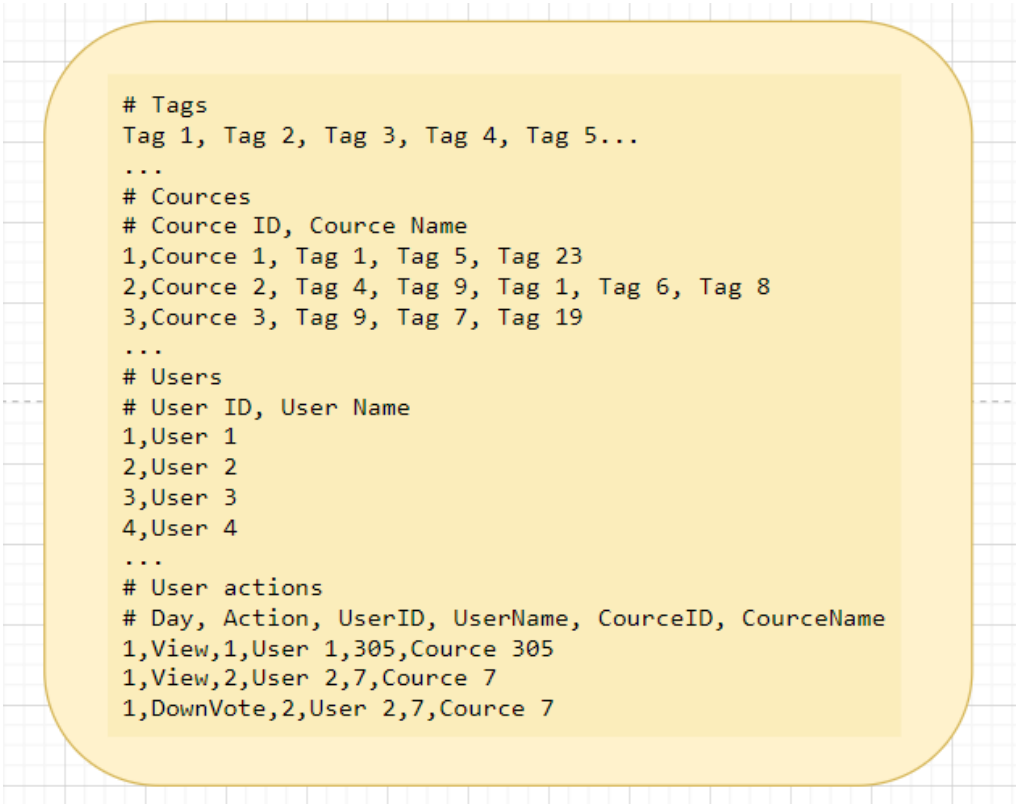

Рис. *34 Структура даних для аналізу рекомендаційною системою*

Детальніше структура даних та відношення між об’єктами, що подаються до рекомендаційної системи зображена на Рис. 35.

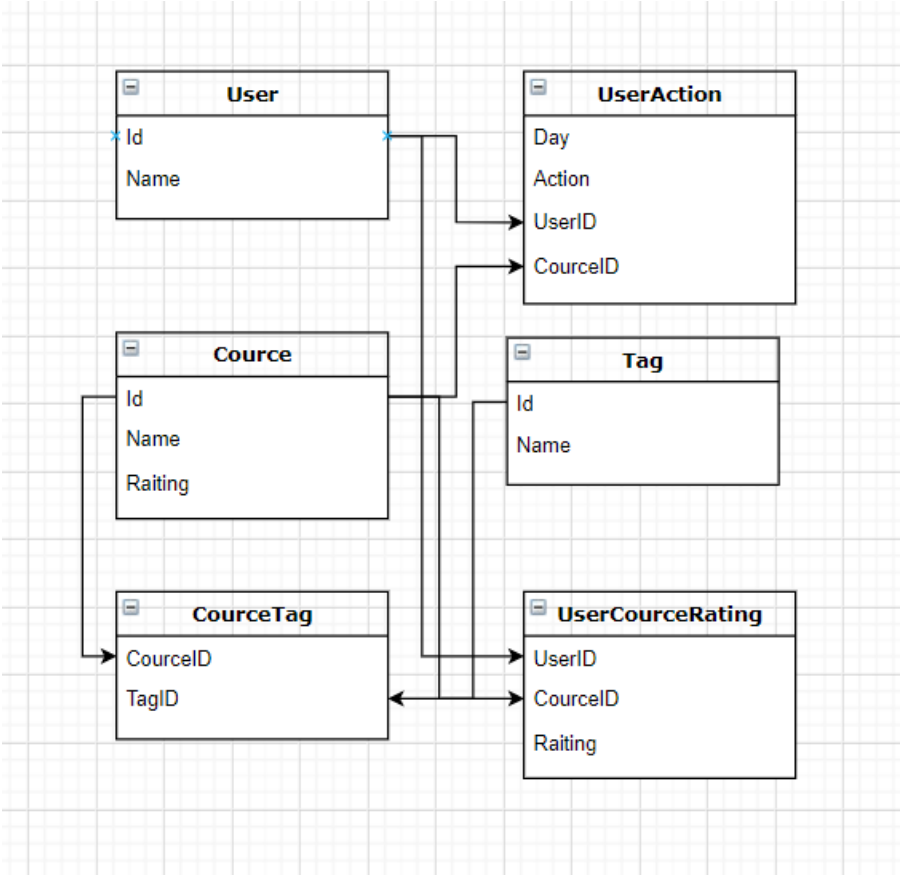

Рис. *35 Структура даних для обробки рекомендаційною системою*

3.5 Вимоги до апаратного забезпечення

Мінімальні вимоги до апаратного забезпечення:

1. Одноядерний процесор з тактовою частотою 1.6 GHz.
2. Оперативна пам’ять ємністю не менше 1 Гб.
3. Монітор з розподільною здатністю 1024x768.

Важливо: мінімальна ємність оперативної пам’яті також дуже залежить від об’єму навчальної та тренувальної вибірки.

Рекомендовані вимоги до апаратного забезпечення:

1. 4-ядерний процесор з тактовою частотою 1.8 GHz.
2. Оперативна пам’ять ємністю 8 Гб.

3. Мінімальні вимоги до програмного забезпечення клієнта.
4. ОС Windows , починаючи з Windows 7 до Windows 10.

Вимоги до користувача десктопного застосунку:

1. Від користувачів вимагається базові навички роботи з десктопними застосунками.

Вимоги до користувача програмного модуля:

1. Базове знання мови C#.
2. Базове розуміння з предмету нормалізації та дискретизації даних.
3. Базове розуміння процесу інтеграції сторонніх програмних модулів у власне програмне забезпечення та взаємодії з ними.

3.6 Опис функціональних можливостей

Оскільки основною метою кваліфікаційної роботи є дослідження стану питання та існуючих алгоритмів та методів вирішення проблеми, а не розробка кінцевого програмного продукту, то функціональні можливості системи орієнтовані на зручність проведення досліджень, а не для використання у якості готової програми для отримання рекомендацій користувачем.

Дана система може бути інтегрована як модуль до складу платформ для електронного навчання або будь-яких інших платформ, де принцип взаємодії користувача з предметами рекомендацій є схожим на взаємодію студента з курсами.

Було реалізовано тестовий застосунок для перегляду результатів роботи алгоритму, до якого можна додати будь-який файл у заданому форматі або заповнити базу даних та перевірити дію алгоритму. Тестовий застосунок здатний прогнозувати рейтинг, який обраний користувач з більшою ймовірністю поставив обраному курсу. Також тестовий застосунок може видавати перелік з N рекомендацій для обраного користувача, розташовуючи рекомендації від найвірогідніших до менш вірогідних.

Взаємодія з програмним інтерфейсом модуля

Розроблений в ході виконання кваліфікаційної роботи програмний модуль може бути підключений до іншого програмного продукту, наприклад до платформи електронної освіти. Існують різні варіанти взаємодії з даним модулем, в залежності від технології, що використовується в платформі, мережевої топології та обмежень, що накладаються на засоби взаємодії між програмами. У даному розділі буде описаний найбільш зручний спосіб взаємодії з програмною бібліотекою для підключення рекомендаційної системи у існуючий програмний продукт.

Взаємодію з бібліотекою в загальному випадку можна описати у трьох кроках:

1. Завантаження даних з CSV файлу для обробки рекомендаційною системою.
2. Обробка даних алгоритмом рекомендаційних систем.
3. Збереження даних про систему.
4. Отримання рекомендацій.

Описані пункти взаємодії можна вмістити у 10 рядків коду (Лістинг 8).

Лістинг 8 Інтеграція з бібліотекою

```
//Define Parser
UserBehaviorDatabaseParser parser = new
UserBehaviorDatabaseParser();

//Load CSV data
UserBehaviorDatabase db =
parser.LoadUserBehaviorDatabase(e.Argument as string);

//Define type of recommendation system
IComparer comparer = new CorrelationUserComparer();
```

```
IRecommender recommender = new
UserCollaborativeFilterRecommender(comparer, 50);

//Train system
recommender.Train(db);

//Predict raiting of cource for User
GetRating args = e.Argument as GetRating;
e.Result = recommender.GetRating(args.UserID,
args.CourceID);

//Get recommendations for the User
GetRecommendation args = e.Argument as GetRecommendation;
e.Result = recommender.GetSuggestions(args.UserID,
suggestionsAmount);

//Save trained model
recommender.Save(savedModel);
```

Окрім цього розроблена система залишається відкритою до модифікацій та розширень. Користувач може реалізувати свій клас парсера, рекомендатора, тощо. Одним з основних і важливих критеріїв була визначена зручність використання бібліотеки на старті. Саме тому комунікація з модулем зводилась до найменшої кількості коду.

3.7 Проект інтерфейсу

Основною метою даної кваліфікаційної роботи є дослідження реалізованих алгоритмів, їх порівняння та дослідження методів вирішення проблеми використання рекомендаційних систем у сфері електронного навчання. Саме

тому для тестування створеної бібліотеки, оцінки зручності взаємодії з нею та для порівняння алгоритмів був створений простий Desktop застосунок.

Після першого запуску застосунку потрібно обрати та завантажити файл з даними для обробки рекомендаційними системами (Рис. 36). Зробити це можна натиснувши на кнопку «Load and Train», а потім обравши файл з даними у діалоговому вікні (Рис. 37).

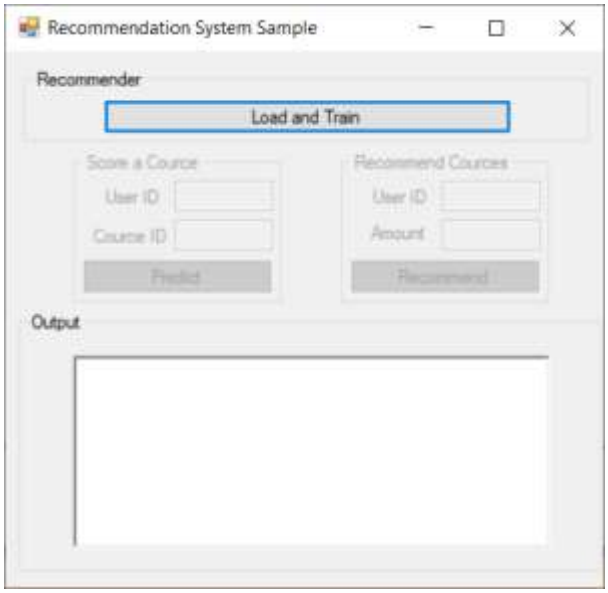

Рис. *36 Перший запуск застосунку*

Рис. *37 Вибір файлу для завантаження та обробки рекомендаційною системою*

Після цього застосунок запускає процес обробки даних гібридною рекомендаційною системою (Рис. 38), яка в свою чергу передає цей процес всім іншим системам, серед яких User-Based Recommender, Item-Based Recommender та Matrix Factorization Recommendation System. Також потрібно звернути увагу, що до складу гібридної рекомендаційної системи входить невеликий відсоток випадкових рекомендацій, що надаються класом Random Recommender.

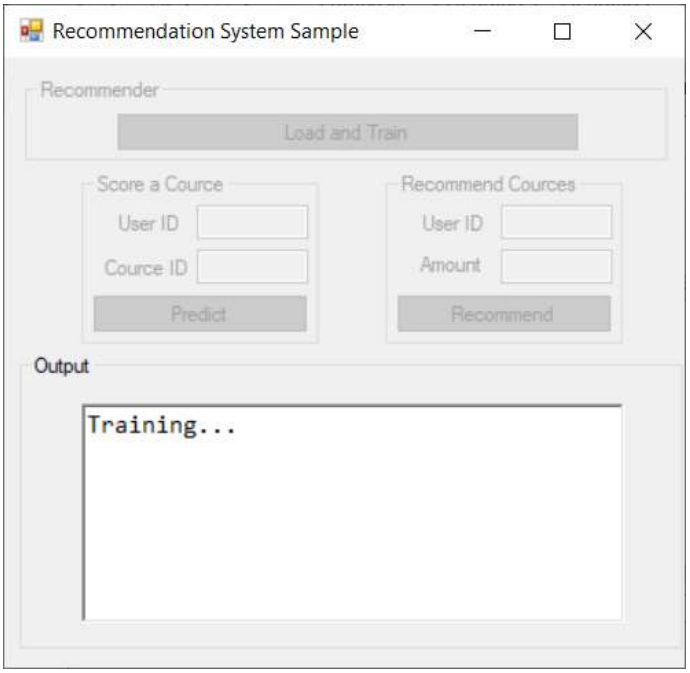

Рис. *38 Процес обробки даних*

Після завершення процесу завантаження даних та їх обробки рекомендаційною системою буде виведена відповідна інформація (Рис. 39) а також система буде збережена до файлу на жорсткому диску (поряд з файлом застосунку за замовчуванням).

На цьому етапі всі дані завантажуються і опрацьовуються в оперативній пам’яті комп’ютера. Одним із способів розширення дослідження та покращення можливостей інтеграції з бібліотекою є оптимізація роботи з пам’яттю та часткове завантаження даних для опрацювання.

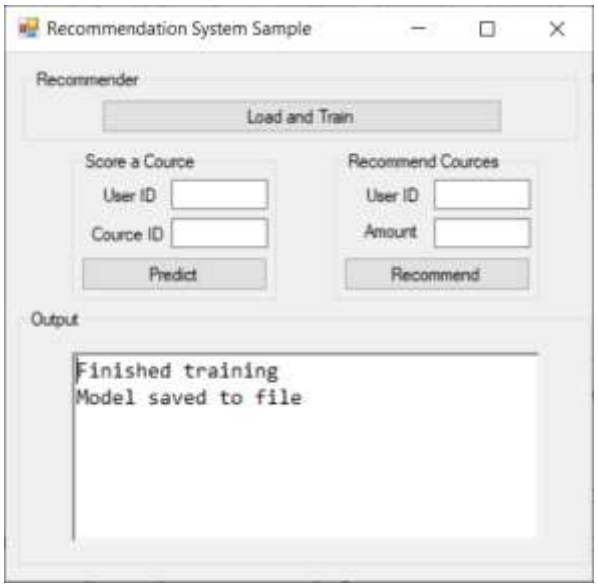

Рис. 39 Завершення процесу обробки CSV файлу даних

Після завершення процесу обробки стає можливим користування рекомендаційною системою. Щоб отримати передбачення рейтингу курсу для користувача потрібно вказати ідентифікатор користувача та ідентифікатор курсу, для якого ми хочемо отримати його потенційний рейтинг та натиснути кнопку «Predict». На Рис.40 відображений результат даної дії.

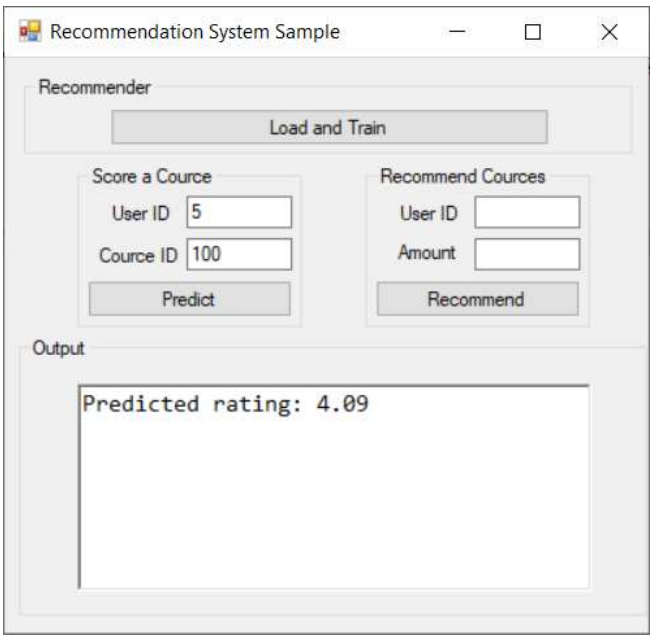

Рис. *40 Обчислення потенційного рейтингу курсу для користувача*

Для отримання переліку рекомендацій для користувача потрібно ввести його ідентифікатор та число рекомендацій, яке потрібно отримати в результаті, і натиснути кнопку «Recommend» (Рис. 41). Після цього система отримає результат від кожної під-системи, відсортує їх та поверне задану кількість.

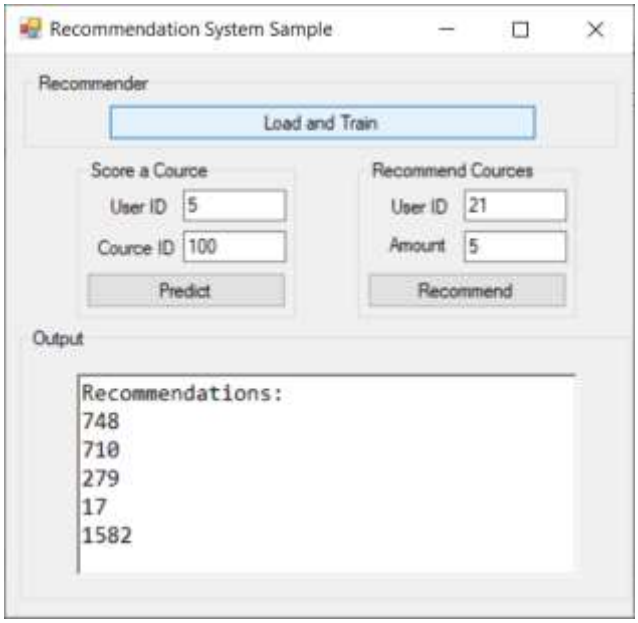

Рис. *41 Отримання переліку рекомендацій для користувача*

Даний інтерфейс створений для тестування і презентації функціональних можливостей, що були реалізовані у модулі рекомендаційної системи. Також на прикладі даного застосунку розробник може ознайомитися з основними функціями модуля, який він може підключити до свого програмного продукту.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДЛЯ ЕЛЕКТРОННОГО НАВЧАННЯ

4.1 Порівняльний аналіз розроблених рекомендаційних систем

Було виконано порівняння роботи чотирьох реалізованих рекомендаційних систем:

1. Коллаборативної фільтрації на основі даних про користувачів.
2. Коллаборативної фільтрації на основі даних про курси.
3. Матричної факторизації.
4. Гібридної рекомендаційної системи (включає у себе частково три види систем з пунктів 1-3 а також заданий відсоток випадкових рекомендацій).

Для експерименту був обраний датасет з ресурсу Kaggle.com, що містить 30 тегів, 3000 елементів (курсів), 3000 користувачів та 105000 дій користувачів над цими курсами (таких як перегляд, оцінка, тощо).

При обробці цих даних в таблиці елементів користувача (або матриці) було близько 9 000 000 комірок (3 000 користувачів по 3000 курсів).

Одна з головних проблем рекомендаційних систем представлена у даній вибірці — це розрідженість (відсутність значень) матриці. Дана таблиця може вмістити 9 мільйонів оцінок, якщо кожен користувач оцінює кожен курс, але насправді наша матриця порожня на 99,2%. Це багато відсутніх даних.

Першим і бажаним кроком у роботі з даними є їх аналіз і розуміння. Тому було створено декілька візуалізацій. І перша — це співвідношення користувачів і кількості курсів, з якими вони взаємодіяли (Рис. 42).

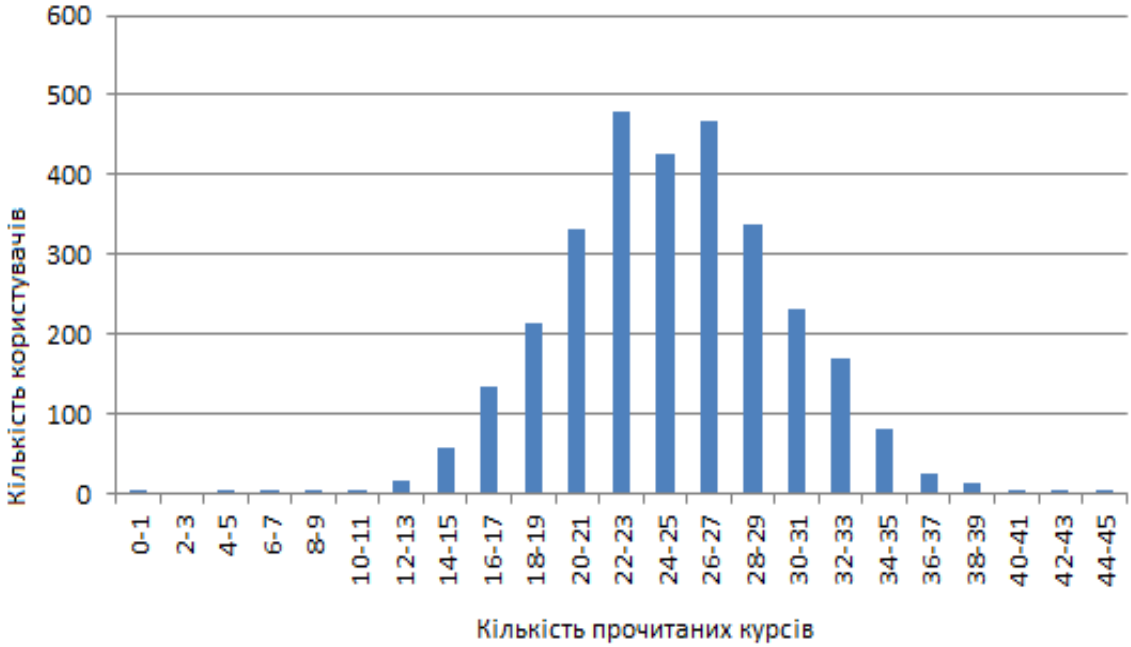

Рис. *42 Кількість користувачів до кількості курсів*

Перш за все, здається, що більшість користувачів взаємодіяли десь з 20-29 курсами у наборі даних.

Якщо поглянути на дані з іншого боку, то стане зрозуміло, що кількість курсів, які прочитали саме N користувачів, є досить однорідною (Рис. 43).

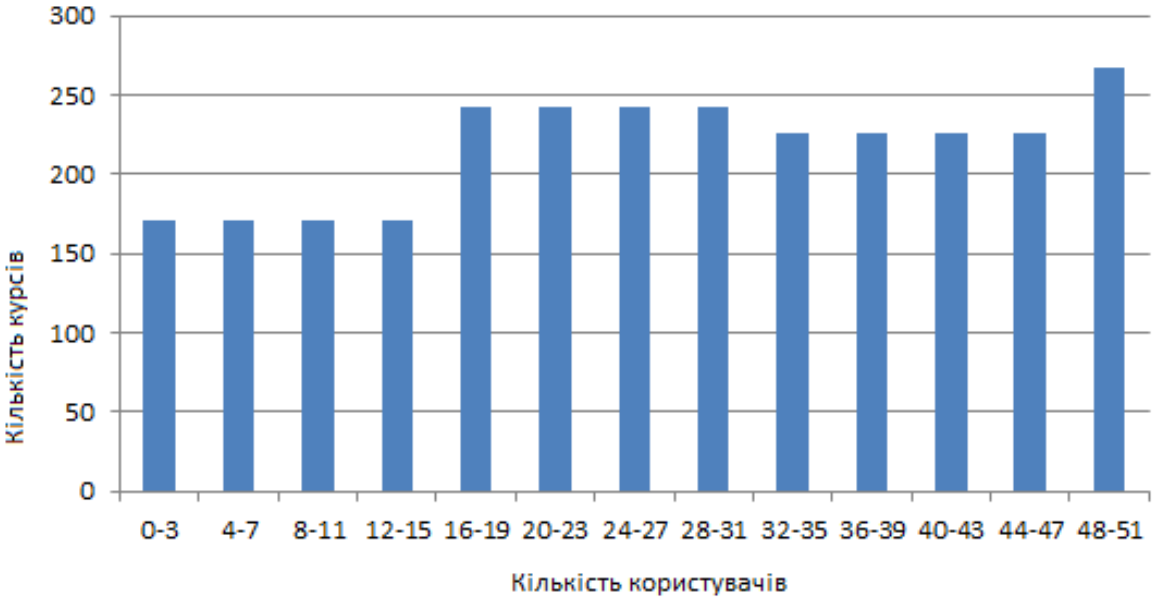

Рис. *43 Кількість курсів до кількості користувачів*

Окрім цього було створене зображення розміром 3000 на 3000 пікселів для представлення кожного рейтингу користувача по кожному елементу (курсу). Чорний піксель означає, що користувач у цьому рядку оцінив курс в цьому стовпці. У версії цього зображення на Рис. 44 чітко видно, що є три темні квадрати.

Рис. *44 Представлення рейтингу кожного користувача до курсу у пікселях*

З певної причини, наші дані свідчать про те, що перші 1000 користувачів зазвичай віддають перевагу першим 1000 курсів, друга тисяча користувачів проходить другу тисячу курсів і так далі для третьої. Ймовірно, це пов’язано із способом формування цього набору даних.

Дані були розділені на два набори. Перший набір містив дані за перші 27 днів і був використаний для навчання. Другий набір містив 3 дні, що залишились, і був використаний для перевірки після закінчення обробки і навчання.

Дані були розділені таким чином, оскільки важливо не використовувати ті самі дані для тестування, що і для навчання. Щоб отримати справжню міру точності нашої рекомендаційної системи, нам потрібні дані, які не використовувались під час навчання.

Наступним кроком був проведений аналіз загальної точності нашої моделі, обчислюючи середньоквадратичну помилку після кожної ітерації. Вона обчислюється шляхом віднімання очікуваного результату від фактичного результату для кожної пари користувач-курс, потім це число зводять до другої степені, додають до загальної суми, а потім беруть квадратний корінь.

Мета будь якого алгоритму, що навчається — мінімізувати функцію помилки під час навчання. Проте врешті-решт модель дійде до точки, коли немає сенсу тренуватися більше. Якщо намагатися видалити кожну останню помилку, є ризик перенавчити модель (де вона запам’ятовує кілька попередніх зразків, а не вивчає загальне правило). Тоді вона буде давати дуже низьку помилку на навчальній вибірці і дуже велику помилку на тестовому датасеті.

На Рис. 45 представлений графік зміни величини помилки протягом часу (зі збільшенням кількості ітерацій навчання).

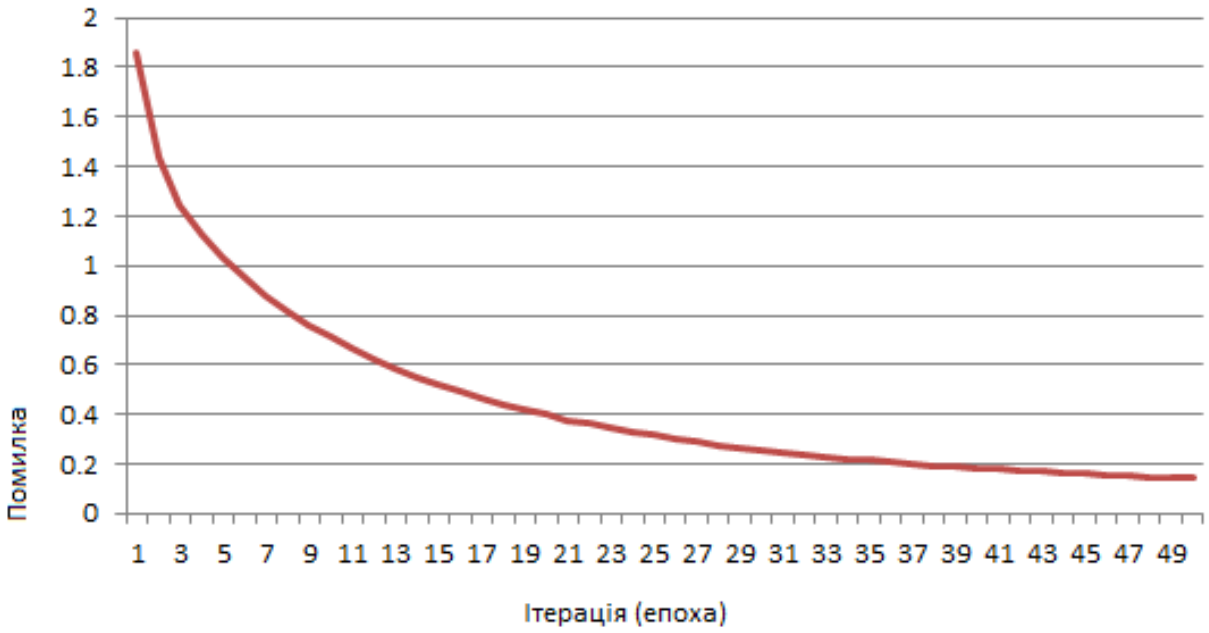

Рис. *45 Зміна величини помилки протягом навчання*

Наступний крок — це перевірка моделей на тестовій вибірці. Були отримані дані (Таблиця 1).

Таблиця 1

Порівняння кількості вдалих рекомендацій на тестовій вибірці

Модель	Відсоток вдалих рекомендацій
User-based collaborative filtering	11%
Item-based collaborative filtering	8%
Matrix factorization	10%
Hybrid recommendation system	8%

Даний результат не є достатньо конкурентним з сучасними системами, але він характерний для перерахованих видів алгоритмів без оптимізацій. Дослідження може бути розширене в напрямку аналізу впливу оптимізацій та змін у вибірці на результат роботи рекомендаційних систем.

4.2 Порівняння реалізованих рекомендаційних систем з системою на ML.NET

В ході аналізу результатів тестування реалізованих систем та пошуку можливостей для розширення дослідження був створений рекомендатор на базі ML.NET.

Рекомендаційна система цього фреймворку базується на алгоритмі матричної факторизації і також має ряд оптимізацій, алгоритми яких не описані у документації до продукту.

Для експерименту був обраний на вже описаний датасет з ресурсу Kaggle.com, що містить 30 тегів, 3000 елементів (курсів), 3000 користувачів та 105000 дій користувачів над цими курсами (таких як перегляд, оцінка, тощо).

Результати цього дослідження показали приріст у кількості вдалих рекомендацій (Таблиця 2).

Таблиця 2

Порівняння результатів алгоритмів з системою на базі ML.NET

Модель	Відсоток вдалих рекомендацій
User-based collaborative filtering	11%
Item-based collaborative filtering	8%
Matrix factorization	10%
Hybrid recommendation system	8%
ML.NET Recommendation system	16%

Приріст у порівнянні з рекомендаційною системою ML.Net (Таблиця 3) обумовлений перш за все оптимізацією алгоритму а також наявністю глибшої і детальнішої попередньої обробки даних.

Таблиця 3

Приріст у результаті в порівнянні з реалізованим алгоритмом

Модель	Відсоток вдалих рекомендацій
User-based collaborative filtering	+45%
Item-based collaborative filtering	+100%
Matrix factorization	+60%
Hybrid recommendation system	+100%

З отриманих результатів можна зробити висновок, що дослідження можна поглиблювати в напрямку покращення попередньої обробки даних, порівняння різних засобів попередньої обробки для заданої предметної області, оптимізації алгоритмів коллаборативної фільтрації та матричної факторизації та додання синтаксичного аналізатору для недетермінованих даних та аналізу текстів (описів курсу, викладачів, студентів, тощо).

ВИСНОВКИ

1. Встановлено актуальність розробки автоматизованої прогнозування / рекомендацій навчальних матеріалів. Шляхом аналізу можливостей області машинного навчання та проведення теоретичних і практичних досліджень було сформовано набір методів для дослідження, проведено їх порівняння та аналіз, а також було визначено стек технологій та інструментів розробки, визначено їх відносну продуктивність, ступінь інтеграції та зручність використання.

2. Проведено порівняльний аналіз розроблених рекомендаційних систем з системою на базі ML.NET. Визначені недоліки, усунення яких може значно покращити систему рекомендацій. Описані можливі способи розширення даного дослідження та покращення результатів.

3. Досліджено сучасні методи використання машинного навчання та статистичних методів для прогнозування вподобань користувачів та рекомендації навчальних матеріалів, визначено їх основні переваги та недоліки та можливі області застосування.

4. Досліджена можливість використання рекомендаційних систем в області онлайн освіти, навчальних платформ. Системи досить актуальні і переважна кількість систем електронного навчання використовує рекомендаційні алгоритми.

5. Основним інструментом для реалізації машинного навчання та рекомендаційних систем обрано мову програмування C# і платформу .NET Core, яка надає можливість використовувати модулі і застосунок на будь-якій платформі.

5. Розроблено програмний застосунок для демонстрації можливостей рекомендаційної системи.

6. Досліджено засоби попередньої обробки даних для покращення машинного навчання та доведено їх вплив на якість отриманої рекомендаційної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space. *CoRR*. P. 81.

2. Pennington J., Socher R., Christopher D. GloVe: Global Vectors for Word Representation. 2014.

3. Open Data Science 2017. Чудесный мир Word Embeddings. URL: https://habr.com/ru/company/ods/blog/329410/ (дата звернення: 12.09.2020).

4. Alammar J. The Illustrated Word2vec. 2017. URL: https://jalammar.github.io/illustrated-word2vec/ (дата звернення: 12.09.2020)..

5. Melville P., Mooney R., Nagarajan R. Content-Boosted Collaborative Filtering for Improved Recommendations. 2002. URL: https://www.aaai.org/Papers/AAAI/2002/AAAI02-029.pdf (дата звернення: 15.09.2020).

6. Анатомия рекомендательных систем. Часть первая. 2018. URL: https://habr.com/ru/company/lanit/blog/420499/ (дата звернення: 15.09.2020).

7. Анатомия рекомендательных систем. Часть вторая. 2018. URL: https://habr.com/ru/company/lanit/blog/421401/ (дата звернення: 16.09.2020).

8. Funk S.. Netflix Update: Try This at Home. 2006. URL: https://sifter.org/~simon/journal/20061211.html (дата звернення: 16.09.2020).

9. Matrix factorization (recommender systems). URL: https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems) (дата звернення: 17.09.2020).

10. Udaya K. E-Learning: Technological Development in Teaching for school kids. *International Journal of Computer Science and Information Technologies*. 2014. P. 6124–6126.

11. Babych M. How To Build E-Learning Platform [From Idea to MVP]. 2020. URL: https://spdload.com/blog/how-to-build-e-learning-website/ (дата звернення: 18.09.2020).

12. Changing Course: Ten Years of Tracking Online Education in the United States. *Babson Survey Research Group*. 2013. P. 4.

13. E-Learning open solutions for inclusive knowledge societies. 2015. URL: https://en.unesco.org/news/e-learning-open-solutions-inclusive-knowledge-societies (дата звернення: 20.09.2020).

14. Навчальний рік 2020-2021. Кабінет Міністрів України. 2020. URL: https://covid19.gov.ua/osvita-ta-batkivstvo (дата звернення: 20.09.2020).

15. Announcing Visual Studio Code. 2015. URL: https://devblogs.microsoft.com/blogs.msdn.com/29 (дата звернення: 25.09.2020).

16. Visual Studio 2019 Platform Targeting and Compatibility. 2019. URL: https://docs.microsoft.com/en-us/visualstudio/releases/2019/compatibility (дата звернення: 28.09.2020).

17. Заметки о выпуске Visual Studio 2019. 2020. URL: https://docs.microsoft.com/ru-ru/visualstudio/releases/2019/release-notes (дата звернення: 01.10.2020).

18. Programming Languages Support. 2020. URL: https://code.visualstudio.com/docs/languages/overview (дата звернення: 02.10.2020).

19. C Sharp. URL: https://ru.wikipedia.org/wiki/C_Sharp#cite_note-MS_CSharp_FAQ-11 (дата звернення: 03.10.2020).

20. Jason R. Building a Simple .NET Compiler. 2018. URL: https://www.codeproject.com/Articles/1227239/Building-a-Simple-NET-Compiler (дата звернення: 08.10.2020).

21. Priyadarshini M. 10 Most Popular Programming Languages In October 2020: Learn To Code. 2020. URL: https://fossbytes.com/most-popular-programming-languages/ (дата звернення: 08.10.2020).

22. .NET documentation. URL: https://docs.microsoft.com/ru-ru/dotnet/fundamentals/ (дата звернення: 08.10.2020).

23. What you should know about .NET Core. 2018. URL: https://www.intelegain.com/dot-net-core/ (дата звернення: 08.10.2020).

24. Richard J. Introducing .NET. 2019. URL: https://devblogs.microsoft.com/dotnet/introducing-net-5/ (дата звернення: 15.10.2020).

25. CSV - Comma Separated Values. 2017. URL: https://datahub.io/docs/data-packages/csv (дата звернення: 15.10.2020).

26. Microsoft Releases ML.NET Open Source Machine Learning Framework Preview. 2018. URL: https://fossbytes.com/microsoft-ml-net-open-source-framework-preview-released/ (дата звернення: 15.10.2020).

27. Machine learning tasks in ML.NET. 2019. URL: https://docs.microsoft.com/en-us/dotnet/machine-learning/resources/tasks (дата звернення: 20.10.2020).

28. McCaffrey J. ML.NET: The Machine Learning Framework for .NET Developers. 2018. URL: https://docs.microsoft.com/en-us/archive/msdn-magazine/2018/connect/machine-learning-ml-net-the-machine-learning-framework-for-net-developers (дата звернення: 15.10.2020).

29. Machine Learning at Microsoft with ML.NET. 2019. URL: https://dl.acm.org/doi/10.1145/3292500.3330667 (дата звернення: 11.11.2020).

30. Пилипенко Анна, студентка магістратури ФЕЕІТ ІІ ЗНУ. Наук. кер.: доц., канд. техн. наук Полякова Н.П. «ЗАСТОСУВАННЯ МЕТОДІВ СТАТИСТИКИ ТА ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ОПТИМІЗАЦІЇ ПРОЦЕСУ ЕЛЕКТРОННОГО НАВЧАННЯ». Збірник наукових праць студентів, аспірантів і молодих вчених «Молода наука-2020» : у 5 т. Запорізький національний університет. Запоріжжя: ЗНУ, 2020. Т.5. С. 94-95.

31. Пилипенко Анна, студентка магістратури ФЕЕІТ ІІ ЗНУ. Наук. кер.: доц., канд. техн. наук Полякова Н.П. «ОПТИМІЗАЦІЯ ВИКОРИСТАННЯ РЕКОММЕНДАЦІЙНИХ СИСТЕМ ТА ШТУЧНОГО ІНТЕЛЕКТУ В СИСТЕМАХ ЕЛЕКТРОННОГО НАВЧАННЯ». Збірник наукових XXV науково-технічної конференції студентів, магістрантів, аспірантів, молодих вчених та викладачів: Запорізький національний університет. Запоріжжя: ЗНУ, 2020. С. 167.

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Пилипенко Анна Василівна, студентка 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти sp115-25@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Застосування статистичних методів та методів штучного інтелекту для оптимізації процесу електронного навчання**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-систем, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2020 Підпис _____ Пилипенко Анна Василівна
(студентка)

Дата 30.11.2020 Підпис _____ Полякова Наталія Петрівна
(науковий керівник)