МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувача кафедри інтелектуальних інформаційних систем, д-р.техн.наук, проф.
_____ Ю. П. Кондратенко
«____»_____2024 року

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА НАДАННЯ РЕКОМЕНДАЦІЙ З ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ

Спеціальність 122 «Комп'ютерні науки»

122 – КРМ – 601.21810110

Виконав студент 6-го курсу, групи 601
_____ О. І. Желтобрюхов
«19» лютого 2024 р.

Керівник: канд. пед. наук, доцент
_____Н. М. Болюбаш
«19» лютого 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р.техн.наук, проф.
_____ Ю. П. Кондратенко
«___» _____ 20 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Желтобрюхову Олегу Ігоровичу

1. Тема кваліфікаційної роботи магістера «Інтелектуальна система надання рекомендацій з перегляду відеоконтенту».

Керівник роботи Болюбаш Надія Миколаївна, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «02» лютого 2024 р. № 21

2. Строк подання студентом роботи 19 лютого 2024 р.

3. Вхідні (початкові) дані до роботи: загальні відомості про підходи до надання рекомендацій у сфері перегляду відеоконтенту, набір даних з рейтинговими оцінками відеофільмів користувачами та характеристиками користувачів і фільмів.

Очікуваний результат роботи: рекомендаційна система для надання персоналізованих рекомендацій з перегляду відеофільмів, реалізована з використанням методів матричної факторизації та інтегрована з чат-ботом, вбудованим у вебзастоснок.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

— дослідження теоретичних засад створення рекомендаційних систем, здійснення аналізу підходів до надання рекомендацій у мережевих онлайн-сервісах з перегляду відеоконтенту;
— обґрунтуваня вибору інструментальних засобів розробки інтелектуальної системи надання рекомендацій;
— розробка та здійснення програмної реалізації системи надання рекомендацій з перегляду відеофільмів, оцінка якості і точності надання рекомендацій.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Охорона праці та безпека у надзвичайних ситуаціях.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	д-р біол. наук, професор Л. І. Григор'єва	
Методична частина	канд. пед. наук, доцент Н. М. Болюбаш	

Керівник роботи __канд. пед. наук, доцент Болюбаш Н. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Желтобрюхов О. І._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «____» _____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи магістера

Тема: «Інтелектуальна система надання рекомендацій з перегляду відеоконтенту»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Визначення керівника і теми КРМ. Подання заяви на затвердження теми КРМ	01.09.2023	19.10.2023	Виконано
2.	Отримання завдання на виконання КРМ	20.10.2023	29.10.2023	Виконано
3.	Складання календарного плану	30.10.2023	5.11.2023	Виконано
4.	Огляд літератури за темою дослідження. Аналіз існуючих підходів до надання рекомендацій у сфері перегляду відеоконтенту, застосуванні методів матричної факторизації	5.11.2023	25.11.2023	Виконано
5.	Проходження переддипломної практики, збір та аналіз матеріалів до КРМ	27.112023	24.12.2023	Виконано
6.	Аналіз предметної області та розробка технічного завдання	25.12.2023	27.12.2023	Виконано
7.	Проєктування та програмна реалізація чат-бота для надання рекомендацій з перегляду відеофільмів, вбудованого у вебзастосунок та інтегрованого із навченою моделлю матричної факторизації FFM	28.12.2023	15.01.2024	Виконано
8.	Робота над розділами фахової частини КРМ	16.01.2024	24.12.2024	Виконано
9.	Розробка методичної частини КРМ та спеціальної частини з охорони праці	25.01.2024	01.02.2024	Виконано
10.	Обговорення отриманих результатів з керівником та попередній захист КРМ	29.01.2024	3.02.2024	Виконано
11.	Корегування роботи за результатами попереднього захисту	4.02.2024	6.02.2024	Виконано
12.	Остаточне оформлення пояснювальної записки та слайдів доповіді до зихисту	7.02.2024	9.02.2024	Виконано
13.	Подання рецензенту та рецензування КРМ	9.02.2024	12.02.2024	Виконано
14.	Подання КРМ, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2024	16.02.2024	Виконано
15.	Захист КРМ перед екзаменаційною комісією (ЕК)	26.02.2024	26.02.2024	Виконано

Розробив студент _____Желтобрюхов О.І._____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи __канд. пед. наук, доцент Болюбаш Н.М._____ _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 2023 р.

АНОТАЦІЯ

до магістерської кваліфікаційної роботи
студента групи 601 ЧНУ ім. Петра Могили

Желтобрюхова Олега Ігоровича

на тему: «**ІНТЕЛЕКТУАЛЬНА СИСТЕМА НАДАННЯ РЕКОМЕНДАЦІЙ З ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ**»

Магістерська кваліфікаційна робота присвячена розробці та здійсненню програмної реалізації інтелектуальної рекомендаційної системи з перегляду відеоконтенту із використанням методів матричної факторизації, інтегрованої з чат-ботом для надання персоналізованих рекомендацій. Що є актуальним в умовах високих темпів інформатизації, оскільки підвищує ефективність надання рекомендацій, допомагає вирішити проблему холодного старту, зменшує складність масштабування.

Об'єкт дослідження – процес пошуку цифрового контенту, який відповідає інтересам та потребам користувачів.

Предмет дослідження – програмні засоби надання рекомендацій та моделі і методи побудови рекомендацій на основі матричної факторизації.

Мета дослідження – підвищення ефективності надання рекомендацій у сфері перегляду відеоконтенту шляхом створення інтелектуальної рекомендаційної системи із використанням методів матричної факторизації.

Магістерська кваліфікаційна робота складається з фахової, методичної і спеціальної частини з охорони праці. Пояснювальна записка фахової частини складається зі вступу, трьох розділів, висновків та додатків. У першому розділі розкрито теоретичні засади використання рекомендаційних систем у сфері перегляду відеоконтенту. У другому розділі обґрунтовано вибір технологій і засобів розробки інтелектуальної рекомендаційної системи. У третьому розділі описано проєктування та програмну реалізацію інтелектуальної системи для надання рекомендацій з перегляду відеофільмів. У методичній частині розроблено лекцію з розкриттям основ матричних факторизаційних моделей. У спеціальній частині розглядаються питання охорони праці та безпеки у надзвичайних ситуаціях.

Магістерська кваліфікаційна робота містить 132 сторінки, 34 рисунка, 5 таблиць, 37 джерел, 6 додатків.

Ключові слова: рекомендаційна система, матрична факторизація, метод декомпозиції сигулярного значення, машинне навчання, чат-бот, машина факторизації з урахуванням поля.

ABSTRACT

to the master's qualification work
by the student of the group 601 of Petro Mohyla Black Sea National University

Zheltobryukhov Oleg Ihorovych

on the subject: «**INTELLIGENT SYSTEM FOR PROVIDING RECOMMENDATIONS FOR VIEWING VIDEO CONTENT**»

The Master's thesis is devoted to the development and implementation of the software implementation of an intelligent recommender system for viewing video content using methods of matrix factorization, integrated with a chatbot for providing personalized recommendations. Which is relevant in conditions of high rates of informatization, as it increases the effectiveness of providing recommendations, helps to solve the problem of cold start, reduces the complexity of scaling.

Object of research – the process of finding digital content that meets the interests and needs of users.

Subject of research – software tools for providing recommendations and models and methods for building recommendations based on matrix factorization.

The purpose of the study is to increasing the efficiency of providing recommendations in the field of viewing video content by creating an intelligent recommendation system using matrix factorization methods.

The master's qualification work consists of a professional, methodical and special part on labor protection. The explanatory note of the professional part consists of an introduction, three sections, conclusions and appendices. The first chapter reveals the theoretical principles of using recommender systems in the field of video content viewing. The second section substantiates the choice of technologies and means of development of the recommender system. The third section describes the design and software implementation of an intelligent system for providing recommendations for watching video films. In the methodological part, a lecture was developed with the disclosure of the basics of matrix factorization models. In a special part, issues of labor protection and safety in emergency situations are considered.

The master’s thesis contains 132 page, 34 figures, 5 tables, 37 sources, 6 appendics.

Key words: recommendation system, matrix factorization, sigular value decomposition method, machine learning, chatbot, field-aware factorization machine.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

CF – Collaborative Filtering

CSS – Cascading Style Sheets

FID – First Contentful Paint

FM – Factorization Machines

FMF – Field-Aware Matrix Factorization

HTML – Hyper Text Markup Language

CLU – Conversational language Understanding

MF – Matrix factorization

ML – Machine Learning

NoSQL – Not only Structured Query Language

ONNX – Open Neural Network Exchange

SVD – Singular Value Decomposition

ВСТУП

Актуальність. Цифрова трансформація сучасного суспільства супроводжується накопиченням великих обсягів відеоінформації, що ускладнює пошук контенту, який відповідає інтересам та потребам користувачів й обумовлює розвиток технологій, пов’язаних із пошуком та наданням персоналізованих рекомендацій. Рекомендаційні системи будують прогнози стосовно інтересів та потреб користувачів, на основі попередньо зібраної інформації про їх поведінку, поведінку інших користувачів із схожими вподобаннями та характеристиками та на основі характеристик елементів відеоконтенту. Однією з проблем такого роду алгоритмів є складність масштабування – для великих систем створення нової рекомендації може займати багато часу. Вирішити цю проблему дозволяє застосування методів генерування рекомендацій на основі факторизації матриць завдяки прискоренню обчислень шляхом зниження розмірності.

Провідні онлайн сервіси доступу до відеоресурсів мають вбудовані рекомендаційні системи, засновані на технологіях пошуку і надання персоналізованих рекомендації стосовно доступу до їх контенту. Однак переважна більшість таких систем націлена на вирішення комерційних задач і не забезпечує ефективну взаємодію користувачів із рекомендаційною системою. Застосування чат-бота, інтегрованого з рекомендаційною системою, дозволяє забезпечити гнучкість такої взаємодії, підвищує ефективність надання персоналізованих рекомендацій та допомагає вирішити проблему холодного старту.

Останнім часом набули широкого поширення підходи, засновані на моделях, які при створенні прогнозу рейтингу елементів відеоконтенту використовують накопичену інформацію для формування моделі та її подальшого навчання. До часто використовуваних моделей такого роду відносять наступні: матрична факторизація (англ. Matrix factorization, MF), машина факторизації (англ. Factorization Machines, FM), нейронна мережа на основі факторизації (англ. Deep Factorization Machines, DeepFM). Більш ефективними є моделі, які при наданні

рекомендацій враховують приховані зв’язки між окремими значеннями характеристик користувачів та елементів відеоконтенту: (англ. Field-Aware Matrix Factorization, FMF), тензорна факторизація парної взаємодії (англ. Pairwise Interaction Tensor Factorization, PITF), машина факторизації з урахуванням поля (англ. Field-Aware Factorization Machine, FFM). Проте ефективність застосування таких моделей у сфері перегляду відеоконтенту є недостатньо дослідженою.

Мета дослідження – підвищення ефективності надання рекомендацій у сфері перегляду відеоконтенту шляхом створення інтелектуальної рекомендаційної системи із використанням методів матричної факторизації.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

1) дослідити теоретичні засади створення рекомендаційних систем та здійснити аналіз підходів до надання рекомендацій у мережевих онлайн-сервісах з перегляду відеоконтенту;

2) обґрунтувати вибір інструментальних засобів розробки інтелектуальної системи надання рекомендацій;

3) розробити та здійснити програмну реалізацію системи надання рекомендацій з перегляду відеофільмів, оцінити якість і точність надання рекомендацій.

Об’єктом дослідження є процес пошуку цифрового контенту, який відповідає інтересам та потребам користувачів.

Предметом дослідження є програмні засоби надання рекомендацій та моделі і методи побудови рекомендацій на основі матричної факторизації.

Методологічною основою дослідження є загальнонаукові аналітичні методи, методи надання рекомендацій на основі матричної факторизації, методи розпізнавання природної мови, методи машинного навчання, методи оцінки точності прогнозу намірів користувача, які дозволили вивчити предмет та об’єкт дослідження, дослідити розвиток науково-методичних засад, напрямів та шляхів

підвищення ефективності надання персоналізованих рекомендацій шляхом інтеграції системи надання рекомендацій з чат-ботом, вбудованим у вебзастосунок.

Наукова новизна одержаних результатів дослідження полягає у тому, що автором: запропоновано та обґрунтовано напрями вдосконалення надання персоналізованих рекомендацій відповідно до потреб та інтересів користувача; одержали подальший розвиток підходи до підвищення ефективності надання рекомендацій користувачам; узагальнено теоретичні засади для надання рекомендацій на основі матричних факторизаційних моделей.

Результати дослідження обговорювалися на XXVI Всеукраїнській науково-практичній конференції «Могилянські читання – 2023: Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» (6-10 листопада 2023 року) та отримали схвалення.

Практичне значення отриманих результатів полягає в тому, що розроблену інтелектуальну систему можна застосувати для надання персоналізованих рекомендацій з перегляду відеоконтенту у різних каналах спілкування: вебзастосунках, месенжерах, соціальних мережах.

Структура магістерської роботи. Відповідно до мети, завдань і предмета дослідження магістерська кваліфікаційна робота містить основну, методичну та спеціальну частини. Основна частина магістерської роботи складається із вступу, чотирьох розділів, висновку, списку використаних джерел та 6 додатків. Загальний обсяг магістерської роботи – 132 сторінки, із них тексту основної частини – 96 сторінок, методичної частини – 12 сторінок, спеціальної – 18 сторінок. Кількість використаних джерел – 37.

1 ТЕОРЕТИЧНІ ЗАСАДИ ВИКОРИСТАННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ У СФЕРІ ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ

1.1 Основні підходи до створення рекомендаційних систем

Накопичення великих обсягів цифрової інформації у різних предметних сферах оточуючої дійсності обумовило появу та стрімкий розвиток рекомендаційних систем. Це такі системи, які дозволяють надавати персоналізовані рекомендації стосовно певних товарів та послуг, оцінюючи переваги користувачів щодо товарів, сервісів, продукції, і роблячи прогноз на основі цієї оцінки.

Основна ідея рекомендаційних систем полягає в тому, що якщо користувачі вибирають схожі товари при покупці в інтернет-магазині, читають однакові статті новин або дивляться одні й ті ж фільми, то велика ймовірність того, що надалі вони будуть поводитися так само схоже. Рекомендаційні системи аналізують товари та послуги двома способами: з точки зору того, наскільки вони подобаються користувачам: за принципом, що людям зі схожими уподобаннями подобаються одні і ті ж продукти, або з точки зору того, наскільки вони популярні серед користувачів, незалежно від подібності переваг окремих покупців. На основі цих двох підходів формуються три основні типи рекомендаційних систем: контента фільтрація, колаборативна фільтрація та гібридна фільтрація [1].

Рекомендаційна система будує рейтинг елементів контенту, товарів та послуг на основі аналізу накопиченої інформації про їх вибір та на певному онлайн-ресурсі та цифрові сліди користувачів при його відвідуванні. До основних методів, які використовують у рекомендаційних системах, відносять наступні [2].

1. Неперсоналізовані рекомендації (англ. non personalized) – формуються без врахування індивідуальних потреб користувачів та вивчення їх попиту.

2. Методи контентної фільтрації (англ. content filtering) – будують систему рекомендацій на основі інформації про поведінку користувача та його потреби й

уподобання.

3. Методи колаборативної фільтрації (англ. collaborative filtering) – будують систему персональних рекомендацій, базуючись на моделі поведінки користувача на основі попередньо зібраної інформації про поведінку інших користувачів із схожими вподобаннями та характеристиками. Серед алгоритмів колаборативної фільтрації виділяють алгоритми, що базуються на даних користувачів та алгоритми, що базуються на даних елементів (товарів чи послуг) [3].

4. Методи гібридної фільтрації (англ. hybrid filtering) – поєднують сильні сторони контентної та колаборативної фільтрації й є більш складними у розробці та підтримці [4].

5. Методи, засновані на знаннях (англ. knowledge filtering) – будують систему персональних рекомендацій, базуючись на знаннях про предметну область (а не про кожен елемент контенту, товар чи послугу). Такий тип рекомендацій має високу точність, пропонуючи користувачеві те, що йому потрібно. Крім цього, система вивчає і аналізує взаємозв'язки між об'єктами, враховує ряд додаткових опцій, що відносяться до індивідуальних властивостей конкретного користувача. Основний мінус – складність розробки та збору даних.

Алгоритми колаборативної фільтрації при формуванні нових рекомендацій базуються на попередньо зібраній інформації про уподобання користувачів, які зберігаються у матриці рейтингів *R*, кожен рядок якої представляє користувача, а кожен стовпець – елемент контенту [3]:

$$R = \begin{vmatrix} r_{11} & \cdots & r_{1n} \\ \cdots & \cdots & \cdots \\ r_{m1} & \cdots & r_{mn} \end{vmatrix}, \qquad (1.1)$$

де r_{ij}– оцінка *j*-го елемента *i*-м користувачем (його рейтинг);

m – кількість користувачів;

n – кількість елементів контенту (товарів, послуг, тощо).

Переважна більшість рекомендаційних систем базуються на великій кількості елементів контенту, значна частина яких не була оцінена користувачами. Це призводить до формування розрідженої матриці рейтингів *R*, що є проблемою для нових рекомендаційних систем та посилює проблему холодного старту, яка полягає у наданні рекомендацій новим користувачам, про яких у системі відсутня інформація. Також у процесі використання рекомендаційної системи збільшується кількість користувачів і з'являється проблема складності масштабування – для великих систем створення нової рекомендації може займати багато часу [4].

Проблему складності масштабування дозволяє вирішити застосування методів генерування рекомендацій на основі факторизації матриць завдяки прискоренню обчислень шляхом зниження розмірності. Застосування чат-боту, інтегрованого з рекомендаційною системою, дозволяє вирішити проблему холодного старту, оскільки у процесі першого спілкування з користувачем рекомендаційна система може виявити його інтереси та переваги.

1.2 Рекомендаційні системи на основі методів матричної факторизації

Матрична факторизація (MF – matrix factorization) передбачає декомпозицію вихідної матриці рейтингів *R* на добуток двох матриць меншого рангу (рис. 1.1):

$$R \approx Q \cdot P^T, \tag{1.2}$$

де *Q* – матриця представлення користувачів;

P – матриця елементів контенту, який необхідно рекомендувати.

Взаємодія користувача з елементом контенту моделюється як скалярний добуток векторів оцінок користувача та елемента у факторному просторі. Необхідно здійснити пошук матриць *Q* та *P*, скалярний добуток яких відповідно до формули 1.2 буде наближеним до існуючої матриці рейтингів *R* [5].

Для декомпозиції матриці рейтингів R можна використати декілька розповсюджених методів: метод головних компонент (англ. Principal Component Analysis, PCA), невід'ємне розкладання матриці (англ. Non-Negative Matrix Factorization, NNMF/NMF), метод декомпозиції сигулярного значення (англ. Singular Value Decomposition, SVD) і його модифікації Funk-SVD, SVD++, Asymmetric SVD, timeSVD++ [6].

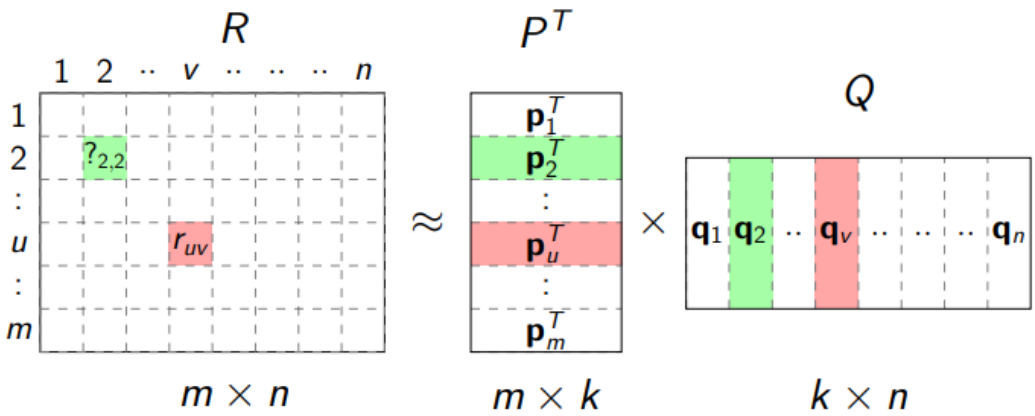

Рисунок 1.1 – Схема факторизації матриці рейтингів

Метод SVD передбачає формування матриці $R^{'}$, у якій відсутні значення матриці рейтингів R заповнено, використовуючи глобальне середньє або середнє значення користувача / елемента. Тоді за теоремою про сингулярний розклад матрицю $R^{'}$ можна розкласти на добуток трьох матриць [7]:

$$R^{'} = U \cdot S \cdot V^{T}, \tag{1.3}$$

де U та V являють собою унітарні матриці розмірністю $m \times m$ та $n \times n$;

S є діагональною матрицею розмірністю $m \times n$ із діагональними елементами, рівними сингулярним значенням матриці R у порядку спадання [3].

Представимо сингулярний розклад в аспекті геометричного перетворення (рис. 1.2). Нехай матриця R' є операцією перетворення. Сингулярний розклад

розбиває одну матрицю перетворення на три: (ортогональна) x (діагональна) x (ортогональна), яку можна розглядати з геометричного аспекту: (обертання) x (розтягування) x (обертання).

Щоб виконати декомпозицію, можна обрати k сингулярних значень, щоб скласти діагональну матрицю S_k і знайти відповідні рядки та стовпці цих k сингулярних значень в U, V як U_k, V_k. Таке перетворення називається урізаним сингулярним розкладом, а розмірність матриці S є рангом факторизації k [7].

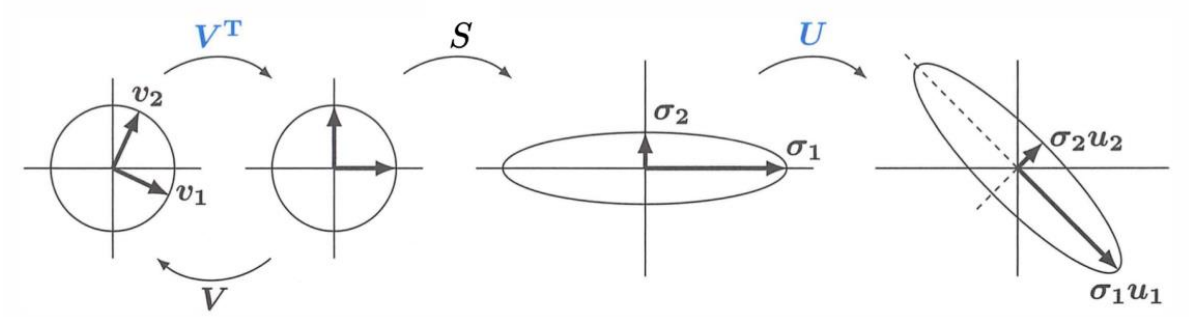

Рисунок 1.2 – Геометричне пояснення сингулярного розкладу

Значення на діагональних елементів матриці S розташовані в порядку спадання, що означає $s_{1,1} \leq s_{2,2} \leq \cdots \leq s_{k,k}$. Ранг факторизації k визначають так, щоб відношення суми квадратів перших r діагональних елементів матриці S було більше за деяке порогове значення, зазвичай у процентах рівне 95%. Окрім цього евристичного методу вибору рангу k, існує емпіричне правило: необхідно обрати k так, щоб сума перших k одиничних значень була принаймні в c разів більша за суму інших сингулярних значень, де c є константою (наприклад, c=10) [4].

Тоді прогнозований рейтинг користувача після декомпозиції можна визначити за формулою [5]:

$$R_k^* = U_k \cdot S_k \cdot {V_k}^T, \tag{1.4}$$

де R_k^* є низькоранговим наближенням матриці R меншої розмірності.

Позначивши $P = (U_k \cdot S_k)^T$ та $Q = {V_k}^T$, декомпозиція матриці рейтингів буде представлена як добуток двох матриць меншої розмірності:

$$R \approx P^T \cdot Q. \tag{1.5}$$

Близькість елементів відеоконтенту, представлених у матриці рейтингів, яка є розрідженою, доцільно розраховувати з використанням косинусної подібності:

$$cos(R_i, R_j) = \frac{R_i \cdot R_j}{\|R_i\| \cdot \|R_j\|}, \tag{1.6}$$

де R_i та R_j - вектори рейтингів *i*-го та *j*-го елеметів (або користувачів);

$\|R_i\|$ і $\|R_j\|$ - норми цих векторів.

Під час розрахунку оцінок подібності враховують лише не пусті значення в матриці рейтингів [3].

Обчислювальну складність традиційного методу SVD можна зменшити, базуючись на машинному навчанні [3]. У моделі матричної факторизації MF матрицю рейтингів розкладають на добуток матриць Q і P таким чином, щоб мінімізувати функцію втрат – відхилення між відомими рейтингами та їх прогнозами шляхом оптимізації цільової функції. Отримана наближена матриця рейтингів для елементів із відомими рейтингами буде містити приблизно рівні їм значення, а для елементів з невідомим рейтингами – розраховані прогнозовані рейтинги *i*-го елемента відеконтенту для *u*-го користувача:

$$\hat{r}_{ui} = \sum_f p_{uf}^T q_{if}, \tag{1.7}$$

де p_{uf} та q_{if} є відповідними елементами матриць Q і P.

Функція втрат у випадку навчання за методом найменших квадратів:

$$\min_{P,Q} \sum_{(u,v\in R)}((r_{u,v} - p_u^T q_v)^2 + \lambda_P \|p_u\|_F^2 + \lambda_Q \|q_v\|_F^2), \quad (1.8)$$

де λ_P і λ_Q – фактори регуляризації, які включають для запобігання перенавчання рекомендаційної системи.

Модель машини факторизації FM притримується базового математичного підходу, комбінуючи факторизацію матриць із регресією для визначення вектору прогнозованого рейтингу $\hat{y}(x)$ [6]:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}\, x_i x_j, \quad (1.9)$$

де w_0 – глобальне зміщення;

n – кількість векторів ознак;

x_i – вектор i-ї ознаки;

w_i – вага вектора x_i;

w_{ij} – вага пари векторів i-ї та j-ї ознак, які утворюють матрицю ваг W, що факторизується.

Під час навчання моделі FM визначають w_0, ваги w_i та факторизовані ваги кожної пари i-ї та j-ї ознак шляхом мінімізації функції втрат.

Набір даних для навчання моделі FM необхідно сформувати у вигляді векторів ознак x_i, перша частина яких кодує користувача, друга – елемент контенту, інші – характеристики елементів та користувачів (рис. 1.3). Такий підхід дає можливість розрахувати рейтинг для нового елементу контенту на основі врахованих його характеристик [6].

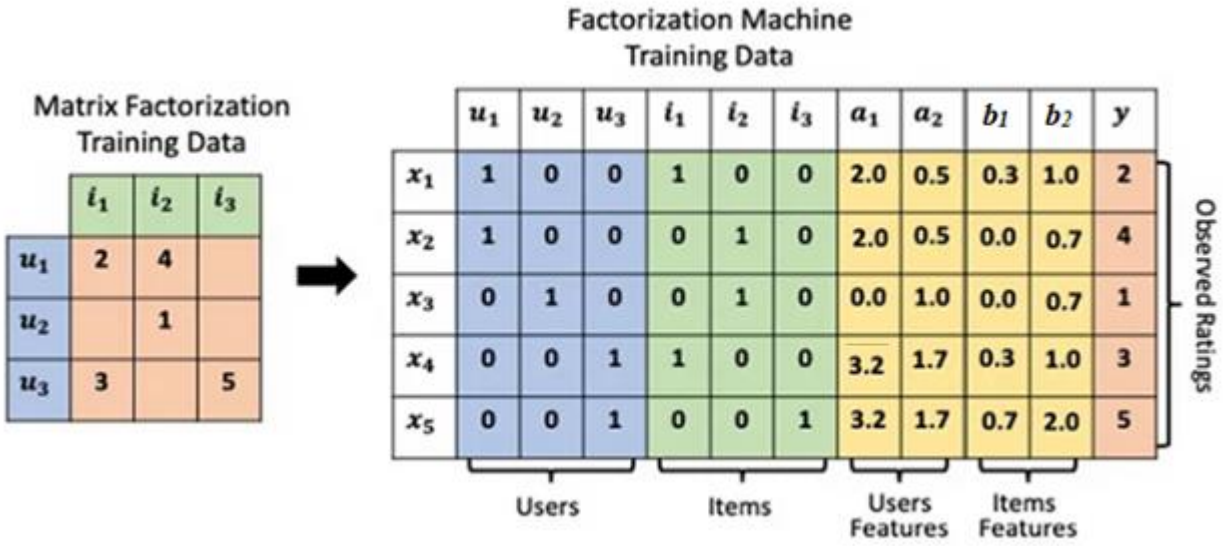

Рисунок 1.3 – Структурування даних в моделі машини факторизації FM

Модель машини факторизації з урахуванням поля FFM є узагальненням та покращенням моделі машини факторизації за рахунок розширення кількості властивостей, на яких базуються рекомендації. При формуванні ознак окремі характеристики користувачів та елементів можуть бути розбиті на поля, які відповідають різним значенням цих характеристик. Вектор прогнозованого рейтингу $\hat{y}(x)$ у моделі FFM визначають за формулою [8]:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_{if(s)}, v_{jf(r)} \rangle x_i x_j, \qquad (1.10)$$

де $f(s)$– поле ознаки x_j;

$f(r)$– поле ознаки x_i;

$v_{if(S)}$ та $v_{jf(r)}$ – факторизовані ваги ознак із урахуванням полів.

Під час навчання моделі FFM визначають w_0, ваги w_i та факторизовані ваги векторів $v_{if(m)}$ із урахуванням виділених полів шляхом мінімізації функції втрат [9].

Для побудови рекомендаційної системи на основі моделей матричної факторизації необхідно дослідити точність прогнозу рейтингу елементів відеоконтенту для вибору більш якісної моделі [10].

З метою оцінки якості матричних факторизаційних моделей та точності переваг та уподобань користувачів при виборі відеофільмів використовують наступні показники [11]:

1) середня абсолютна похибка (англ. Mean Absolute Error, MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \quad (1.11)$$

2) середньоквадратична похибка (англ. Mean Squared Error, MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad (1.12)$$

3) корінь квадратний із середньоквадратичної похибки (англ. Root Mean Squared Error, RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \quad (1.13)$$

де $\hat{y}_i$ і y_i – прогнозоване та фактичне значення рейтингу.

Рекомендації на основі матричних факторизаційних моделей складно робити у режимі реального часу при наявності великої кількості елементів та користувачів, оскільки час навчання моделей буде суттєвим [12]. Тому при створення такі моделі спочатку навчають на підготовленому наборі даних, а потім розробляють рекомендаційну систему на базі навченої моделі [13].

1.3 Надання рекомендацій у мережевих сервісах доступу до відеоконтенту

Рекомендаційні системи використовують в усіх відомих онлайн сервісах для перегляду відеоконтенту: Netlix, Megogo, YouTube, HBO, Amazon Prime, Disney+, Hulu, Окко, Spotify. Зробимо коротких огляд підходів до надання рекомендацій у основних компаніях-гігантах з надання доступу до відеоконтенту.

Онлайн-відеосервіс Netflix застосовує технологію на основі штучного інтелекту і машинного навчання для показу своїм абонентам персоналізованих трейлерів фільмів і серіалів з врахуванням глядацьких уподобань [14]. Netflix аналізує «кожен клік» мільйонів своїх користувачів, прагнучи отримати унікальну деталізацію переваг перегляду та надає рекомендації, розбиваючи їх на групи: особисті рекомендації на основі вподобань користувача, контент, що знаходиться у тренді, рекомендації на основі нещодавно переглянутого фільму/серіалу, новинки кіно (рис. 1.4).

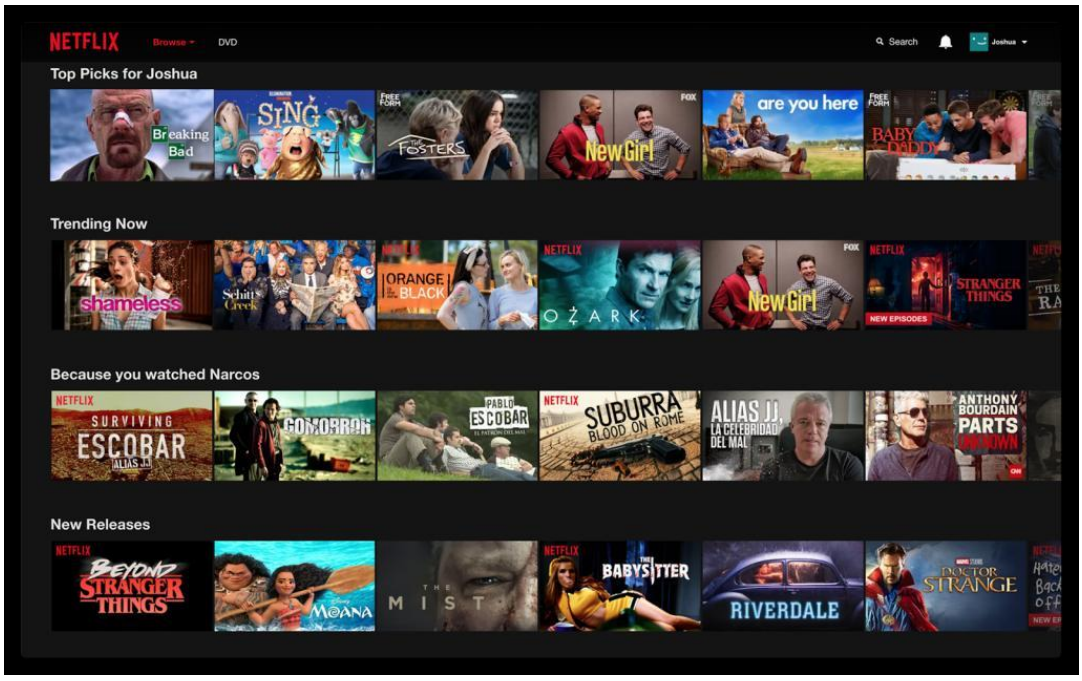

Рисунок 1.4 – Рекомендації Netflix за різними показниками

Саме у цій компанії у рамках змагання по машинному навчанню Netflix Prize у 2006 році було застосовано матричну факторизацію для формування персоналізованих рекомендацій з перегляду відеоконтенту. Еволюція підходів до надання рекомендацій у компанції Netflix відбувалася від прогнозування рейтингу до генерації сторінок та створення індивідуального персоналізованого досвіду продемонстровано на рисунку 1.5. [14]

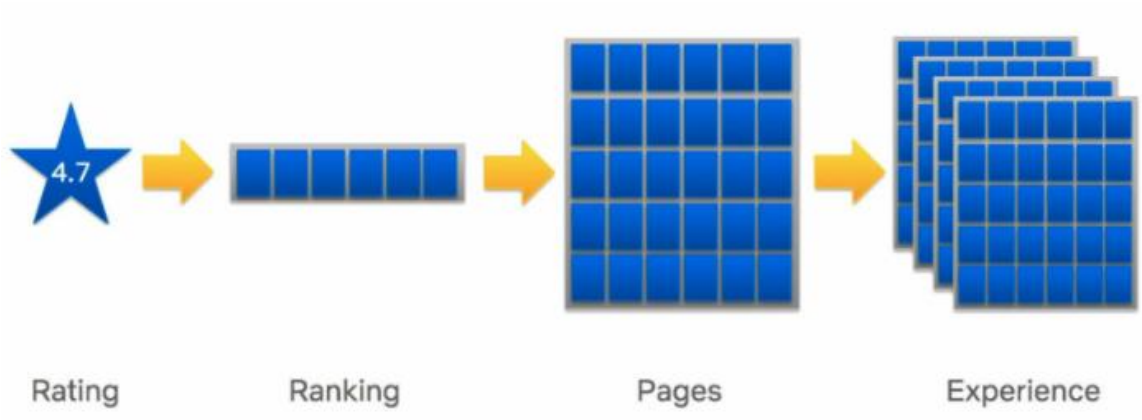

Рисунок 1.5 – Еволюція підходів до надання рекомендацій Netflix

Сьогодні компанія Netflix є глобальним стрімінговим сервісом із більше ніж 180 мільйонами передплатників. Якщо раніше завдання рекомендації було завданням регресії - передбачити рейтинг фільму для користувача / кластера користувачів, то зараз вона перетворилася на безліч завдань – ранжирування, генерації сторінок, максимізації часу стримування для конкретного користувача (тобто змусити користувача провести за переглядом серіалів як можна більше часу). Більшість передплатників переглядають рекомендації, при цьому 80% переглядів всього стримінгу робляться саме на основі отриманих рекомендацій.

Megogo – це гібридний сервіс (OTT/VoD) з можливістю як потокового мовлення ТВ-каналів, так і доступу до відео та аудіо на запит із сумарною аудиторією понад 50 мільйонів унікальних користувачів з 15 країн світу. Компанія почала працювати 13 років тому, коли рівень піратства зашкалював, тому довелося надавати контент безкоштовно, заробляючи на рекламі. Сьогодні MEGOGO

перейшов на підписну модель, вийшов на окупність і став одним із найбільших міжнародних медіасервісів у Східній Європі.

На сайті працює інтелектуальна система рекомендацій на основі матричної факторизації та запам'ятовування місця перегляду (рис. 1.6). Джерелом для формування рекомендацій є внутрішня система трекінгу фактичних переглядів контенту – WatchStat. Персональні рекомендації виробляються для всіх користувачів, які переглянули як мінімум 2 відео.

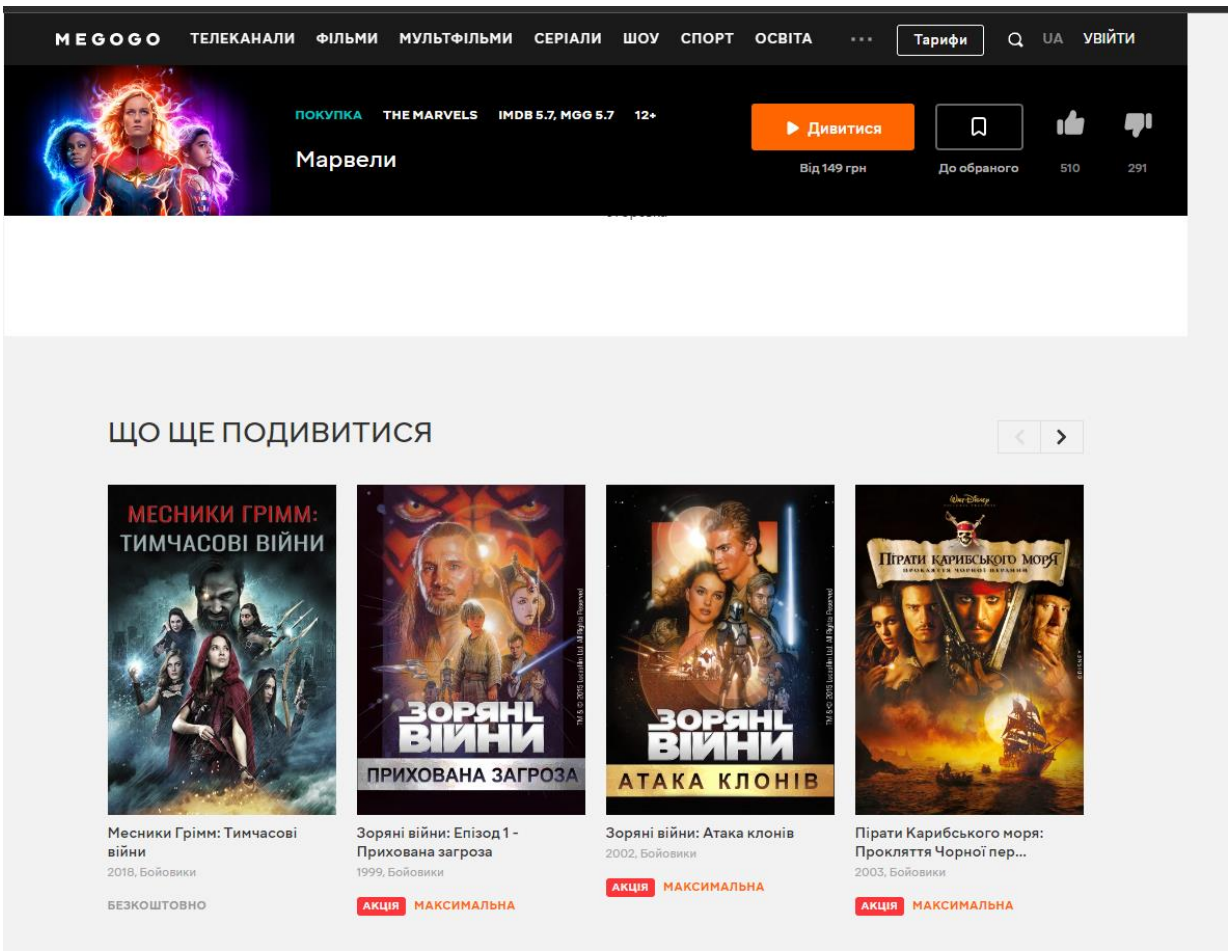

Рисунок 1.6 – Рекомендації Megogo на основі переглянутого фільму

Як заявляють самі розробники системи рекомендацій, останнім часом їм вдалося впровадити нову систему, яка дозволила удвічі збільшити показник клікабельності добірки «рекомендоване» для користувачів – з 14 до 28%. А значить користувачі вважають її набагато більш інформативною, ніж інші системи

рекомендацій, використані до цього. Звичайно, реальну ефективність рекомендацій, створених на основі цього алгоритму, можна визначити тільки дізнавшись чи дійсно користувач переглянув контент після того, як дізнався про нього завдяки системі рекомендації.

YouTube з щохвилинним завантаженням великої кількості відеоконтенту відрізняється однією з найбільш просунутих систем рекомендацій, створеній на основі штучного інтелекту [15]. Вона набагато відрізняється від механізмів, які використовуються на Netflix, Megogo або Spotify, забезпечуючи обробку постійно оновлюваного контенту і формування рекомендацій в режимі реального часу (рис. 1.7). На вхід системи подаються мільйони відеороликів, а на виході вона пропонує ті десятки відео, які потрапляють користувачеві на екран у вкладці з рекомендаціями [15].

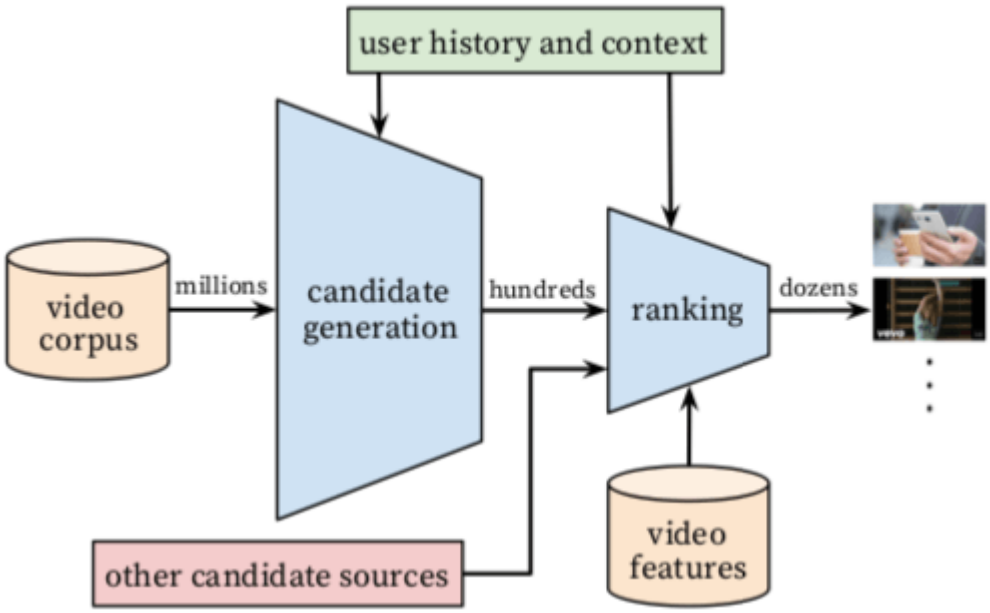

Рисунок 1.7 – Архітектура рекомендаційної системи YouTube

Рекомендаційна система складається з двох згорткових нейронних мереж. Перша є нейронною мережею, яка призначена для генерації кандидатів на основі історії переглядів користувачів YouTube. Це дозволяє забезпечити широку

персоналізацію з подальшою фільтрацією контенту за такими ідентифікаторами, як кількість і час переглядів відео, демографічна інформація та пошукові запити. Нейронна мережа другого рівня серед відібраних відсікає клікбейт та малоцікаві відеоролики з низькою залученістю користувачів. Саме тому відеоролики, які довше дивляться, частіше лайкають та коментують, потрапляють на найперші місця у рекомендаціях, якщо відповідають тематиці, яка цікава користувачеві [15].

Сервіс MovieLens – це рекомендаційна система, яка рекомендує фільми за допомогою колаборативної фільтрації виходячи з оцінок користувачів зі схожими оцінками. Під час реєстрації користувач повинен пройти деяке попереднє анкетування і може отримати рекомендації. При подальшому оцінюванні фільмів, рекомендації уточнюються, але після виставлення рейтингів, щоб отримати новий список фільмів, необхідно оновити сторінку. Користувачеві також відображаються фільми, які він дивився, що може бути незручно. Для більш точних рекомендацій необхідно оцінити достатню кількість фільмів [16].

Здійснений аналіз дозволив установити, що основні сервіси з доступу до відеоконтенту націлені на надання персоналізованих рекомендацій з метою вирішення комерційних задач. Переважна більшість сервісів надає рекомендації на основі платних підписок. Тому є потреба у створенні систем, орієнтованих на інтереси та потреби конкретного користувача, які б забезпечували гнучкість у взаємодії з рекомендаційною системою.

Чат-ботів як компонентів рекомендаційних систем перегляду відеоконтенту не було виявлено. Аналіз рекомендаційних систем у інших сферах дозволив виявити надання рекомендацій з предметів одягу. Наприклад, бренд одягу H&M використовує бот в мобільному застосунку Kik та надає рекомендації образів на основі переваг користувача в його компонентах: кольорі, стилі тощо (рис. 1.8). Однак при цьому не можна отримати рекомендацію тільки на основі попередніх образів, що сподобалися, не уточнюючи їх компоненти.

Аналогічно працює бот у мобільному застосунку Kik для бренду косметики Sephora, який використовується для рекомендації макіяжу, на основі переваг

користувача у стилі макіяжу для очей, губ окремо. Обидва боти обмежені лише використанням у мобільному застосунку, що може створити незручність, якщо користувач захоче в процесі розмови з ботом, подивитися, наприклад, ціну товару, що рекомендується, або його характеристики — для цього йому окремо доведеться відкрити веб-сайт. Розмова у цих ботах побудована на схемі «питання», без використання розпізнавання запитів користувача природною мовою.

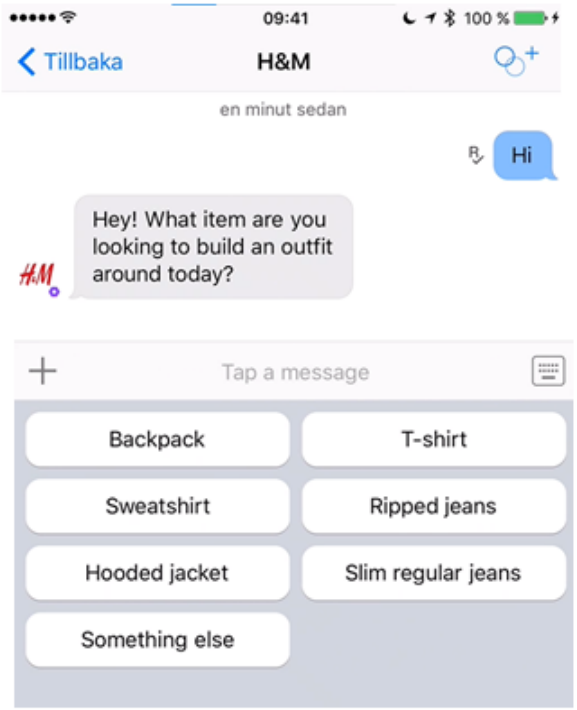

Рисунок 1.8 – Бот-застосунок H&M для підбору зовнішнього вигляду

Здійснений аналіз дозволив установити, що для надання персоналізованих рекомендацій із перегляду відеоконтенту, доцільно рекомендаційну систему, розроблену на основі матричних факторизаційних моделей, інтегрувати чат-ботом, який спрощує взаємодію користувача з системою та вирішує проблему холодного старту.

1.4 Можливості чат-ботів у наданні рекомендацій

У цифровому світі чат-боти стали невід'ємною частиною комунікації інтернет-користувачів з різними онлайн-сервісами та платформами. Ці автоматизовані програми призначені для взаємодії з людьми шляхом розпізнавання природної мови та спілкування у вигляді голосових або текстових повідомлень. Вони можуть мати різні функції від відповіді на запитання та надання рекомендацій до обробки замовлень і підтримки певних тематичних розмов.

Чат-бот є програмою або моделлю штучного інтелекту, призначеною для імітації людської розмови за допомогою текстової чи голосової взаємодії. У міру того, як спілкування з комп'ютерами стає все більш звичним і природним, зростає роль спілкування шляхом діалогу природною мовою. З боку користувача спілкування з ботом більше нагадує взаємодію з людиною чи розумним роботом, ніж роботу з комп'ютером. Таке спілкування може бути представлене як простий діалог з питаннями та негайними відповідями, так і як складна інтелектуальна розмова, в ході якої користувачеві надається доступ до різних сервісів.

При цьому всі дії виконуються за деяким заздалегідь заданим алгоритмом. Ключова відмінність чат-ботів від інших програм полягає у своєрідній імітації дій людей, чи то голос, чи текстові повідомлення, так що користувачеві іноді не відразу вдається зрозуміти, що він спілкується не з людиною, а лише з програмою [17]. Боти можуть бути використані для перенесення простих завдань, таких як збір профільної інформації, на автоматизовані системи, і це більше не вимагатиме безпосередньої участі людини. Людям властиво втомлюватися від монотонних процесів, у своїй втрачається гострота реакції і порушується концентрація на деталях, тоді як бот не втомлюється і виконує самі завдання швидше і точніше. Користувачі можуть спілкуватися з ботом, використовуючи текстові повідомлення, інтерактивні картки та мовлення, при цьому взаємодія з ботом може бути як простим діалогом з питаннями та швидкими відповідями, так і складною

інтелектуальною розмовою, яка дозволить забезпечити доступ користувача до різних послуг або служб.

Чат-боти є вебсервісами і можуть робити те саме, що й інші типи програмного забезпечення: працювати з файлами, використовувати бази даних, підключатися до різних API, розв’язувати обчислювальні задачі. У той же час унікальними їх робить використання механізмів, які зазвичай призначені для спілкування між людьми [17].

У залежності від сфери застосування, функції чат-бота можуть бути різними. Чат-бот як програма, головне завдання якої полягає у веденні діалогу з користувачем, відповідаючи на їхні запитання та задаючи свої, є найпоширенішим типом ботів, який використовується в різних сферах:

– у службах підтримки, допомагаючи вирішити найпростіші питання, наприклад, такі як зміна пароля; для пошуку інформації, наприклад прогнозу погоди, допомоги у виборі подарунка;

– у сфері подорожей, при цьому чат-бот пропонує напрями подорожі, доступні рейси, готелі;

– для допомоги роботодавцям та здобувачам у процесі пошуку роботи та підбору кадрів;

– як персональні помічники і таке інше.

Комерційні боти залучають клієнтів і розсилають пакети інформаційного контенту: новин та акційних пропозицій або актуальних товарів. Шкідливі боти застосовуються для шахрайства з метою нечесного заробітку. Їх також можуть використовувати для координації атак на комп'ютери.

Чат-боти відіграють важливу роль у сучасному бізнесі, допомагаючи підприємствам покращити обслуговування клієнтів, збільшити продуктивність та ефективність роботи. Вони можуть використовуватися для автоматизації клієнтського обслуговування, впровадження програм лояльності, обробки замовлень, моніторингу стану запасів, аналізу даних та багатьох інших завдань. При розв’язанні таких задач чат-боти заощаджують час та зусилля користувачів,

автоматизуючи різні процеси, які їм необхідно здійснити під час використання програми.

У цілому чат-боти можна класифікувати по платформі впровадження, технологіям розробки, способу спілкування з користувачами та функціональності.

У залежності від сфери застосування чат-ботів та задач, які вони вирішують, можна виділити наступні види ботів.

1. *Інтерактивні чат-боти* – призначені для активної взаємодії з користувачами. Вони можуть відповідати на запитання, збирати інформацію від клієнтів, давати рекомендації та вести діалоги. Цей тип чат-ботів часто використовується для створення інтерактивних розважальних розмов та рекламних кампаній.

2. *Інформаційні чат-боти* – полегшують доступ до інформації. Вони можуть надавати користувачам новини, погоду, фінансову інформацію, рецепти та багато іншого. Ці роботи зазвичай не вимагають складної взаємодії, а просто надають корисну інформацію на запит користувача.

3. *Чат-боти для обслуговування клієнтів* – розроблені для покращення якості обслуговування клієнтів. Вони можуть давати відповіді на типові питання, вирішувати проблеми, стежити за статусом замовлень та забезпечувати підтримку 24/7. Ці чат-боти спрощують комунікацію між клієнтами та компанією.

4. *Чат-боти для автоматизації завдань* – призначені для виконання рутинних задач та оптимізації бізнес-процесів. Вони можуть бути використані для автоматичної обробки замовлень, планування зустрічей, нагадування про важливі події, моніторинг запасів та багатьох інших завдань, які можуть бути піддані автоматизації.

Кожен із цих видів чат-ботів має свої власні особливості та можливості, та їх використання залежить від конкретних предметних сфер їх застосування та потреб користувачів.

За функціональністю можна виділити такі види чат-ботів:

– *чат-бот для продажу* – консультує покупців і допомагає їм підібрати

потрібний товар, повідомляє про статус замовлення, розповідає про акції та знижки;

– *лід-бот* – збирає дані відвідувачів сайту, пропонує записатися на демонстрацію продукту;

– *транзакційний бот* виконує різні транзакції: розміщення замовлення, резервування, грошові перекази;

– *бот-інформатор* – відповідає на запити, надає інформацію про варіанти перельотів, ціни, тощо;

– *чат-бот для підтримки* – допомагає у питаннях використання продукту чи послуги.

– *бот-асистент* – інтегрується з іншими платформами і допомагає користувачеві вирішувати відразу кілька завдань, таких як пошук у Google, встановлення нагадування, відбір новин [18].

Таким чином, застосування чат-ботів для цих цілей допомагає покращити бізнес-процеси, зменшити витрати та підвищити задоволеність клієнтів та працівників. Найчастіше чат-боти використовують у продажах, підтримці клієнтів та маркетингу. Чат-боти справляються з рутинними операціями, які можна звести до конкретного алгоритму, шукають та агрегують дані, розповсюджують інформацію. Все це підвищує продуктивність різних команд та покращує клієнтський сервіс [18].

Проведений аналіз можливостей чат-ботів та їх основних сфер застосування показав, що у процесі спілкування з користувачами їх можна засосовувати як один із компонентів рекомендаційної системи. Вони можуть відповідати на прості питання, збирати дані користувачів, допомагати знайти у базі даних чи на сайті потрібну інформацію, у змозі давати персональні рекомендації та консультувати.

Досить зручним є те, що їх можна впровадити у різні канали спілкування – веб-застосунки, месенджери, соціальні мережі. Чат-бот у змозі надавати рекомендації, спілкуючись із користувачем у режимі реального часу. Що є досить

зручним у разі розробки рекомендаційної системи, орієнтованої на задоволення інтересів та потреб користувачів.

Застосування чат-ботів у рекомендаційних системах вимагає створення певної моделі обробки природної мови з метою виявлення намірів корситувача [19]. Для оцінки якості такої моделі та точності розпізнання контексту повідомлень користувача використовують наступні показники:

1) Reccal – кількість правильно розпізнаних повідомлень:

$$Reccal = \frac{TP}{TP+FN}, \quad (1.14)$$

2) Precision – точність правильно розпізнаних повідомлень:

$$Precision = \frac{TP}{FP+TP}, \quad (1.15)$$

3) F1-оцінка – міра точності моделі, яка об’єднує частоту позитивного відклику Recall та точність позитивних результатів Precision:

$$F = \frac{2 \cdot Reccal \cdot Precision}{Reccal+Precision}, \quad (1.16)$$

де *TP* (англ. True Positive): істинно позитивні повідомлення, які належать до певного класу повідомлень і правильно класифіковані;

FP (англ. False Positive): хибно позитивні повідомлення, які віднесені до певного класу повідомлень, але не надежать до нього;

FN (англ. False Negative): хибно негативні повідомлення, які які належать до певного класу повідомлень і неправильно класифіковані як такі, що йому не належить.

1.5 Постановка задачі

Провівши аналіз підходів до надання рекомендацій з перегляду відеоконтенту було зроблено висновок про необхідність розробки інтелектуальної системи, інтегрованої з чат-ботом для надання персоналізованих рекомендацій та забезпечення гнучкості взаємодії користувача з рекомендаційною системою.

Для системи рекомендацій має бути вирішена проблема холодного старту та спрощення складності масштабування. З цією метою було вирішено рекомендаційну систему розробити на основі моделі матричної факторизації. Таким чином, не буде необхідності проводити регулярне трудомістке перенавчання моделі, що є безумовним мінусом багатьох систем рекомендацій. Щоб користувач міг отримувати рекомендації в режимі реального часу і в зручному форматі - за допомогою запитів природною мовою, було вирішено використовувати чат-бот, інтегрований з рекомендаційною системою та вбудований у вебзастосунок. Рекомендації користувачеві будуть надаватися в списку з п’яти фільмів, що найбільш підходять йому.

Для фільмів має бути реалізована можливість оцінки (виставлення рейтингу) – таким чином кожен наступний список рекомендацій буде більш точним. Вже оцінені фільми будуть виключені з наступних рекомендацій, щоб уникнути повторення та рекомендації користувачеві тих самих фільмів. Для нових користувачів повинні надаватися початкові рекомендації – список із двадцяти фільмів, найбільш популярних за середнім рейтингом серед усіх користувачів. Після оцінки деяких з цих фільмів, користувачу будуть надані персоналізовані рекомендації.

Об’єктом дослідження є процес пошуку цифрового контенту, який відповідає інтересам та потребам користувачів.

Предметом дослідження є програмні засоби надання рекомендацій та моделі і методи побудови рекомендацій на основі матричної факторизації.

Мета дослідження – підвищення ефективності надання рекомендацій у сфері перегляду відеоконтенту шляхом створення інтелектуальної рекомендаційної системи із використанням методів матричної факторизації.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

1) дослідити теоретичні засади створення рекомендаційних систем та здійснити аналіз підходів до надання рекомендацій у мережевих онлайн-сервісах з перегляду відеоконтенту;

2) обґрунтувати вибір інструментальних засобів розробки інтелектуальної системи надання рекомендацій;

3) розробити та здійснити програмну реалізацію системи надання рекомендацій з перегляду відеофільмів, оцінити якість і точність надання рекомендацій.

Висновки до розділу 1

Досліджено основні підходи до надання рекомендацій у рекомендаційних системах. Установлено, що застосування методів матричної факторизації полегшує складність масштабування, а використання у рекомендаційній системі чат-бота дозволяє вирішити проблему холодного старту та забезпечує гнучність у взаємодії користувача з системою. Здійснений аналіз мережевих сервісів доступу до відеоконтенту дозволив установити, що при наданні рекомендацій вони націлені на вирішення комерційних задач і не використовують чат-боти для спілкування з користувачами. Обгрунтовано доцільність розробки інтелектуальної системи на основі методів матричної факторизації, інтегрованої з чат-ботом для надання персоналізованих рекомендацій з перегляду відеоконтенту.

Виявлено, що модель матричної факторизації MF спрощує розрахункову складність системи за рахунок декомпозиції вихідної матриці рейтингів на добуток двох матриць меншого рангу. Існує багато гібридних моделей MF, до яких

відносять машини факторизації FM та машини факторизації з урахуванням специфіки поля FFM. Машини факторизації притримуються базового математичного підходу, комбінуючи факторизацію матриць із регресією. Машини факторизації з урахуванням специфіки поля є узагальненням та покращенням машин факторизації за рахунок розширення кількості властивостей, на яких базуються рекомендації.

Виявлення багатьох модифікацій методів матричної факторизації потребує їх подальшого дослідження з метою установлення оптимальних умов їх застосування при розробці рекомендаційних систем для покращення персоналізованих рекомендацій.

2 ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ НАДАННЯ РЕКОМЕНДАЦІЙ

2.1 Бібліотека машинного навчання ML.NET

Системи рекомендацій реалізуються за допомогою алгоритмів, що базуються на машинному навчанні. Раніше розробка моделей машинного навчання була можлива тільки за допомогою таких популярних мов програмування як Python і R [20]. Однак з ряду причин іноді розробники не можуть використовувати ці мови, наприклад, якщо у них є вже готова кодова база іншою мовою або відсутність досвіду роботи з Python або R [21]. У той же час однією з найпопулярніших мов сьогодні є C#, яка використовується для розробки багатьох типів програмного забезпечення. Щоб використати можливості машинного навчання C#, компанія Microsoft реалізувала платформу ML.NET. Це середовище з відкритим вихідним кодом, розроблене для навчання, створення та оцінки моделей машинного навчання, яке дозволяє перенести механізм конвеєра машинного навчання у .NET [22].

Центральним елементом ML.NET є машинне навчання. Модель визначає кроки, необхідні для перетворення вхідного набору даних у прогноз [21]. ML.NET дозволяє навчити нову модель, підібравши потрібний алгоритм, а також імпортувати попередньо навчені моделі. Створену та навчену модель також можна експортувати для інтеграції з іншими проектами та застосунками.

ML.NET підтримується такими операційними системами як Windows, Linux та macOS під час використання .NET Core, а також Windows з використанням .NET Framework. Платформа ML.NET надає кілька основних алгоритмів машинного навчання. Вона може використовуватися для вирішення наступних типів завдань [22]:

– класифікація/категоризація – включає двійкову (бінарну) класифікацію та багатокласову класифікацію. Використовується для прогнозування якого з двох чи

більше класів (категорій) належить об'єкт. Наприклад, для поділу коментарів користувачів на сайті на позитивні та негативні, або визначення породи собаки по фото;

– регресія/прогнозування безперервних значень – використовується для прогнозування значення мітки за набором пов'язаних ознак. Наприклад, для прогнозу ціни певного товару, з його характеристик;

– кластеризація – використовується для угрупування об'єктів, що містять подібні характеристики, у кластери. Наприклад, визначення споживчих сегментів та його демографічних характеристик, щоб виходячи з цього побудувати цільову рекламну кампанію;

– виявлення аномалій – використовується для вирішення таких завдань як виявлення операцій, що потенційно є шахрайськими, або створення навчальних шаблонів, що вказують на те, що сталося мережеве вторгнення;

– системи рекомендацій – використовуються для створення списків продуктів або сервісів, які можуть сподобатися користувачам. Наприклад, для пропозиції продуктів в онлайн магазині, які покупці можуть захотіти купити;

– тимчасові ряди/послідовності – використовуються для створення прогнозів на майбутнє. Наприклад, для прогнозів погоди.

Розглянемо більш детальніше схему процесу побудови моделі в ML.NET та можливості її подальшого використання (рис. 2.1). Процес побудови моделі у ML.NET починається з підбору набору даних, які будуть використовуватись для навчання моделі [23].

Дані надходять на вхід додатку ML.NET – сервісу для навчання моделі. Там набір даних насамперед перетворюється залежно від обраного алгоритму навчання. Далі відбувається навчання моделі та її оцінка. Навчання даних часто є ресурсомістким процесом, тому часто для нього використовуються потужні обчислювальні кластери, що надаються Azure Cloud Services або Amazon Web Services. У результаті на виході цього процесу отримуэмо навчену модель ML.NET.

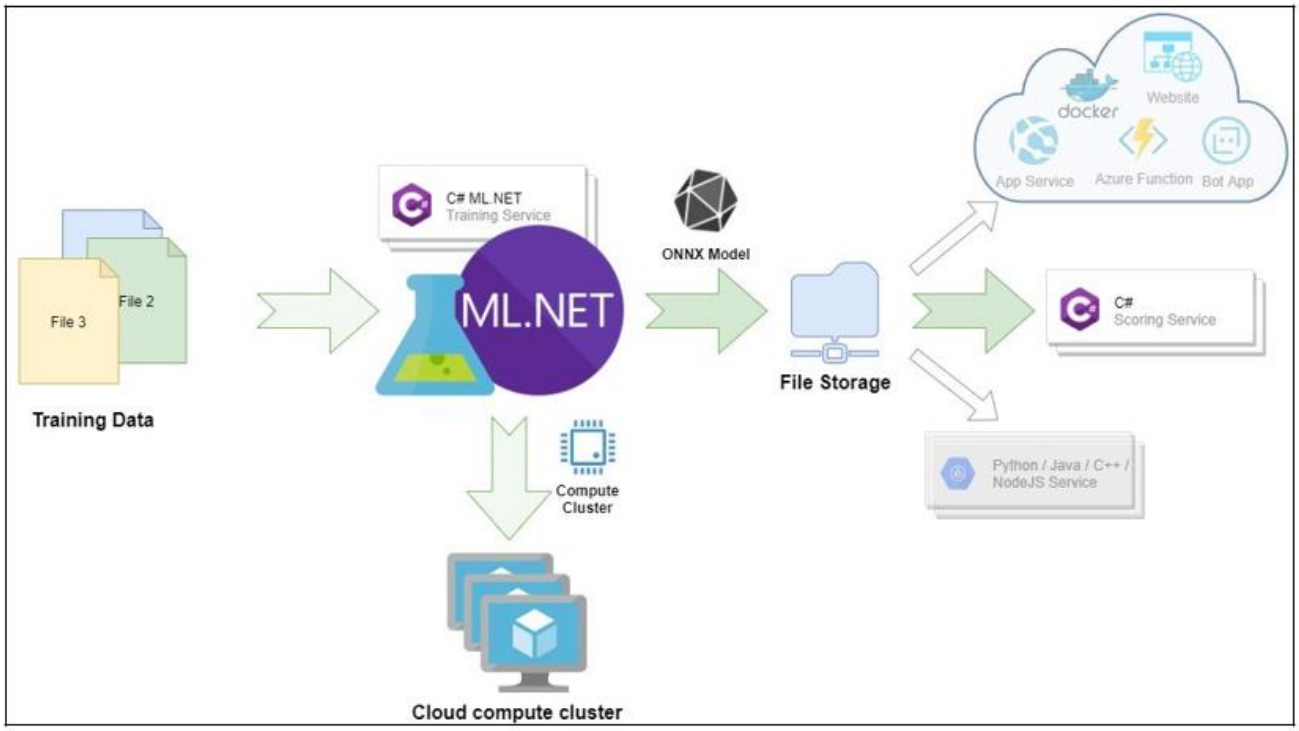

Рисунок 2.1 – Схема побудови моделі в ML.NET та її використання

ML.NET дозволяє імпортувати попередньо навчені моделі (у форматі TensorFlow та ONNX), а також експортувати навчену модель у дані формати. TensorFlow є бібліотекою з відкритим вихідним кодом для машинного навчання, розроблена Google і може бути використана для вирішення завдань побудови та навчання нейронної мережі для класифікації образів [24]. ONNX (англ. Open Neural Network Exchange) – це відкритий та сумісний стандартний формат для представлення моделей машинного навчання (а також глибокого навчання), що дозволяє зберігати навчені моделі, створені на будь-якій платформі, у форматі ONNX та запускати їх на різних платформах, включаючи .NET [25].

ONNX є найпоширенішим форматом моделей машинного навчання. Головна його перевага – підтримка взаємодії між різними платформами. Це означає, що модель можна навчити, використовуючи одну з багатьох популярних платформ для машинного навчання, наприклад PyTorch, перетворити її у формат ONNX і використовувати отриману модель ONNX вже на іншій платформі, наприклад, ML.NET [25]. При цьому ML.NET надає можливість не лише імпортувати навчені

моделі, а й експортувати їх у формат ONNX і потім використовувати у різних сервісах, у тому числі реалізованих на інших платформах. На рисунку 2.1 відображені можливі шляхи використання моделі: у .NET застосунку мовою C#, у різних хмарних сервісах, а також у застосунках іншими мовами програмування (Java, Python і так далі).

Таким чином, модель, навчену в ML.NET, можна використовувати для створення прогнозів на різних пристроях, операційних системах, платформах - там, де підтримується ONNX Runtime – високопродуктивний механізм виведення для моделей машинного навчання в ONNX.

Процес машинного навчання в ML.NET починається з об'єкта типу MLContext, який містить різні каталоги [23]. Каталог – це фабрика для завантаження та збереження даних, функцій перетворення та навчання даних, а також компонентів операцій моделі. Кожен об'єкт каталогу має свої методи створення різних типів цих компонентів. Наприклад, для завантаження, кешування та збереження даних призначений каталог DataOperationsCatalog, для перетворення даних перед навчанням – TransformsCatalog, для збереження, завантаження моделі та створення прогнозів – ModelOperationsCatalog, а алгоритми навчання представлені в декількох каталогах залежно від задачі, що розв’язується (бінарна класифікація в 21 BinaryClass рекомендації в RecommendationCatalog і так далі).

На рисунку 2.2 зображено діаграму діяльності, що відображає процес побудови моделі машинного навчання у ML.NET. Діаграма діяльності – це поведінкова діаграма, яка відображає потік управління від початкової до кінцевої точки програми, показуючи різні шляхи прийняття рішень, які існують під час виконання дії.

Процес розробки моделі в ML.NET представимо у вигляді послідовності кроків:

1) збір та завантаження даних для навчання в об’єкт типу, що реалізує інтерфейс IDataView;

2) розподіл набору даних на навчальний (тренувальний) та тестовий набори у заданому співвідношенні;

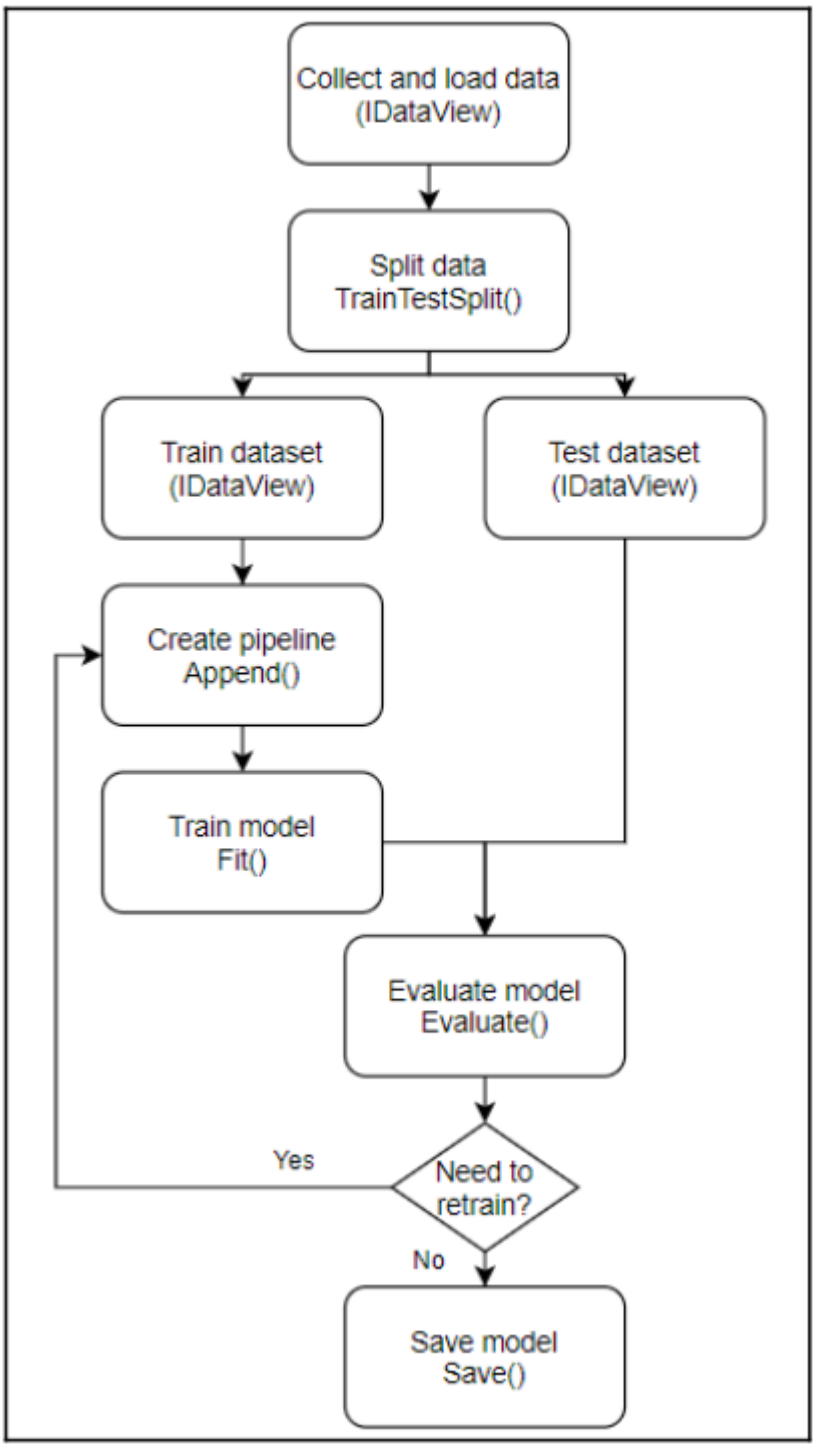

Рисунок 2.2 – Діаграма діяльності процесу навчання моделі в ML.NET

3) створення конвеєра та включення до нього операцій (за допомогою методу Append) для підготовки тренувальних даних до навчання (перетворення у потрібний формат, вибір необхідних ознак, об'єднання стовпців тощо), а також обраного алгоритму машинного навчання;

4) навчання моделі шляхом виклику методу Fit раніше створеного конвеєра;

5) оцінка моделі (метод Evaluate) та повторення попередніх пунктів за потреби;

6) збереження моделі у двійковому форматі для використання в інших програмах (метод Save).

Після збереження модель може бути використана в інших програмах для створення прогнозів. На рисунку 2.3 зображено діаграму діяльності процесу використання навченої моделі.

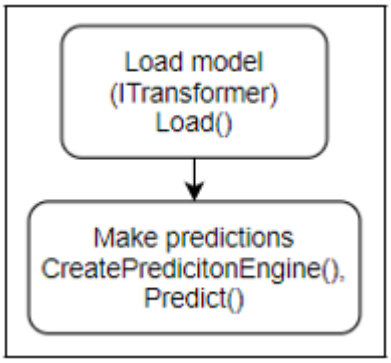

Рисунок 2.3 – Діаграма діяльності процесу використання моделі

Для використання навченої моделі виконуються наступні два етапи.

1. Завантаження моделі в об’єкт типу, що реалізує інтерфейс ITransformer, за допомогою методу Load.

2. Прогнозування, яке включає створення об'єкта типу PredictionEngine за допомогою методу CreatePredictionEngine. PredictionEngine – API, який дозволяє виконувати прогнозування на одному примірнику тестових даних, куди як два універсальні параметри передається тип тестових даних, для яких буде здійснюватися прогноз, і тип самого прогнозу. Далі викликається метод Predict

об'єкта PredictionEngine, куди як аргумент передаються вхідні дані, котрим буде робитися прогнозування.

2.2 Бібліотека Pytorch-Acceleratd

Pytorch-accelerated — це бібліотека, призначена для прискорення процесу навчання моделей Python бібліотеки PyTorch, забезпечуючи мінімальний, але розширюваний навчальний цикл, інкапсульований в одному об’єкті Trainer, який є достатньо гнучким, щоб обробляти більшість випадків використання та здатний використовувати різні параметри апаратного забезпечення без необхідності змін коду [26].

Pytorch-accelerated пропонує спрощений набір функцій і робить величезний акцент на простоті та прозорості, щоб користувачі могли точно зрозуміти, що відбувається усередині, але без необхідності самостійно писати та підтримувати шаблон. Основні характеристики [26]:

– простий і місткий, але легко настроюваний навчальний цикл, який повинен працювати з коробки в простих випадках; поведінку можна налаштувати за допомогою успадкування та/або зворотних викликів;

– обробляє розміщення пристроїв, змішану точність, інтеграцію DeepSpeed, мультиграфічні процесори та розподілене навчання без змін коду;

– використовує чисті компоненти PyTorch без додаткових модифікацій чи обгорток і легко взаємодіє з іншими популярними бібліотеками, такими як timm, transformers і torchmetrics;

– невеликий, оптимізований API гарантує, що існує мінімальна крива навчання для існуючих користувачів PyTorch.

Pytorch-acceleration створено на основі Accelerate Hugging Face, який відповідає за переміщення даних між пристроями та запуск навчальних конфігурацій. Accelerate надає зручні функції для таких операцій, як збір тензорів.

У контексті даної кваліфікаційної роботи бібліотека Pytorch-accelerated є корисною, оскільки має засоби для створення та навчання моделі машини факторизації з урахуванням поля, використовуючи для навчання метод градієнтного спуску. На відміну від інших класифікаторів, які можуть підтримувати лише один стовпець ознак, машина факторизації з урахуванням поля може використовувати кілька стовпців ознак. Кожен стовпець розглядається як контейнер деяких функцій, і такий контейнер називається полем. Усі стовпці функцій мають бути плаваючими векторами, але їхні розміри можуть бути різними. Мотивація поділу функцій на різні поля полягає в незалежному моделюванні функцій із різних розподілів.

2.3 Технології створення ботів Azure Bot Service та Azure Bot Framework

Azure – це повноцінна хмарна платформа, на якій можна розмістити існуючі програми та спростити розробку нових програм. Azure інтегрує хмарні сервіси, необхідні для розробки, тестування, розгортання та керування програмами, використовуючи при цьому всі переваги ефективності хмарних обчислень. Для керування всіма сервісами використовується портал Azure, який дозволяє їх налаштовувати та відстежувати стан ресурсів. Сервіси Azure надають REST API, що дозволяє використовувати їх у будь-яких операційних системах, пристроях та платформах [27].

Створення та управління інтелектуальним чат-ботом для надання рекомендацій реалізовано з використанням Microsoft Bot Framework і Azure Bot Service (рис. 2.4). Таке поєднання технологій дозволяє використовувати бот як інтегрований компонент у різних каналах спілкування: месенджерах, соціальних мережах, вебзастосунках.

Azure Bot Service та Bot Framework надають засоби для збирання, тестування, розгортання та управління інтелектуальними ботами в єдиному середовищі. За допомогою даної платформи можна створювати ботів, які розуміють природну

мову, можуть обробляти питання і давати на них відповіді та багато іншого. Бот – це веб-сервіс, який реалізує інтерфейс для діалогу з користувачем та взаємодіє з Bot Framework Service (один з компонентів Azure Bot Service та Bot Framework) для надсилання та отримання повідомлень та подій [28].

Бот, як і будь-який інший складний застосунок має бути протестований перед публікацією. Для цього є кілька способів:

– тестування бота локально за допомогою емулятора – програми, що надає інтерфейс чату та засоби налагодження, щоб допомогти зрозуміти, як і чому виникає проблема з ботом;

– перевірка бота в Інтернеті: після його розгортання на бот Azure можна протестувати за допомогою каналу Web Chat, доступного на порталі Azure;

– модульне тестування (англ. unit test).

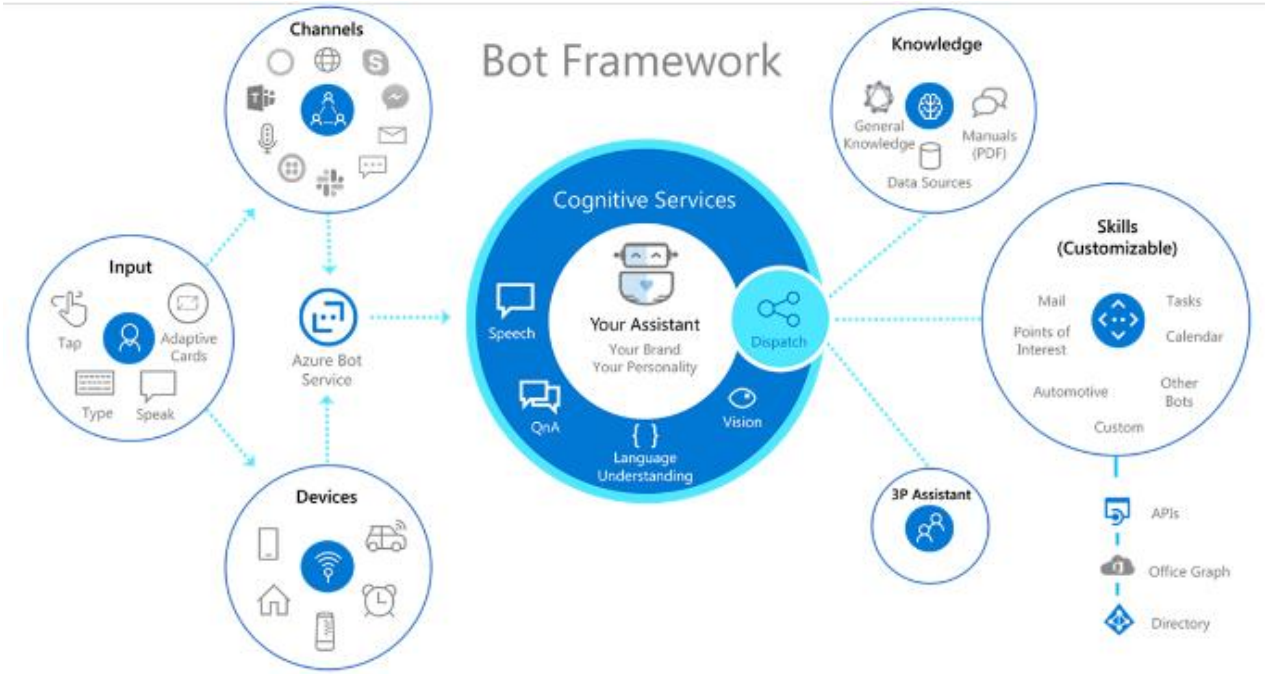

Рисунок 2.4 – Створення бота на основі технології Bot Framework та Azure Bot Service

Коли бот розроблено і протестовано, його можна опублікувати в Azure або у власному веб-сервісі або центрі обробки даних: це перший крок до того, щоб бот міг використовуватися на сайті або всередині каналів для чату. Бот може бути підключений до таких каналів, як Web Chat, Skype, Telegram та інших. Bot Framework виконує велику частину роботи, необхідної для відправлення та отримання повідомлень з усіх цих різних платформ, - бот-застосунок отримує уніфікований, нормалізований потік повідомлень незалежно від кількості та типу каналів, до яких він підключений.

2.4 Azure Cloud Services для обробки природної мови та моніторингу роботи чат-бота

Для розширення функціоналу бота можна використовувати додаткові компоненти, які надаються платформою Azure Cloud Services. Основні сервіси, що використовуються при створенні бот-застосунку, це:

– Azure Monitor – для збирання, аналізу та обробка даних телеметрії з хмарних та локальних середовищ у процесі роботи чат-бота [29];

– API Azure Cognitive Services та Language Understanding – для обробки природної мови та розпізнавання намірів користувача.

Azure Monitor забезпечує максимальну доступність і продуктивність застосунків і сервісів, надаючи функціонал для збору, аналізу та обробки даних телеметрії, що допомагає виявляти проблеми в застосунках та ресурсах, що залежать від них. Всі дані, зібрані Azure Monitor, входять до одного з двох основних типів – метрики та журнали. Метрики – це числові значення, що описують певний аспект системи у певний момент часу. Дані телеметрії, такі як події та трасування, а також дані про продуктивність зберігаються у вигляді журналів, щоб їх можна було поєднати для аналізу [29].

Сервіс Application Insights (компонент Azure Monitor) здійснює контроль доступності, продуктивності та використання як локальних, так і розташованих у

хмарі веб-застосунків. Він використовує потужну платформу аналізу даних в Azure Monitor, щоб надати докладні відомості про операції, що виконуються в програмі, та діагностувати помилки.

Також Azure Monitor надає функціонал для реагування на критичні умови, виявлені в даних, що збираються. Оповіщення в Azure Monitor дозволяють заздалегідь повідомляти адміністратора про критичні стани в застосунку, а також намагаються вжити коригуючих дій.

Azure Cognitive Services – це API та сервіси, призначені для використання під час створення інтелектуальних програм без безпосереднього використання штучного інтелекту. Когнітивні сервіси розділені на різні категорії [30]:

– прийняття рішень – додавання рекомендації до програми для ефективного та обґрунтованого прийняття рішення;

– мова – можливість програми розпізнавати природну мову за допомогою попередньо побудованих сценаріїв, оцінювати наміри користувачів, а також переклад тексту мовами, що підтримуються;

– пошук – додавання API пошуку Bing до програми, щоб мати можливість переглядати безліч веб-сторінок, зображень, відео та новин за допомогою одного виклику API;

– мова – перетворення мови на текст і навпаки, розпізнавання мовлення на основі аудіозаписів;

– зір – розпізнавання та ідентифікація фотографій, відео та рукописного введення.

Cognitive Services, як група сервісів, може не вимагати жодних, деяких або всіх даних користувача для навченої моделі.

Сервіс, який надає повністю навчену модель, можна розглядати як чорну скриньку, тобто користувачеві не треба знати, які дані були використані для навчання. Деякі сервіси дозволяють надати власні дані, а потім навчити модель. Це дозволяє розширити модель, використовуючи дані сервісу та власні дані. Вихід відповідає потребам користувача. Сервіс може вимагати дані для навчання моделі.

Наприклад, в бот-застосунку, що розробляється, використовувався сервіс розпізнавання мови CLU, для якого необхідно надати власні дані для навчання моделі.

Conversational language understanding (CLU) – це хмарний сервіс, який застосовує штучний інтелект до тексту природною мовою для прогнозування загального сенсу фрази та отримання деякої додаткової інформації. Conversational Language Understanding від Microsoft – це комплексний набір інструментів і сервісів, спрямованих на розробку та впровадження розмовно-орієнтованих інтерфейсів та застосунків [31]. Ці рішення дозволяють розробникам створювати різноманітні застосунки, такі як чат-боти, віртуальні асистенти та інші системи, які взаємодіють із користувачем природною мовою для виконання деяких завдань.

Одним з ключових компонентів Conversational Language Understanding від Microsoft є Azure Cognitive Services, які надають набір інтелектуальних сервісів для аналізу, розпізнавання та розуміння мовлення. Сервіси включають в себе розпізнавання мовлення, розуміння мови, генерацію мовлення, аналіз намірів та інші.

Ключовими функціями Conversational Language Understanding від Microsoft є наступні [31]:

– розпізнавання та перетворення мовлення на текст за допомогою розпізнавання мовлення;

– аналіз контексту та змісту повідомлень для визначення намірів користувача за допомогою розуміння мови;

– створення відповідей та взаємодія з користувачем на основі розпізнаного тексту за допомогою генерації мовлення.

З використанням цих інструментів та сервісів розробники можуть створювати рішення, які дозволяють користувачам спілкуватися з додатками та системами у природній мові, забезпечуючи більш зручну та ефективну взаємодію.

Створюючи проект CLU, розробники можуть ітеративно позначати висловлювання, тренувати та оцінювати продуктивність моделі, перш ніж зробити

її доступною для використання. Для розробки моделі розуміння природної мови, базовими є три важливих поняття: висловлювання, намір і сутність.

Висловлювання відноситься до окремого, чіткого твердження або повідомлення, зробленого користувачем природною мовою. Воно представляє те, що користувач говорить або друкує під час взаємодії з чат-ботом або будь-якою системою обробки природної мови. Намір представляє основну мету або мету висловлювання користувача. Він відповідає на запитання: «Чого користувач хоче досягти?» У контексті чат-ботів і розуміння природної мови виявлення намірів користувача має вирішальне значення для надання релевантних і відповідних контексту відповідей. Сутність – це певна частина інформації або даних у висловлюванні, яка відповідає намірам користувача. Сутності надають контекст і деталі про запит користувача. Вони відповідають на такі запитання, як «що», «де», «коли», «скільки» або «який». Ці концепції надзвичайно корисні для досягнення цінних результатів за допомогою CLU.

2.5 NoSQL-сховище Azure Table storage

Для роботи рекомендаційної системи необхідним є сховище даних, яке буде зберігати дані, накопичувані у процесі роботи цієї системи.

Сховище таблиць Azure – це сервіс, який дозволяє зберігати нереляційні структуровані дані, або структуровані дані NoSQL, у хмарному сховищі ключів/значень без попередньо визначеної структури бази даних [32].

Завдяки тому, що табличне сховище не має схеми, дані можна легко адаптувати у міру розвитку потреб програми. Головною перевагою табличного сховища перед традиційною базою даних SQL для аналогічних обсягів даних є швидкість доступу та економічність ресурсів.

Azure Table storage дозволяє зберігати різні гнучкі набори даних, наприклад, дані користувача, та іншу необхідну для програми інформацію. Даний сервіс дозволяє зберігати будь-яку кількість сутностей у таблиці, а обліковий запис Azure

дозволяє створювати будь-яку кількість таблиць.

Сутність – це набір властивостей, подібних до рядка бази даних, а властивості – пари ключ-значення [32]. Кожна сутність має три системні властивості: ключ розділу, ключ рядка і мітку часу. Ключ розділу потрібен для того, щоб об'єкти, що належать одному розділу, можна було вставляти/оновлювати в атомарній операції та вимагати для читання швидше. Ключ рядка об'єкта – це його унікальний ідентифікатор у розділі. Таблиця являє собою набір сутностей, при цьому за рахунок гнучкості сховища в одній таблиці можна зберігати різні сутності, з різним набором властивостей.

Сховище таблиць Azure відмінно підходить для наборів даних, які не вимагають зовнішніх ключів, складних з'єднань з іншими таблицями, процедур та можуть бути денормалізовані для швидкого доступу до потрібної інформації.

Отримати доступ до даних можна за допомогою протоколу OData та запитів LINQ за допомогою бібліотек .NET.

Таблиці масштабуються зі збільшенням набору даних, тому Azure Table storage добре підходить для зберігання навіть величезної кількості інформації.

2.6 Засоби програмування дизайну: HTML, CSS

Чат-бот вбудовано у вебзастосунок, що реалізує інтерфейс для відправки та отримання повідомлень і подій. для розробки вебінтерфейсу було використано HTML та CSS.

HTML – (англ. hyper text markup language), стандартизована мова гіпертекстової розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора [33].

CSS (Cascading Style Sheets) – мова опису стилів, що застосовується для визначення вигляду та оформлення веб-сторінок. Вона розділяє представлення веб-

сторінки (визначене за допомогою HTML) від її структури, дозволяючи змінювати зовнішній вигляд без впливу на вміст. CSS включає такі принципи [34]:

Селектори: Вказують, які елементи HTML або групи елементів стилізуються. Селектори можуть бути елементами HTML, класами, ідентифікаторами, атрибутами та іншими.

Властивості: Визначають конкретні аспекти вигляду елементів, такі як колір тексту, розмір шрифту, відступи тощо.

Значення: Конкретні значення, які надаються властивостям, наприклад, "червоний" або "12px".

Спадковість: Властивості можуть успадковуватися від батьківських елементів до дочірніх, що дозволяє застосовувати стилі до багатьох елементів одночасно.

Пріоритет: Визначає, як вирішуються конфлікти між різними правилами CSS, застосованими до одного елемента.

Каскадінг: Механізм, який визначає порядок застосування стилів, коли вони конфліктують або перекриваються.

CSS дозволяє розробникам ефективно керувати виглядом веб-сторінок, полегшуючи їх підтримку та оновлення. Він використовується для створення привабливих та користувацьких інтерфейсів для веб-сайтів.

Висновки до розділу 2

Для створення моделей машинного навчання MF та FM було обрано платформу ML.NET, яка надає середовище для навчання, тестування й оцінки моделі та дозволяє використовувати її у застосунках не тільки платформи .NET, але й іншими мовами, а також у хмарних сервісах. Створення та навчання моделі FM реалізовано з використанням бібіотеки Pytorch-Acceleratd.

Створення та управління інтелектуальним чат-ботом для надання рекомендацій реалізовано з використанням Microsoft Bot Framework і Azure Bot

Service. Таке поєднання технологій дозволяє використовувати бот як інтегрований компонент у різних каналах спілкування: месенджерах, соціальних мережах, вебзастосунках. Обробка природної мови та розпізнавання намірів користувача здійснювалося із використанням хмарних API Azure Cognitive Services та Conversational language understanding. Сервіс Azure Monitor було застосовано з метою моніторингу й аналізу даних у процесі роботи чат-бота. Для зберігання даних використано NoSQL-сховище Azure Table storage, завдяки чому було реалізовано масштабування у відповідності з потребами. У даному дослідженні чат-бот було інтегровано у середовище вебзастосунку, для розробки якого використано мову розмітки HTML та CSS як засіб стилізації.

3 СТВОРЕННЯ ТА НАВЧАННЯ МОДЕЛЕЙ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

3.1 Попередня обробка на підготовка набору даних до навчання

У якості набору даних для навчання моделі машинного навчання для системи рекомендацій фільмів використовувався набір рейтингових даних про фільми з веб-сайту MovieLens, які були зібрані та надані дослідницькою лабораторією GroupLens.

Набір даних про фільми містить двадцять сім мільйонів рейтингів за 5-бальною шкалою, які представляють собою оцінку 58.000 фільмів 280.000 користувачами. Дані були зібрані в період з січня 1995 року по вересень 2021 року (https://grouplens.org/datasets/movielens/25m/).

Набір даних містить файли з даними – links.csv, movies.csv, ratings.csv, tags.csv, users.dat. Для навчання моделі використовувалися дані з файлів movies.csv, ratings.csv та users.dat. У набір даних входять фільми, в яких є хоча б один тег або оцінка користувача. Структура файлу movies.csv (57980 рядків х 3 колонки) продемонстровано на рисунку 3.1.

Уся інформація про рейтинги міститься у файлі ratings.csv (27007502 рядків х 4 колонки). Файл також містить заголовок, а далі дані у файлі є оцінкою певного фільму деяким користувачем (рис. 3.2).

Таким чином інформація про кожну оцінку включає ідентифікатор користувача, який оцінив фільм, представлений також ідентифікатором, власне рейтинг фільму, оцінений у п’ятибальній шкалі, та мітку часу – кількість секунд з півночі 1 січня 1970 року за всесвітнім координованим часом.

movieId	title	genres
208297	"BLOW THE NIGHT!" Let's S...	Documentary|Drama
51372	"Great Performances" Cats ...	Musical
136604	#1 Cheerleader Camp (2010)	Comedy|Drama
183901	#Captured (2017)	Horror
195955	#Female Pleasure (2018)	Documentary
203417	#FollowMe (2019)	Horror|Thriller
151789	#Horror (2015)	Drama|Horror|Mystery|Thriller
179057	#Lucky Number (2015)	Comedy
177545	#realityhigh (2017)	Comedy
190861	#SCREAMERS (2016)	Horror
201174	#SquadGoals (2018)	Drama|Thriller
205461	#Stuck (2014)	Comedy|Drama|Romance

Рисунок 3.1 – Структура файлу movies.csv

userId	movieId	rating	timestamp
1	296	5.0	1147880044
1	306	3.5	1147868817
1	307	5.0	1147868828
1	665	5.0	1147878820
1	899	3.5	1147868510
1	1088	4.0	1147868495
1	1175	3.5	1147868826
1	1217	3.5	1147878326
1	1237	5.0	1147868839
1	1250	4.0	1147868414
1	1260	3.5	1147877857

Рисунок 3.2 – Структура файлу ratings.csv

Список користувачів та їх характеристик, таких як стать та вік, знаходиться у файлі users.dat – 281233 рядків х 3 колонки (рис. 3.3). Список фільмів та їх характеристики містяться у файлі movies.csv. До характеристик фільмів відноситься його назва, рік виходу фільму та жанр (рис. 3.4). У наборі даних передбачено 18 жанрів: дія (англ. Action), пригоди (англ. Adventure), анімація (англ.

Animation), дитячий (англ. Children's), комедія (англ. Comedy), злочинність (англ. Crime), документальний фільм (англ. Documentary), драма (англ. Drama), фантазія (англ. Fantasy), фільм-нуар (англ. Film-Noir), жах (англ. Horror), музичний (англ. Musical), містичний фільм (англ. Mystery), романтика (англ. Romance), наукова фантастика (англ. Sci-Fi), трилер (англ. Thriller), війна (англ. War), західний (англ. Western).

user_id	sex	age_group
1	F	1
2	M	56
3	M	25
4	M	45
5	M	25

Рисунок 3.3 – Структура файлу users.dat

Для створення на навчання матрчних факторизаційних моделей дані з трьох файлів: ratings.csv, movies.csv та users.dat, завантажуються і об'єднуються в один набір даних (рис. 3.4).

user_id	sex	age_group	movie_id	rating	unix_timestamp
1	M	25	1193	5	978300760
1	M	25	661	3	978302109
1	M	25	914	3	978301968
1	M	25	3408	4	978300275
1	M	25	2355	5	978824291

Рисунок 3.4 – Data Set, сформований для навчання

У підготовленому наборі даних було здійснено виявлення порожніх або невикористаних стовпців, які було вилучено. Виявлено та вилучено дублікати, здійснено перевірку некоректно введених даних [35]. Задля відсіювання маловідомих фільмів було вилучено інформацію про фільми, які оцінило менше, ніж 15 користувачів.

Деякі фільми було віднесено до декількох жанрів, що потребувало здійснення денормалізації жанру: кожен рядок, що містив список жанрів, було розбито на кілька рядків, тож у кожному рядку стає по одному жанру (рис. 3.5). Дата створення фільму містилаля у його назві у дужках. Для виділення дати виходу фільму як його характеристики було здійснено вилучені дати з назви кожного фільму.

Після виконання описаних вище методів остаточна кількість рейтингових даних становить 30872062 рядків із 26897455 унікальною назвою фільмів і 274952 унікальним користувачем.

userId	movieId	rating	title	genres
4	10140	4.0	Nut Job (2014)	Adventure
4	10140	4.0	Nut Job (2014)	Animation
4	10140	4.0	Nut Job (2014)	Children
4	10140	4.0	Nut Job (2014)	Comedy
4	10140	4.0	Nut Job (2014)	Fantasy

Рисунок 3.5 – Денормалізація жанру

Для створення та навчання моделі матричної факторизації MF з набору даних було використано ідентифікатори користувачів і фільмів та виставлені рейтинги фільмів. Під час створення та навчання моделі машини факторизації FM, було враховано характеристики користувачів: їх стать та вік. Характеристику віку було

дискретизовано з інтервалом у 5 років. Для фільмів було враховано як характеристики їх назви, рік створення та жанр.

Для створення та навчання моделі матричної факторизації з урахуванням поля FFM характеристики користувачів було розбито на поля: стать на поля – жіноча стаь та чоловіча стать, вік – вікові проміжки з періодом у 5 років. Для фільмів характеристику жанр було розбито на поля, що відповідають кожному жанру.

3.2 Навчання та оцінка точності матричних факторизаційних моделей

До навчаючої множини відбиралося 80% об’єктів набору даних, до тестової – 20%. Розподіл на навчаючу та тестову множини здійснювався за допомогою вбудованої функції TrainTestSplit, її використання відображено у функції LoadData класу DataProcessor (додаток А).

По замовчуванню, коли дані обробляються, вони передаються потоком відкладеним чином, тобто алгоритми навчання можуть завантажувати дані і проходити кілька разів під час навчання. Тому для наборів даних рекомендується робити кешування, при цьому дані розміщуються в пам’яті, що зменшує кількість звернень до накопичувача для завантаження даних. Кешування даних навчаючої множини проводиться відразу після поділу даних на тренувальний та тестовий набори та здійснюється за допомогою вбудованої функції Cache.

У машинному навчанні стовпці, які використовуються для прогнозування, називаються ознаками, а стовпець з повернутим прогнозом називається міткою. У цьому випадку ідентифікатор, стать і вік користувача та ідентифікатор, назва і жанр фільму – ознаки, а рейтинг – мітка. Алгоритми машинного навчання ML.NET вимагають, щоб вхідні дані були представлені у числовій шкалі та у одному числовому діапазоні. Тому наступним етапом підготовки даних до навчання є отримання даних у форматі, який очікується моделлю ML.NET.

Для категоріальних даних, представлених у номінальній шкалі – жанру фільмів, здійснено їх бінарізацію – dummy-кодування. Яке передбачає представлення даних у вигляді бінарних векторів довжиною, яка дорівнює кількості жанрів та одиницею у позиції, індивідуальній для кожного жанру. Описані вище перетворення були реалізовані за допомогою вбудованої функції ML.NET OneHotEncoding для прямого унітарного кодування та CustomMapping - довільного відображення даних, що визначається користувача класом IsRecommendedCustomAction, для перетворення мітки до типу bool.

Навчання моделей MF та FFM здійснювалося з використанням бібліотеки ML.NET(C#). Однак ML.NET не надає можливості навчання та тестування моделей машини факторизації, тому модель FM було навчено на мові програмування Python із використанням бібліотеки Pytorch-Acceleratd. Код наведено у додатку Б.

З метою оцінки якості матричних факторизаційних моделей та точності переваг та уподобань користувачів при виборі відеофільмів було використано наступні показники: середня абсолютна похибка MAE (формула 1.11), середньоквадратична похибка MSE (формула 1.12) та корінь квадратний із середньоквадратичної похибки RMSE (формула 1.13).

Під час навчання кількість епох для кожної моделі була рівна 30 з застосуванням методу early stopping.. Результати начання моделей представлено у таблиці 3.1.

Таблиця 3.1 – Результати оцінки точності прогнозу рейтингу

№ з/п	Модель	Показники оцінки			Час навчання, сек	Кількість епох
		MAE	MSE	RMSE		
1.	MF	1,01	1,72	1,31	7200	12
2.	FM	0,87	1,7	1,30	7200	15
3.	FFM	0,83	1,62	1,27	7200	13

Проведене дослідження дозволило установити, що кращу точність прогнозу мала модель машини факторизації з ураховунням поля FFM. Тому для створення рекомендаційної системи з надання рекомендацій по перегляду відеофільмів було обрано модель FFM.

3.3 Навчання та оцінка якості моделі обробки природної мови

У даній роботі взаємодія користувача з рекомендаційною системою реалізована з використанням чат-боту, для якого необхідно створити модель обробки природної мови.

Здатність розуміти, що користувач хоче сказати і який намір вкладає у свою фразу, забезпечує природніше відчуття розмови з ботом. Сервіс CLU дозволяє зробити так, щоб бот міг розуміти контекст повідомлень користувача, використовувати більше природної мови при діалозі з користувачем і краще спрямовувати розмову. Для використання CLU було виконано наступні кроки:

– натиснути кнопку New app для створення нової програми CLU;

– обрати вкладку Manage, розділ Versions для створеної програми;

– натиснути кнопку Import і вибрати зі списку Import as JSON;

– обрати раніше підготовлений файл з намірами користувачів RecommendationCLU.json, розташований у папці CognitiveModels проекту;

– навчити чат-бот як реалізацію моделі обробки природної мови;

– опублікувати застосунок у робочому середовищі;

Файл RecommendationCLU.json містить чотири можливі наміри користувача: GetRecommendations – отримати рекомендації, GetRecommendationsByGenre – отримати рекомендації за конкретним жанром, Evaluate – оцінити роботу бот-застосунку, Help – отримати допомогу у вигляді настанови від бота, Cancel – скасувати поточну дію та None – нічого. Наміри використовують для того, щоб зрозуміти, що має на увазі і що хоче отримати користувач, коли відправляє повідомлення боту.

Скріншот з порталу CLU для пунктів 1-3 наведено на риснку 3.6.

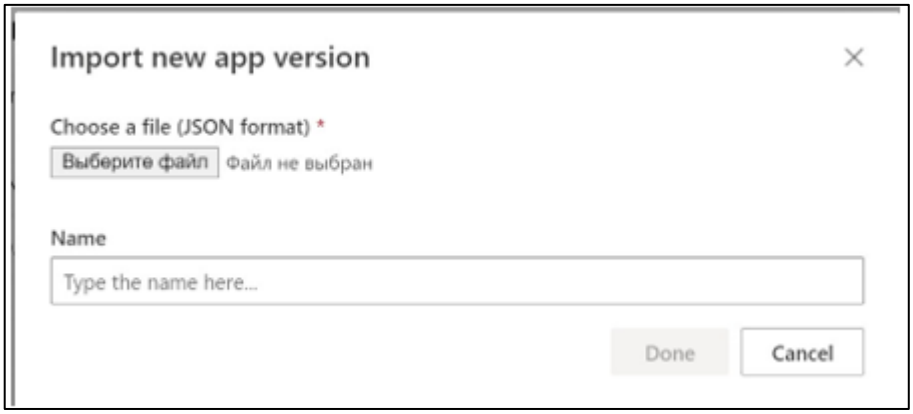

Рисунок 3.6 – Імпортування нового застосунку на портал CLU

Користувацька класифікація текстів підтримує проєкти двох типів:

– класифікація за однією міткою – кожному документу з набору даних можна призначити лише один клас: наприклад, сценарій фільму можна класифікувати лише як "Романтичний фільм" або "Комедія";

– класифікація за декількома мітками – кожному документу з набору даних можна призначити кілька класів: наприклад, сценарій фільму можна класифікувати як "Комедія" або "Романтичний фільм" та "Комедія".

Створення проекту класифікації користувацького тексту зазвичай включає декілька етапів, зображених на рисунку 3.7. У рамках даної роботи для побудови та тестування моделі розпізнавання мови було виконано такі кроки:

– визначення схеми: було визначно класи, між якими розрізняються запити користувача, щоб уникнути неоднозначності, наприклад, клас «GetRecommendations» відповідає безпосередньо за виклик функції розрахунку рекомендацій для поточного користувача;

– маркування даних: документам, які стосуються одного класу, потрібно призначити той самий клас;

– навчання моделі: починається з вивчення промаркованих даних та внесення діалгових конструкцій;

– оцінка якості моделі;

– розгортання моделі: розгорнута модель стає доступною для використання за допомогою API;

– класифікація текстів: використання навченої моделі для завдань класифікації текстів з метою обробки природної мови.

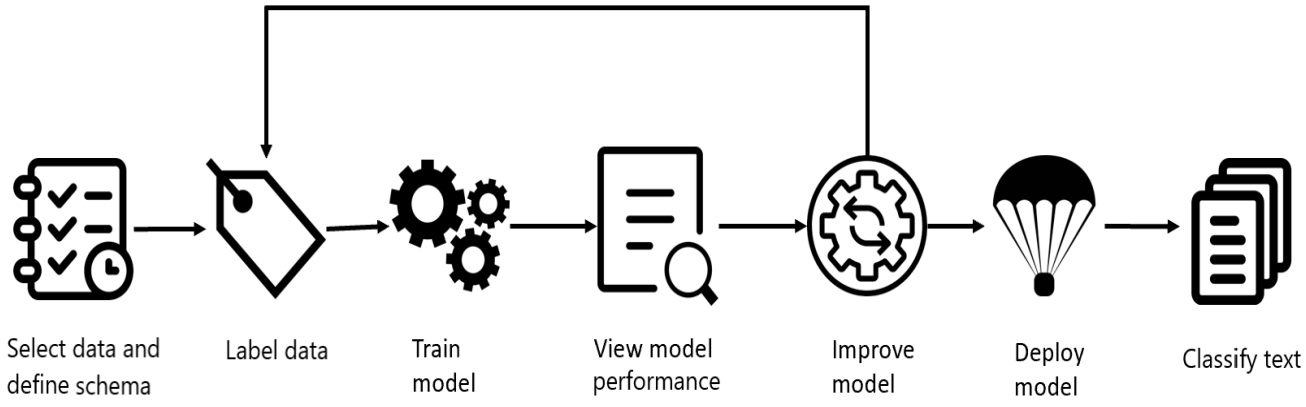

Рисунок 3.7 – Схема створення та тестування моделі розпізнавання мови

Діалоги призначені для того, щоб виконувати певні завдання у визначеному порядку. Порядок виконання діалогів дозволяє спрямовувати розмову з користувачем. Діалоги можна викликати у різний спосіб: у відповідь на повідомлення користувача або деякі зовнішні події, а також з інших діалогів. Бібліотека діалогів надає ряд вбудованих функцій: основні це каскадні діалоги і діалоги введення. Каскадні діалоги об’єднують кілька кроків у послідовність, що дозволяє боту передавати інформацію, отриману на кожному етапі, на наступний крок. Більш детально реалізовані каскадні діалоги буде описано у наступному параграфі.

Для оцінки якості моделі розпізванавання мови та намірів користувача було використано наступні метрики: Reccal – кількість правильно розпізнаних повідомлень (формула 1.14), Precision – точність правильно розпізнаних

повідомлень (формула 1.15) та F1-score – міра точності моделі, яка об’єднує частоту позитивного відклику Recall та точність позитивних результатів Precision (формула 1.16). Результати оцінки якості побудованої мовної моделі наведено на рисунку 3.8. Вони свідчать про високу точність розпізнавання мови та намірів користувача при спілкування з чат-ботом – 99,17%.

За результатами проведеного дослідження точність моделі розпізнавання мови є високою.

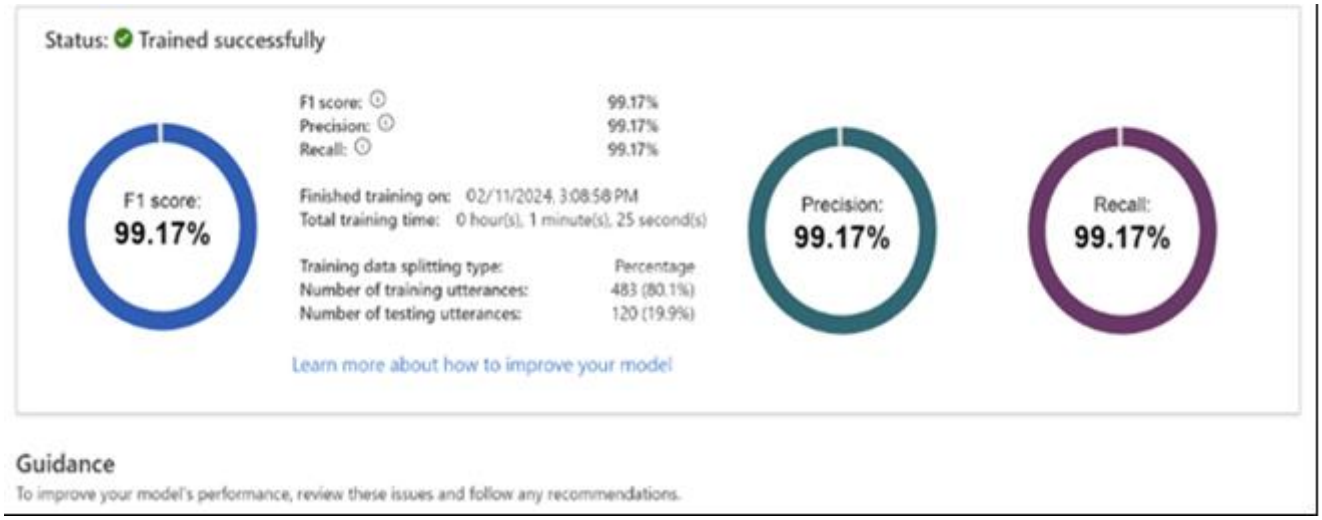

Рисунок 3.8 – Оцінка якості та точності моделі розпізнавання мови

Висновки до розділу 3

Попередня обробка даних та їх підготовка до навчання включала виявлення порожніх або невикористаних стовпців, які було вилучено, виявлення та вилучення дублікатів, перевірку некоректно введених даних, вилучення інформації про фільми, які оцінило менше, ніж 15 користувачів, та перетворення даних до числових шкал. Під час створення та навчання моделі машини факторизації FM, було враховано характеристики користувачів: їх стать та вік. Характеристику віку було дискретизовано з інтервалом у 5 років. Для фільмів було враховано як характеристики їх назви, рік створення та жанр. Для створення та навчання моделі матричної факторизації з урахуванням поля FFM характеристики користувачів

було розбито на поля: стать на поля – жіноча стать та чоловіча стать, вік – на вікові проміжки з періодом у 5 років. Для фільмів характеристику жанр було розбито на поля, що відповідають кожному жанру.

Якість матричних факторизаційних моделей та точності погнозу було оцінено з використанням показників MAE, MSE та RMSE. Оцінка якості факторизаційних моделей показала, що кращу точність прогнозу має модель машини факторизації з ураховунням поля FFM. Тому для створення рекомендаційної системи з надання рекомендацій по перегляду відеофільмів було обрано модель FFM.

Оцінка якості створеної на навченої моделі обробки природної мови із використанням сервісу Conversational language understanding показала високу точність розпізнавання мови та намірів користувача при спілкування з чат-ботом – 99,17%.

4 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ НАДАННЯ РЕКОМЕНДАЦІЙ З ПЕРЕГЛЯДУ ВІДЕОФІЛЬМІВ

4.1 Діаграма діяльності взаємодій користувача з системою рекомендацій

Основним напрямком використання діаграми діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно надати алгоритми їх виконання [36].

Для того щоб відобразити послідовність взаємодій рекомендаційної системи та користувача була створена діаграма діяльності, наведена на рисунку 4.1.

Під час запуску програми користувача просять ввести ім'я. Якщо такий користувач є в системі, він може ввести одну з чотирьох команд: отримати рекомендації, отримати рекомендації за жанром або роком випуску фільму, попросити допомогу або вийти з програми.

Якщо користувач обирає перший варіант, то першим кроком є завантаження даних із його профілю: за раніше виставленими рейтингами та обраними жанрами система визначає переваги даного користувача. Далі для нього шукається користувач із схожими уподобаннями з набору даних MovieLens, потім для знайденого користувача підбирається список із п’яти фільмів, які йому найбільше сподобаються. Ці фільми система просить оцінити: нові дані з виставленими рейтингами зберігаються в його профіль, і наступного разу допоможуть у складанні точніших рекомендацій.

У разі вибору другого варіанту, користувачу необхідно ввести бажаний для перегляду жанр чи рік фільму. Після цього серед усіх фільмів за обраним жанром або роком формується список із п’яти фільмів із найвищим рейтином. Враховуються також фільми, які користувач переглядав та оцінював у застосунку, такі фільми у списку не з’являться.

Якщо вибрано третій варіант з меню, користувачу виводиться приклад запиту для отримання рекомендацій.

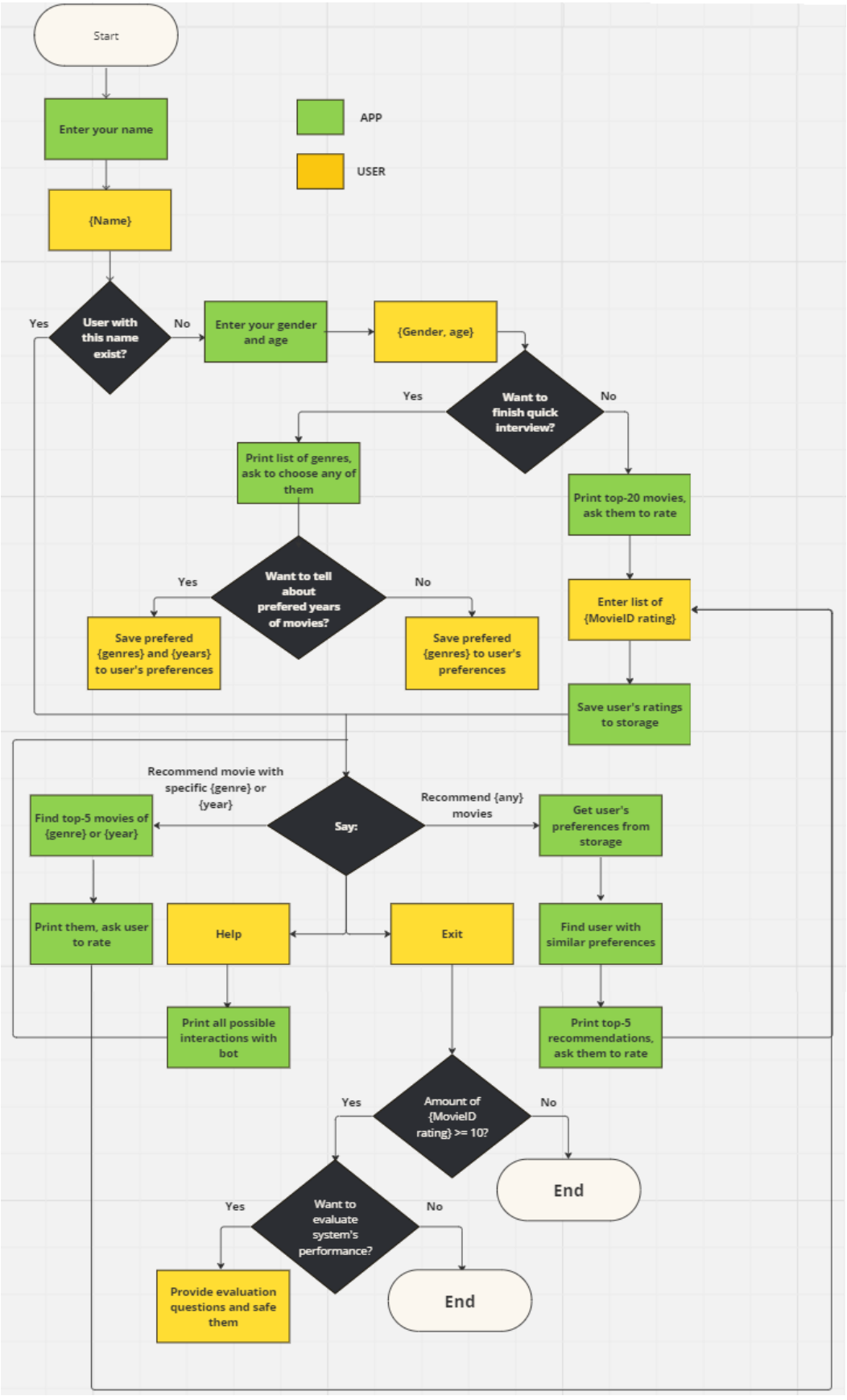

Рисунок 4.1 – Діаграма діяльності взаємодій користувача з системою рекомендацій

При виході з програми (четвертий варінт) застосунок перевіряє чи оцінив поточний користувач принаймні 10 фільмів. Якщо ні – вихід із програми. Якщо так – застосунок питає, чи хоче користувач надати оцінку роботі застосунку. У разі негативної відповіді – вихід із програми. У разі позитивної – проводиться коротке опитування користувача, яке описано у підрозділі 4.7.

У випадку, коли користувач вперше зайшов у систему, і про нього немає жодних даних, йому необхідно надати інформацію щодо його статі та віку, потім пропонується надати деякі свої вподобання про улюблені жанри або роки виходу фільмів. Якщо користувач відмовляється надавати цю інформацію – виводиться список із двадцяти найпопулярніших фільмів, які він може оцінити, тим самим склавши свій початковий профіль. Після такого анкетування користувач може вводити раніше згадані команди, і алгоритм дій повторюється.

4.2 Зберігання інформації про користувачів

Щоб програма могла складати ефективні рекомендації для користувачів, необхідно зберігати інформацію про нього та його попередні вподобання. Така інформація включає рейтинги, проставлені фільмам, які користувач просто дивився, та/або які йому сподобалися, його вік та стать.

Для збереження цієї інформації було обрано сервіс Azure Table storage. Сховище таблиць Azure відмінно підходить для цієї мети, тому що для даних не потрібні зовнішні ключі, складні з'єднання з іншими таблицями, і в цілому Azure Table storage зручний у використанні та надає швидкий доступ до даних.

Щоб отримати доступ до сервісу Azure Table storage, необхідна підписка Azure та ресурс Storage account, створений на порталі Azure. На рисунку 4.2 представлено створений на порталі Azure ресурс Storage account.

Для початку роботи з Azure Table storage у програмі необхідно встановити NuGet пакет Microsoft.Azure.Cosmos.Table. Далі на порталі Azure в розділі Access keys раніше створеного ресурсу береться рядок підключення до сховища, він

записується у файлі програми і потрібен для створення об'єкта класу CloudStorageAccount, який дозволяє створити об'єкт класу CloudTableClient, що відповідає за всі операції з таблицями. Він дозволяє отримувати таблиці (а також створювати, якщо таблиці не існує, за допомогою методу CreateIfNotExists) та сутності, що зберігаються у сховищі таблиць.

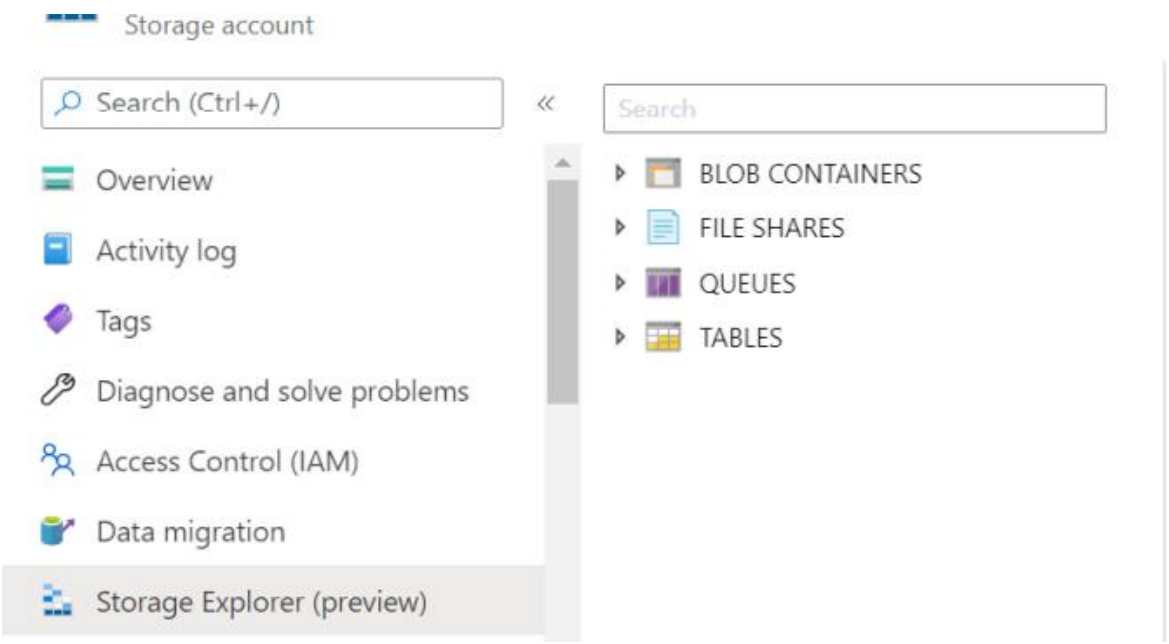

Рисунок 4.2 – Ресурс Storage account на порталі Azure

Реалізація підключення до сховища та операцій з таблицею та даними у ній представлена у класі UserStorage (додаток Д).

Щоб визначити сутність, яка представлятиме об’єкт, що зберігається в таблиці, необхідно створити клас, який наслідується від класу TableEntity, при цьому клас повинен мати конструктор без параметрів. Реалізація класу UserRatingEntity також представлена в додатку Д. Як ключ рядка використовується ім'я (логін) користувача, а ключ розділу для всіх користувачів - "Users". Ключ рядка та ключ розділу однозначно ідентифікують кожного користувача у таблиці. Користувацькі властивості сутностей, які зберігаються в таблиці, повинні бути загальнодоступними властивостями класу сутності та підтримувати як читання (метод get), так і запис (метод set) значень. У класі UserRatingEntity така властивість

одна - RatingsInternal. Ця властивість має тип string і отримує своє значення шляхом JSON серіалізації списку об'єктів класу Rating. Клас Rating представлений двома властивостями: MovieId – унікальний ідентифікатор фільму та RatingValue – значення рейтингу від одного до п'яти, виставленого даним користувачем відповідного фільму.

Всі операції з даними здійснюється за допомогою використання об'єкта класу CloudTableClient та виклику його методу Execute, якому як параметр передається об'єкт класу TableOperation - таблична операція. Для вставки в таблицю інформації про нового користувача використовується метод InsertOrReplace класу TableOperation, для оновлення інформації про рейтинги, виставлені вже існуючим користувачем Replace, для отримання запису по ключу Retrieve.

4.3 Вирішення проблеми холодного старту для користувача

Якщо користувач вперше зайшов у застосунок, то про нього ще немає жодних даних: які йому фільми подобаються, які навпаки, що він уже дивився, а що ні, і таке інше. І тут система рекомендацій неспроможна зробити прогноз, оскільки немає початкових даних, на яких має грунтуватися рекомендація [37]. Основна стратегія роботи з новим користувачем у такому разі – попросити надати деяку інформацію про його переваги у фільмах для створення початкового профілю користувача: система матиме деяку відправну точку для роботи з цим користувачем. Вся ця інформація збирається у процесі реєстрації. Діаграма діяльності цієї стратегії наведено на рисунку 4.3.

Рисунок 4.3 – Діаграма діяльності процесу роботи з новим користувачем

Таким чином, після входу нового користувача в додаток повинен бути запит на надання деяких початкових даних. Для цього користувачу необхідно відповісти боту про свій вік та стать у форматі «Чоловік 22» або «Жінка 35». Далі користувачу буде запропоновано пройти попереднє опитування щодо його вподобань, в яке включається вибір улюблених жанрів та бажані роки випуску фільмів (хтось любить класику, а хтось тільки нові стрічки).

Користувачу надається право пропустити обидва питання, або пропустит питання з приводу бажаних років. У випадку, коли користувач не матиме бажання відповідати на обидва запитання, для генерування перших особистих рекомендацій йому буде необхідно оцінити деякі фільми, які мають найвищий рейтинг. Список цих фільмів складається заздалегідь шляхом проходу по всіх рейтингах із файлу ratings.csv набору даних MovieLens, угруповання цього списку за фільмами, та сортування за найбільшим сумарним рейтингом для кожного фільму. Отриманий список записується у файл best_movies.csv для швидкого доступу до даних, таким чином, для кожного нового користувача не доведеться знову знаходити найпопулярніші фільми. Приклад представлених даних у файлі best_movies.csv відображено на рисунку 4.4.

318	Shawshank Redemption, The (1994)	Crime|Drama
356	Forrest Gump (1994)	Comedy|Drama|Romance|War
296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
593	Silence of the Lambs, The (1991)	Crime|Horror|Thriller
2571	Matrix, The (1999)	Action|Sci-Fi|Thriller
260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi
527	Schindler's List (1993)	Drama|War
480	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller
2959	Fight Club (1999)	Action|Crime|Drama|Thriller
110	Braveheart (1995)	Action|Drama|War

Рисунок 4.4 – Список фільмів з навищим рейтингом за увесь час

Після того, як користувачу виведено список із двадцяти найкращих фільмів у системі, йому пропонується оцінити деякі з них. Швидше за все користувач дивився деякі з цих фільмів, або чув щось про сюжет, і може скласти свою думку про кінокартину. Якщо ж ні, то можна зробити висновки, ґрунтуючись на жанрі фільму. Таким чином, користувач виставляє рейтинги за шкалою від одного до п’яти цим фільмам. При цьому абсолютно не обов’язково оцінювати всі двадцять фільмів, достатньо п’яти, щоб система рекомендацій могла зробити досить ефективний прогноз щодо переваг користувача.

4.4 Знаходження користувача зі схожими уподобаннями

Якщо користувач раніше використовував програму, то йому не потрібне анкетування, і немає необхідності виставляти рейтинги фільмам зі списку двадцяти найкращих у системі, оскільки він уже робив це раніше – при реєстрації в системі.

Якщо користувач виявляє бажання отримати рекомендації, то починається процес прогнозування. Насамперед завантажується інформація про виставлені рейтинги зі сховища Azure Table storage. Для цього використовується метод Retrieve класу TableOperation, за допомогою якого за ключом (ім’я або логін користувача) з таблиці вибирається список рейтингів фільмів, раніше виставлених користувачем. Для всіх операцій з профілем користувача був створений клас UserProfile, який у своїх методах працює з об'єктом класу UserStorage (додаток Д), що містить дані про ім’я, вік та стать користувача. Об’єкт цього класу безпосередньо працює з Azure Table storage. Отриманий список рейтингів передається в метод PredictTop5 класу Predictor (Додаток Е), створеного для реалізації функціоналу, пов'язаного безпосередньо з прогнозуванням.

У методі PredictTop5 спочатку викликається метод FindSimilarUserId, в якому для користувача знаходиться ідентифікатор користувача, найбільш схожого на нього за уподобаннями у фільмах.

Для цього рейтинги користувача перетворюються на вектор переваг, так як раніше це робилося для кожного користувача з набору даних MovieLens при складанні файлу user_ratings.csv: якщо фільм був оцінений користувачем, то у вектор записується значення рейтингу, якщо ні – нуль. Потім вектор нормалізується. Далі для отриманого вектора та кожного вектора з файлу user_ratings.csv знаходиться косинусна подібність.

Для обчислення косинусної подібності використовувалася платформа Accord.NET, зокрема бібліотека Accord.Math. Accord.NET надає методи для статистичного аналізу, машинного навчання, обробки зображень та комп'ютерного зору для .NET додатків. Платформа розділена на бібліотеки, доступні для встановлення як пакети NuGet. Бібліотека Accord.Math надає чисельні функції та інструменти для роботи з матрицями. Для знаходження косинусної подібності двох векторів використовувалася структура Cosine та її метод Similarity. Серед усіх отриманих значень знаходиться максимальне – користувач, якому належить відповідний вектор переваг, і є схожим на вихідного. Після цього для знайденого користувача обчислюється список рекомендацій з п’яти фільмів, які найімовірніше йому сподобаються, і які ще не оцінив вихідний користувач. Для цього використовується раніше навчена за допомогою технологій ML.NET модель.

4.5 Використання навченої моделі

Модель, що зберігається локально (проте можливе зберігання віддалено), може бути використана в інших застосунках. Насамперед її треба завантажити: для цього достатньо використання функції Load, куди як параметр передається шлях до файлу формату zip з навченою моделлю. Потім для більш детальної інформації про фільми завантажуються дані з файлу movies.csv. Нарешті, робиться прогноз.

Для цього використовується об'єкт класу PredictionEngine, куди як два універсальні параметри передається тип тестових даних, для яких буде

здійснюватися прогноз (MovieRating), і тип прогнозу (MovieRatingPrediction). Далі викликається метод Predict, куди як аргумент передаються вхідні дані, для яких робитиметься прогнозування.

Так як у застосунку необхідно вивести найкращі п'ять фільмів для користувача, то метод Predict викликається в циклі: здійснюється перебір всіх фільмів із файлу movies.csv, для кожного з них створюється об'єкт класу MovieRating, ініціалізований ідентифікаторами користувача та поточного фільму, і для кожного такого об'єкта робиться прогноз. Далі всі об'єкти сортуються за зменшенням якості Score об'єкта Prediction отриманого прогнозу, і вибираються п'ять перших елементів з тих фільмів, що користувач ще не дивився (інформації про них немає в його профілі).

Далі користувачеві пропонується оцінити рекомендовані фільми за шкалою від одного до п’яти для того, щоб дані про його переваги розширювалися: чим більше даних про користувача буде мати система рекомендацій, тим точніше будуть подальші прогнози. Це і є головна перевага побудованої системи рекомендацій: чим частіше користувач заходитиме у застосунок і проситиме нових рекомендацій, а потім даватиме зворотній відгук системі про те, чи сподобалися йому запропоновані фільми, тим точніше будуть рекомендації наступного разу.

4.6 Розробка застосунку чат-бота

Бот — це програма, з якою користувачі взаємодіють у розмовному режимі, використовуючи текст, графіку (наприклад, інтерактивні картки або зображення) або мовлення. На рисунку 4.5 представлено архітектуру бот-застосунку. На схемі відображено сам бот, а також його залежності та зв’язки із зовнішніми сервісами та компонентами.

Бот-застосунок розгортається за допомогою Azure App Service та виконується в Azure. У процесі роботи надсилаються дані телеметрії до App Insights, які надалі використовуються для аналізу роботи програми.

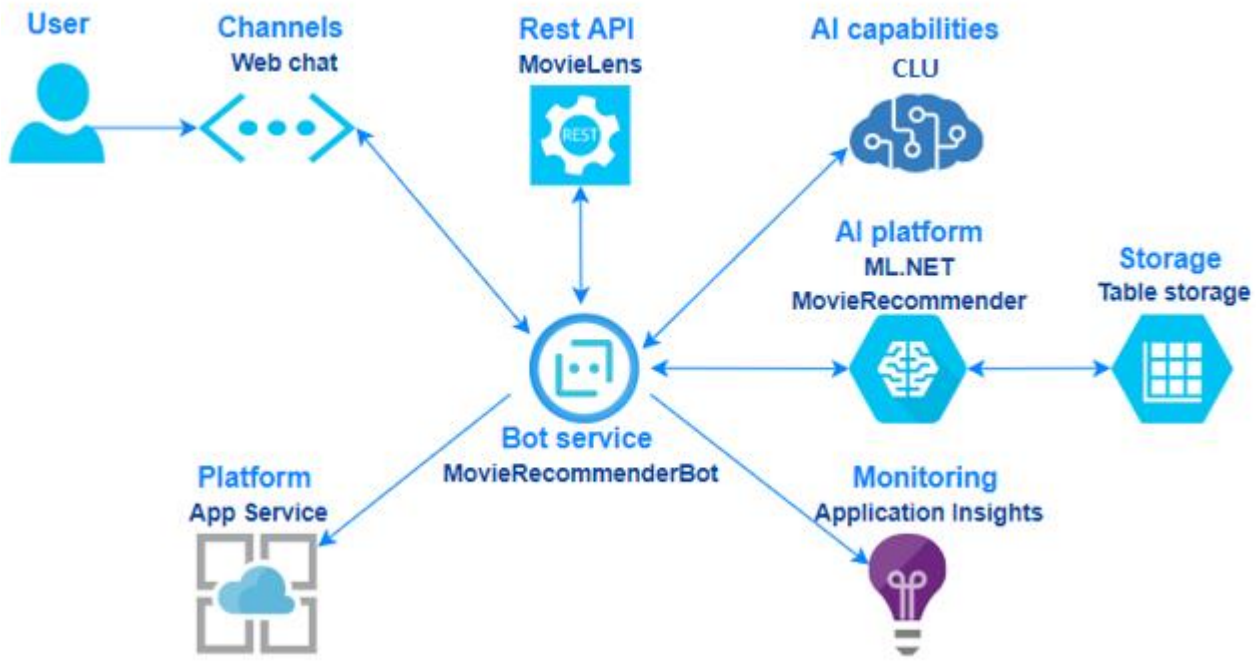

Рисунок 4.5 – Архітектура бота-застосунку

Розглянемо послідовність взаємодії компонентів бота:

– користувач не підключається безпосередньо до бота, а взаємодіє з ним через канал, який використовується для підключення бота до різних додатків, таких як Web Chat, Email, Skype і так далі;

– бот – це вебсервіс, який надає API /api/messages шляху, цей API використовується для отримання вхідних повідомлень від користувача та відповіді на них;

– за допомогою каналу і потім бот-програми користувач робить запит на отримання рекомендацій;

– бот використовує сервіс розпізнавання природної мови Conversational Language Understanding для обробки текстового запиту;

– бот інтегрується з раніше навченою за допомогою платформи ML.NET рекомендаційною системою;

– для отримання та збереження інформації про переваги (виставлені рейтинги) користувача використовується Azure Table storage;

– бот завантажує постери фільмів для інтерактивного відображення рекомендацій за допомогою MovieLens API.

Розглянемо детальніше принципи роботи бота. Сервіс Bot Framework, компонент Azure Bot Service, відправляє інформацію між застосунками користувача, які також називають каналами, і ботом.

У розмові люди зазвичай говорять один за одним, по черзі здійснюючи свій «хід» протягом розмови. У сервісі Bot Framework «хід» являє собою вхідну дію користувача, що ініціюється при зверненні до бота, і дію, що надсилається ботом у відповідь. Таким чином, «хід» - це свого роду обробка, пов’язана з надходженням будь-якої дії в діалозі. Дія надходить з HTTP запитом і подається у вигляді JSON об’єкта, який потім десеріалізується та відправляється для обробки до класу адаптера бота – наслідника стандартного класу BotFrameworkHttpAdapter. Код класу адаптера AdapterWithErrorHandler розробленого застосунку представлений у додатку В. Адаптер ініціалізує контекст «ходу», додаючи в нього відомості про саму дію, включаючи відправника та одержувача, канал та інші дані, необхідні для його коректної обробки, та передає його класу бота.

У класі бота, що є наслідником стандартного класу ActivityHandler, який у свою чергу реалізує інтерфейс IBot, визначені обробники дій (реалізація класу Bot відображена в додатку Г).

Базовим обробником є метод OnTurnAsync – так званий обробник "ходу". Вся обробка дій у діалозі починається саме там, а потім для кожної отриманої дії викликається метод індивідуальної обробки, залежно від його типу. Наприклад, якщо користувач ініціює дію, яка є відправкою повідомлення, базовий обробник перенаправить його обробнику OnMessageActivityAsync. Дія, що генерується при додаванні нових користувачів до діалогу, передається в метод

OnMembersAddedAsync. Так як надалі бот підключається до каналу Web Chat, то зручно також перевизначити метод OnConversationUpdateActivityAsync, який викликається при додаванні або видаленні користувача з діалогу, і в першому випадку викликає OnMembersAddedAsync.

У розроблюваному застосунку чат-бота при додаванні нового користувача метод SendWelcomeMessageAsync класу Bot, викликаний згаданими вище обробниками, виводить вітальне повідомлення для користувача, а також запит на введення імені. Реалізовані у чат-боті каскадні діалоги складаються з наступних кроків.

На початку бот звертається до користувача з проханням отримання його імені, виводить вітання та перевіряє, чи є користувач у системі: якщо про користувача вже є дані – перехід до наступного кроку, якщо ні – проводиться попереднє опитування щодо статі та віку (рис. 4.6).

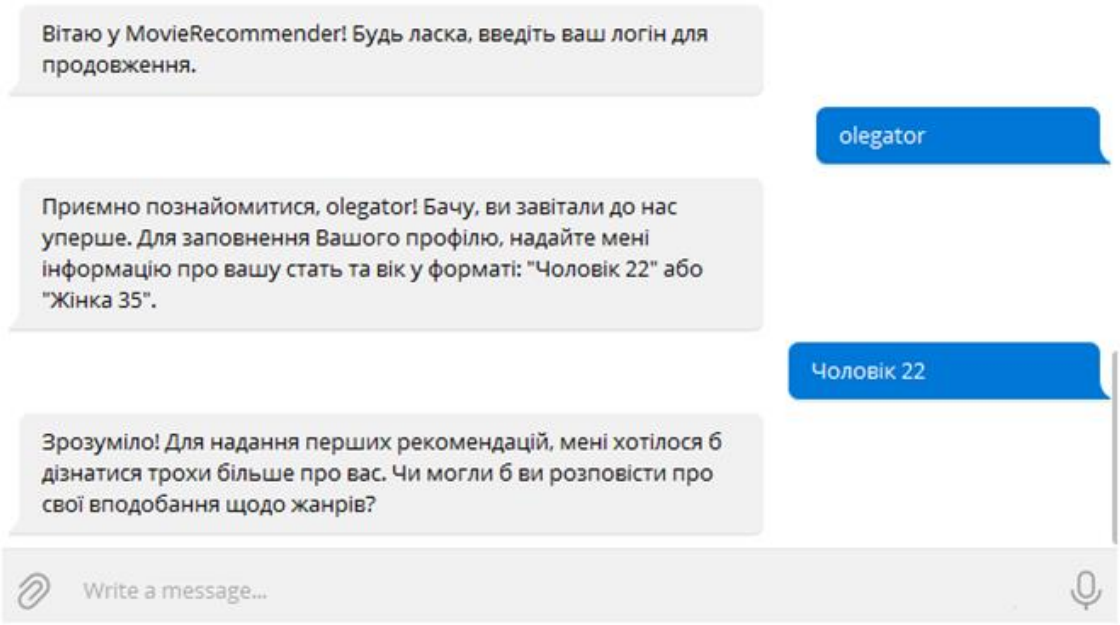

Рисунок 4.6 – Реєстрація нового користувача

Далі пропонується навести деякі відомості про вбодобання, користувач має можливість надати інформацію щодо вподобань (рисунок 4.7) або відмовитися

надати інформацію щодо вподобань та оцінити деякі з фільмів (рисунок 4.8), оцінені фільми зберігаються у профілі користувача.

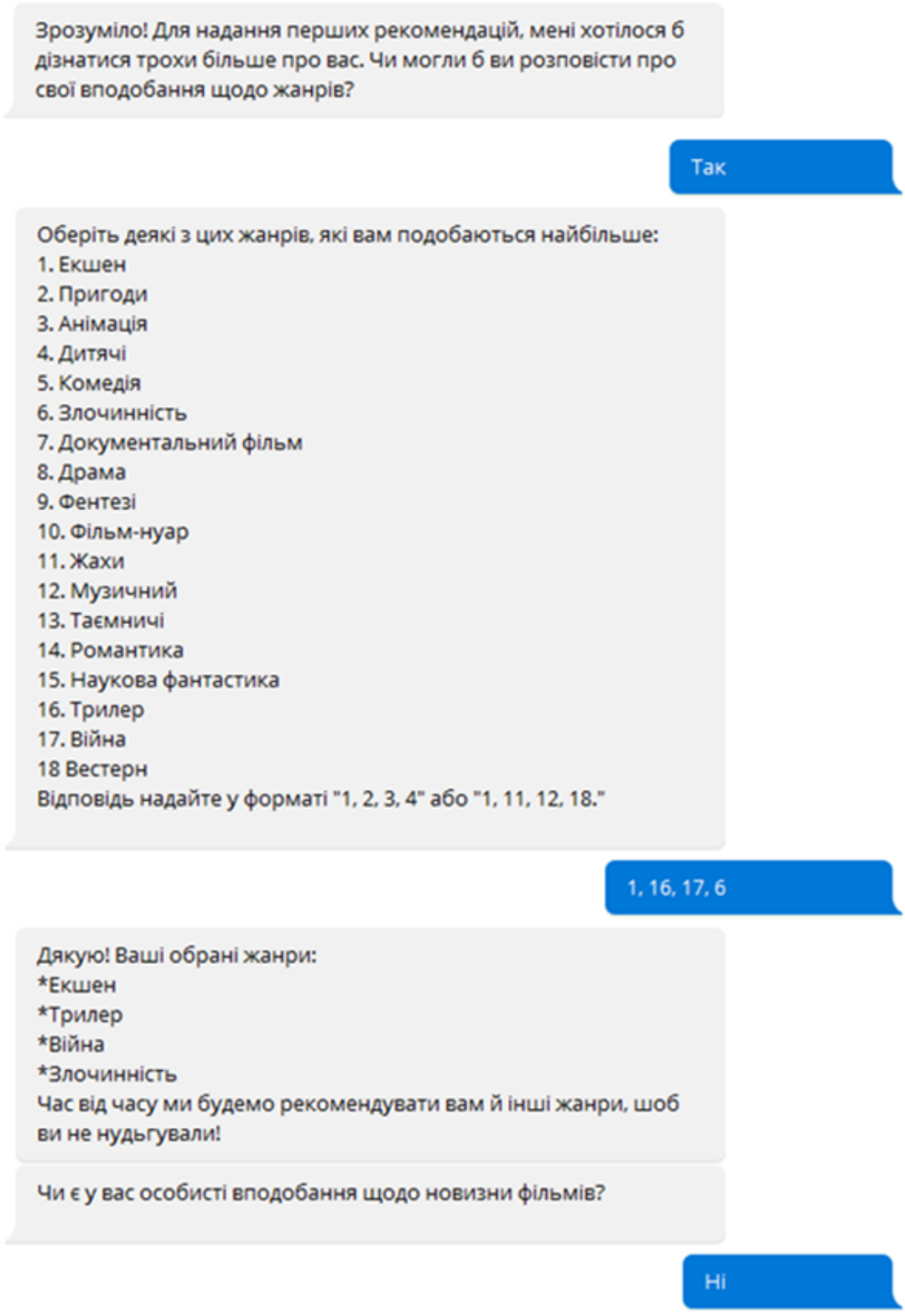

Рисунок 4.7 – Надання інформації з боку користувача

Запит користувача на отримання рекомендацій та відповідь застосунку наведено на рисунку 4.9.

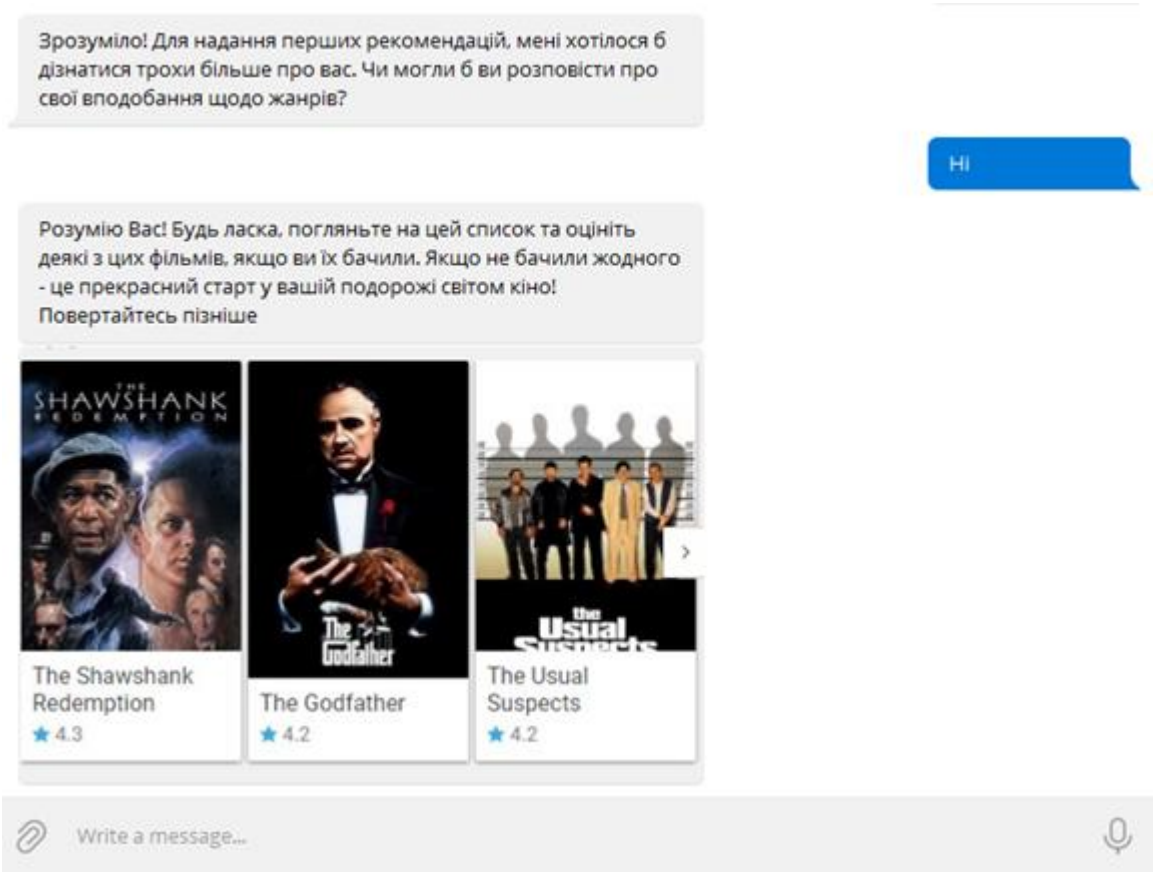

Рисунок 4.8 – Користувач відмовляється надавати дані про свої вподобання

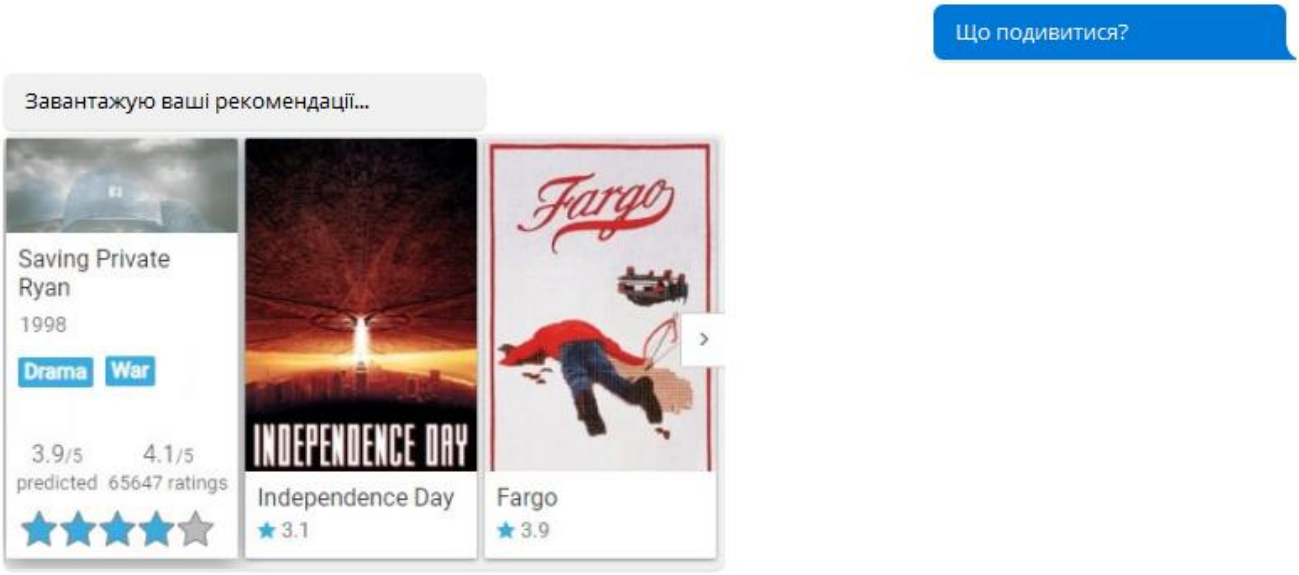

Рисунок 4.9 – Отримання рекомендацій від застосунку

Можемо бачити, що відповідь застосунку включає можливість оцінки рекомендованих фільмів безпосередньо у чаті а також виводить деякі дані щодо фільмів: рік випуску фільму, жанри до яких він відноситься, очікувану оцінку від даного користувача, загальний рейтинг фільму та кількість оцінок від інших користувачів.

Запит користувача на отримання рекомендацій конкретного жанру (фільми, які відмічені декількома жанрами також можуть входити у список) та відповідь застосунку наведено на рисунку 4.10.

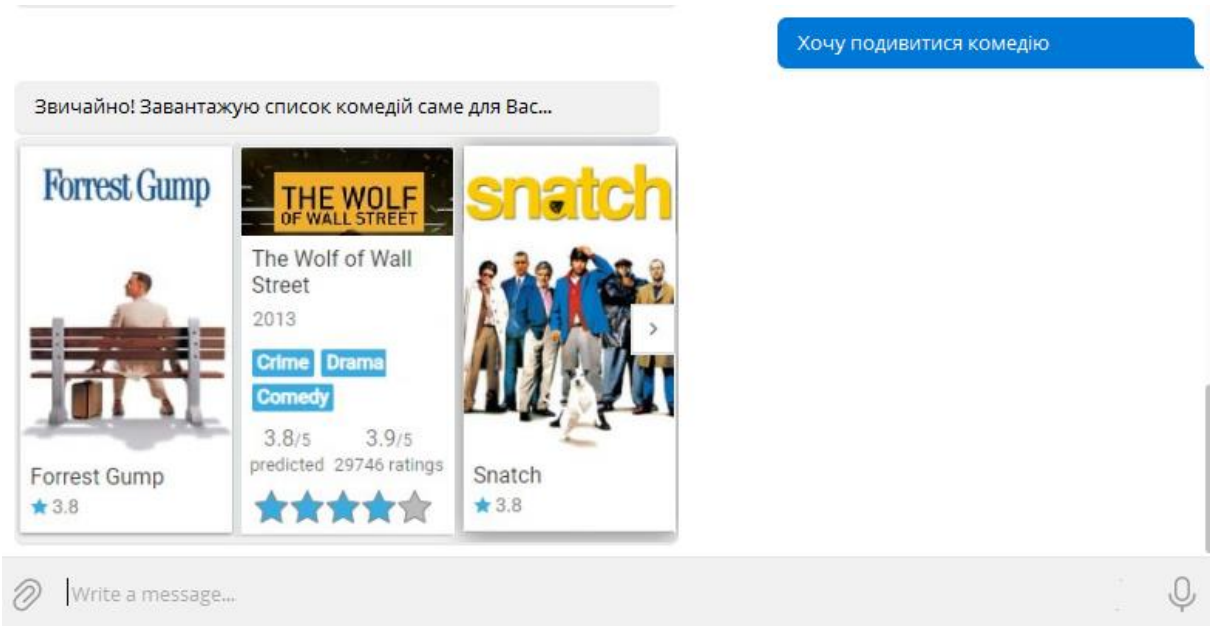

Рисунок 4.10 – Отримання рекомендацій від застосунку за жанром

Запит на отримання допомоги та вихід з облікового запису користувача наведено на рисунку 4.11.

Чат-бот вбудовано у вебзастосунок, початкова сторінка якого містить кнопку для виклику бота та спілкування з ним з метою отримання рекомендацій з перегляду відеофільмів (рисунок 4.12).

Рекомендовані фільми можна переглянути у будь-якому сервісі, який надає доступ до відеоконтенту.

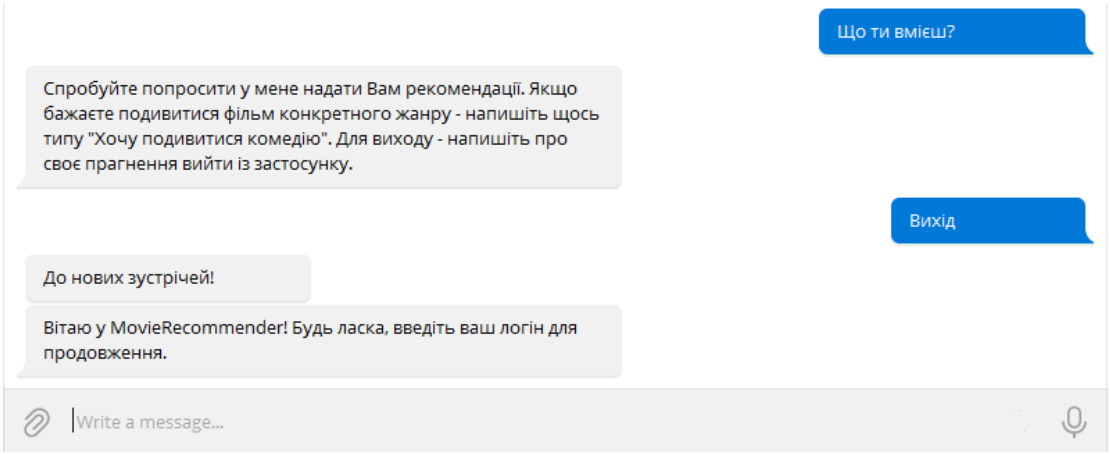

Рисунок 4.11 – Надання рекомендацій з отримання допомоги та виходу з облікового запису

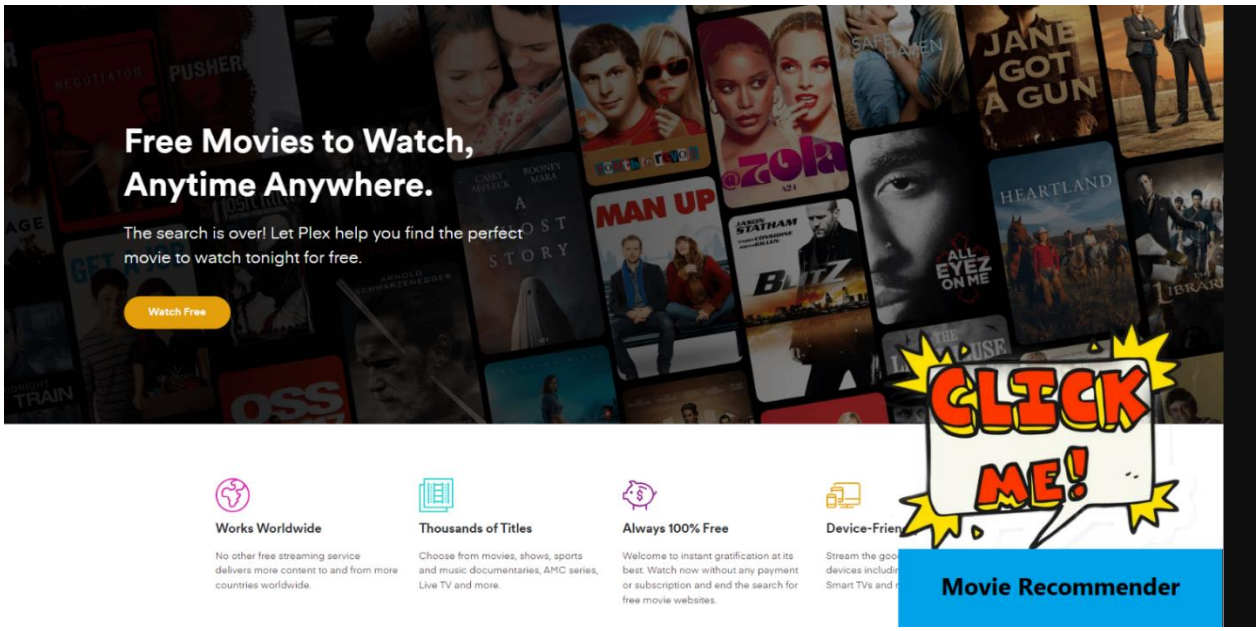

Рисунок 4.12 – Початкова сторінка вебзастосунку для отримання рекомендацій з перегляду відеофільмів

4.7 Оцінка ефективності чат-бота

Для дослідження ефективності взаємодії користувачів із чат-ботом було проведено анкетування та оброблено його результати. Анкетування проводилось онлайн серед 30 респондентів віком від 18 до 26 років, більшість з яких були

студентами. Користувачі саме цього вікового діапазону добре знайомі з цифровими сервісами та часто переглядають фільми.

Розроблена анкета містить 7 питань, згрупованих за 6 факторами, а саме інформативність (INF), легкість у використанні (ETU), сприймана якість рекомендацій (PRQ), легкість розуміння (EOU), довіра (TR) і сприймана ефективність (PE) (див. табл. 4.1). У таблиці 4.2 наведено можливі варіанти відповідей на питання анкети та їх ваги.

Таблиця 4.1 – Питання анкети

ID	Factor	Питання
P1	ETU (Easy To Use)	Ви можете легко орієнтуватися та взаємодіяти з системою чат-бота
P2	ETU (Easy To Use)	Інструкції та вказівки, які надає система, чіткі та інтуїтивно зрозумілі
P3	EOU (Ease Of Understanding)	Вам легко зрозуміти рекомендації, які надає система
P4	PE (Perceived Efficiency)	Ви задоволені швидкістю та чутливістю системи
P5	PRQ (Perceived Recommendation Quality)	Рекомендовані відео відповідають вашим вподобанням та інтересам
P6	INF (Informative)	Система надає достатньо інформації про кожен рекомендований відео контент
P7	TR (Trust)	Ви скористаєтеся рекомендаціям системи для майбутнього вибору відео

Остаточна кількість балів з кожного питання анкети розраховується за формулою:

$$Final\ Score = \frac{Total\ score}{Maximum\ score} \times 100, \quad (4.1)$$

де *Total score* – отримана сумарна кількість балів,

Maximum score – максимально можлива кількість балів, рівна $5 \times 30 = 150$.

Таблиця 4.2 – Вагові коефіцієнти оцінки відповідей на питання анкети

Опис	Ваги	Оцінка
DS	Не згоден (*Disagree*)	1
SD	Частково не згоден (*Somewhat Disagree*)	2
NT	Нейтрально (*Neutral*)	3
SA	Частково згоден (*Somewhat Agree*)	4
ST	Абсолютно згоден (*Strongly Agree*)	5

Результати опитування зберігаються у окрему таблицю user_evaluate.csv, яка має у своїй структурі ID користувача та надані цим користувачем оцінки параматрів системи. Коли користувач прагне вийти із застосунку, бот пропонує пройти невеличке опитування та надає можливість виставити оцінки за відомою користувачеві системою, а саме у вигляді інтерактивних карток (рисунок 4.13 та 4.14).

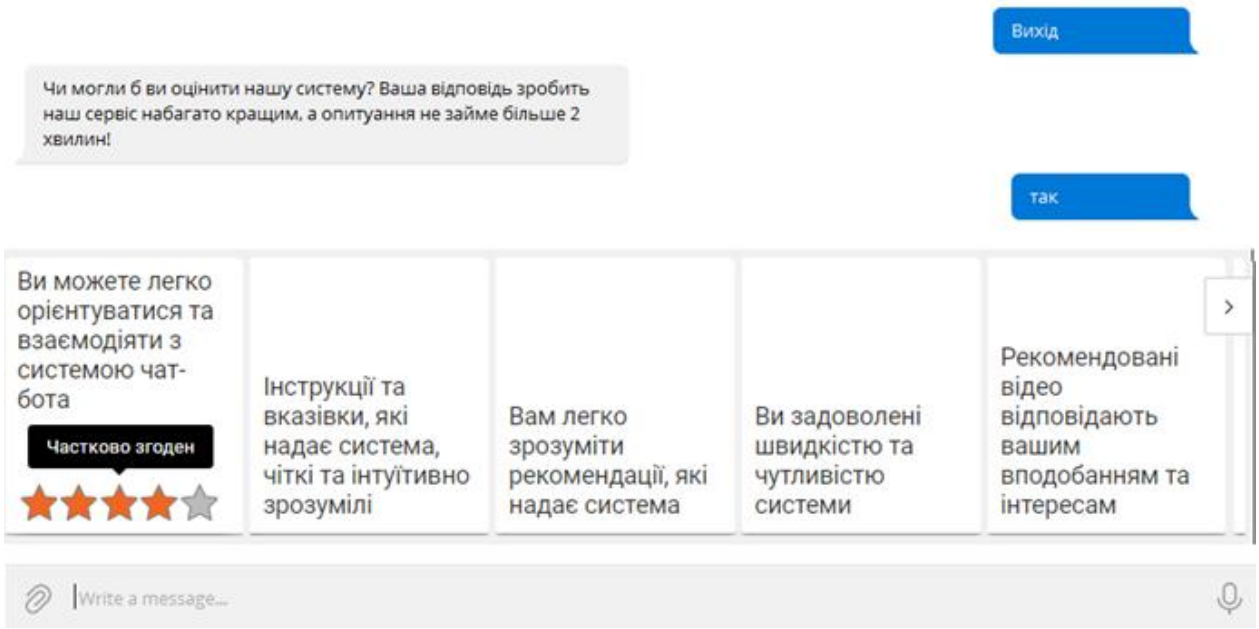

Рисунок 4.13 – Опитування користувача щодо ефективності та зручності застосунка

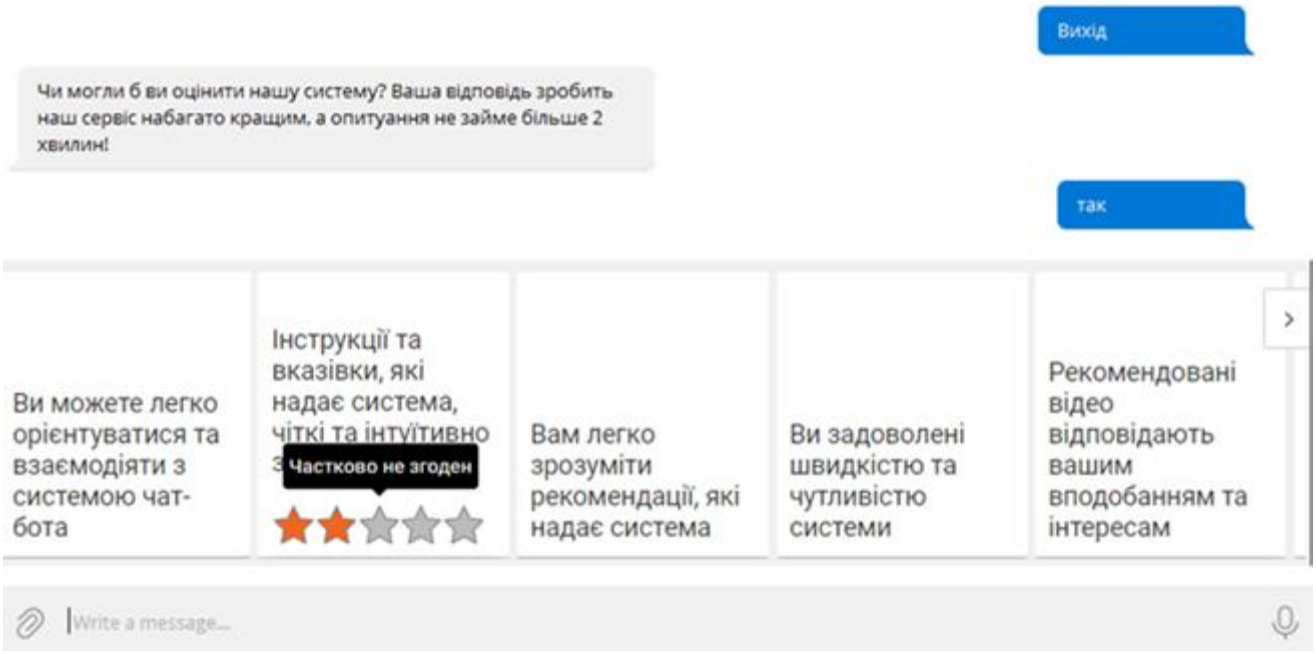

Рисунок 4.14 – Демонстрація інтерактивності під час опитування

У таблиці 4.3 показано результати проведеного тестування за допомогою анкети, які показують, що запропонована система рекомендацій на основі чат-бота може забезпечити задовільні результати для користувачів із кінцевим рівнем їх задоволеності 86,6%.

Таблиця 4.3 – Результати проведеного анкетування

Питання анкети	Кількість відповідей на питання анкети					Final Score
	DS	SD	NT	SA	ST	
P1			7	10	13	$(126/150) \times 100 = 84{,}0\%$
P2			5	9	16	$(131/150) \times 100 = 87{,}3\%$
P3			2	11	17	$(135/150) \times 100 = 90{,}0\%$
P4		1	2	14	13	$(129/150) \times 100 = 86{,}0\%$
P5			4	13	13	$(129/150) \times 100 = 86{,}0\%$
P6		1	2	15	12	$(128/150) \times 100 = 85{,}3\%$
P7			2	15	13	$(131/150) \times 100 = 87{,}6\%$
Остаточний результат						86,6%

Таким чином, дослідження ефективності взаємодії користувачів із чат-ботом показало високий рівень задоволеності користувачів результатами спілкування та отриманими рекомендаціями.

Висновки до розділу 4

Здійснено розробку та програмну реалізацію інтелектуальної системи для надання рекомендацій з перегляду відеофільмів. Рекомендаційна система розроблена на основі моделі матричної факторизації з урахуванням полів FFM та інтегрована з чат-ботом, впровадженим у веб-застосунок. Розпізнавання намірів користувача здійснюється у процесі його спілкування з чат-ботом шляхом реалізації каскадних діалогів, які передають інформацію, отриману на кожному етапі розмови, на наступний крок. Таким чином реалізовано гнучку взаємодію рекомендаційної системи з користувачем, який може отримати рекомендації з урахування накопиченої інформації про його інтереси та уподобання та з урахуванням вказаного жанру та дати виходу фільму у прокат у момент спілкування з чат-ботом. Користувач також може попросити допомогу у наданні рекомендацій. Рекомендовані для перегляду фільми система просить оцінити, оновлюючи інформацію про їх рейтинги у профілі користувача. Вирішення проблеми холодного старту у системі реалізовано шляхом виявлення у процесі спілкування з новим користувачем його інтересів та уподобань. Рекомендовані фільми можна переглянути у будь-якому сервісі, який надає доступ до відеоконтенту.

Дослідження ефективності взаємодії користувачів із чат-ботом показало високий рівень задоволеності користувачів результатами спілкування та отриманими рекомендаціями – 86,6%.

ВИСНОВКИ

Досліджено основні підходи до надання рекомендацій у рекомендаційних системах. Установлено, що застосування методів матричної факторизації полегшує складність масштабування, а використання у рекомендаційній системі чат-бота дозволяє вирішити проблему холодного старту та забезпечує гнучність у взаємодії користувача з системою. Здійснений аналіз мережевих сервісів доступу до відеоконтенту дозволив установити, що при наданні рекомендацій вони націлені на вирішення комерційних задач і не використовують чат-боти для спілкування з користувачами. Обгрунтовано доцільність розробки інтелектуальної системи на основі методів матричної факторизації, інтегрованої з чат-ботом для надання персоналізованих рекомендацій з перегляду відеоконтенту.

Виявлено, що модель матричної факторизації MF спрощує розрахункову складність системи за рахунок декомпозиції вихідної матриці рейтингів на добуток двох матриць меншого рангу. Існує багато гібридних моделей MF, до яких відносять машини факторизації FM та машини факторизації з урахуванням специфіки поля FFM. Машини факторизації притримуються базового математичного підходу, комбінуючи факторизацію матриць із регресією. Машини факторизації з урахуванням специфіки поля є узагальненням та покращенням машин факторизації за рахунок розширення кількості властивостей, на яких базуються рекомендації. Виявлення багатьох модифікацій методів матричної факторизації потребує їх подальшого дослідження з метою установлення оптимальних умов їх застосування при розробці рекомендаційних систем для покращення персоналізованих рекомендацій.

Для створення моделей машинного навчання MF та FM було обрано платформу ML.NET, яка надає середовище для навчання, тестування й оцінки моделі та дозволяє використовувати її у застосунках не тільки платформи .NET, але й іншими мовами, а також у хмарних сервісах. Створення та навчання моделі FM реалізовано з використанням бібіотеки Pytorch-Acceleratd.

Створення та управління інтелектуальним чат-ботом для надання рекомендацій реалізовано з використанням Microsoft Bot Framework і Azure Bot Service. Обробка природної мови та розпізнавання намірів користувача здійснювалося із використанням хмарних API Azure Cognitive Services та Conversational language understanding. Сервіс Azure Monitor було застосовано з метою моніторингу й аналізу даних у процесі роботи чат-бота. Для зберігання даних використано NoSQL-сховище Azure Table storage, завдяки чому було реалізовано масштабування у відповідності з потребами. У даному дослідженні чат-бот було інтегровано у середовище вебзастосунку, для розробки якого використано мову розмітки HTML та CSS як засіб стилізації.

Попередня обробка даних та їх підготовка до навчання включала виявлення порожніх або невикористаних стовпців, які було вилучено, виявлення та вилучення дублікатів, перевірку некоректно введених даних, вилучення інформації про фільми, які оцінило менше, ніж 15 користувачів, та перетворення даних до числових шкал. Під час створення та навчання моделі машини факторизації FM, було враховано характеристики користувачів: їх стать та вік. Характеристику віку було дискретизовано з інтервалом у 5 років. Для фільмів було враховано як характеристики їх назви, рік створення та жанр. Для створення та навчання моделі матричної факторизації з урахуванням поля FFM характеристики користувачів було розбито на поля: стать на поля – жіноча стать та чоловіча стать, вік – на вікові проміжки з періодом у 5 років. Для фільмів характеристику жанр було розбито на поля, що відповідають кожному жанру.

Оцінка якості факторизаційних моделей показала, що кращу точність прогнозу має модель машини факторизації з ураховунням поля FFM. Тому для створення рекомендаційної системи з надання рекомендацій по перегляду відеофільмів було обрано модель FFM. Оцінка якості створеної на навченої моделі обробки природної мови із використанням сервісу Conversational language understanding показала високу точність розпізнавання мови та намірів користувача при спілкування з чат-ботом – 99,17%.

Здійснено розробку та програмну реалізацію інтелектуальної системи для надання рекомендацій з перегляду відеофільмів. Рекомендаційна система розроблена на основі моделі матричної факторизації з урахуванням полів FFM та інтегрована з чат-ботом, впровадженим у веб-застосунок. Розпізнавання намірів користувача здійснюється у процесі його спілкування з чат-ботом шляхом реалізації каскадних діалогів, які передають інформацію, отриману на кожному етапі розмови, на наступний крок. Таким чином реалізовано гнучку взаємодію рекомендаційної системи з користувачем, який може отримати рекомендації з урахування накопиченої інформації про його інтереси та уподобання та з урахуванням вказаного жанру та дати виходу фільму у прокат у момент спілкування з чат-ботом. Користувач також може попросити допомогу у наданні рекомендацій. Вирішення проблеми холодного старту реалізовано шляхом виявлення у процесі спілкування з новим користувачем його інтересів та уподобань. Рекомендовані фільми можна переглянути у будь-якому сервісі, який надає доступ до відеоконтенту. У процесі користування рекомендаційною системою накопичується інформація стосовно інтересів та потреб користувача і точність рекомендацій стає вищою.

Дослідження ефективності взаємодії користувачів із чат-ботом показало високий рівень задоволеності користувачів результатами спілкування та отриманими рекомендаціями – 86,6%.

Поставлені завдання виконано повністю, однак додавання у систему інформації стосовно нових фільмів у режимі реального часу не є вирішеним, що обумовлює необхідність вдосконалення функціоналу застосунку у цьому напрямку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fayyaz Z., Ebrahimian M., Nawara D., Ibrahim A., Kashef R. Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities. *Applied Sciences*. Vol. 10(21). 2020. URL: https://doi.org/10.3390/app10217748 (дата звернення 25.01.2024).

2. Mehdi Elahi, Matthias Braunhofer, Francesco Ricci, Marko Tkalcic. Personality-Based Active Learning for Collaborative Filtering Recommender Systems. URL: http://www.cp.jku.at/research/papers/elahi2013aixia.pdf. (дата звернення 25.01.2024).

3. Prem Melville, Raymond J. Mooney, Ramadass Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations . URL: https://www.aaai.org/Papers/AAAI/2002/AAAI02-029.pdf. (дата звернення: 12.12.2023).

4. D. Billsus, M. J. Pazzani. Learning collaborative information filters. in Icml, vol. 98, 1998. 46–54 с.

5. Мелешко Є. В., Хох В. Д., Босько В.В. Дослідження матричних факторизаційних моделей рекомендаційних систем. *Системи управління, навігації та зв'язку*. 2019. Вип. 6. 58-62 с. URL: http://nbuv.gov.ua/UJRN/suntz_2019_6_13. (дата звернення: 10.02.2024)

6. Meleshko E. Дослідження методів побудови рекомендаційних систем в мережі Інтернет / E. Meleshko, S. Semenov, V. Khokh // Системи управління, навігації та зв’язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 1 (47). 131-136 с.

7. Tim Roughgarden , Gregory Valiant. The Modern Algorithmic Toolbox Lecture #9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations. URL: https://web.stanford.edu/class/cs168/l/l9.pdf. (дата звернення 22.12.2023).

8. An Intuitive Explanation of Field Aware Factorization Machines. URL: https://towardsdatascience.com/anintuitive-explanation-of-field-aware-factorization-machines-a8fee92ce29f. (дата звернення: 13.12.2023).

9. Peng Yan, Xiaocong Zhou, Yitao Duan. E-Commerce Item Recommendation

Based on Field-aware Factorization Machine. URL:https://www.researchgate.net/publication/282846395_E-Commerce_Item_Recommendation_Based_on_Field-aware_Factorization_Machine. (дата звернення 12.01.2024).

10. Zhang Z., Lui Y., Zhang Zh. Field-Aware Matrix Factorization for Recommender Systems. *IEEE Access*. Vol. 6. 2018. P. 45690-45698. URL: https://doi.org/10.1109/ACCESS.2017.2787741. (дата звернення 25.01.2024).

11. The ultimate guide to binary classification metrics. URL: https://towardsdatascience.com/the-ultimate-guide-to-binaryclassification-metrics-c25c3627dd0a (дата звернення 12.01.2024).

12. Introduction to recommender systems. URL: https://towardsdatascience.com/introduction-to-recommender-systems6c66cf15ada (дата звернення 13.12.2023).

13. Jayalakshmi S., Ganesh N., C¨ep R., Senthil Murugan J. (2022). Movie recommender systems: concepts, methods, challenges, and future directions. *Sensors*. Vol. 22(13). URL: https://doi.org/10.3390/s22134904. (дата звернення 25.01.2024).

14. Gomez-Uribe C.A., Hunt N. The netflix recommender system: algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*. Vol. 6, No 4. 2016. P. 1–19. URL: https://dl.acm.org/doi/10.1145/2843948. (дата звернення 25.01.2024).

15. Abbas M., Riaz M.U., Rauf A., Khan M.T., Khalid S. Context-aware Youtube recommender system. In *International Conference on Information and Communication Technologies* (*ICICT*). Karachi: IEEE, 2017. P. 161–164. URL: https://doi.org/10.1109/ICICT.2017.8320183 (дата звернення 25.01.2024).

16. MovieLens Latest Full. URL: https://www.kaggle.com/grouplens/movielens-latest-full. (дата звернення 07.01.2024).

17. 27 Incredible Chatbot Statistics. URL: https://www.netomi.com/chatbot-statistics. (дата звернення 05.01.2024).

18. Bowlin G. Building Progressive Web Apps, Createspace Independent Pub. 2017.

178 с. (дата звернення: 10.02.2024)

19. Nugraha M., Baizal Z.K.A., Richasdy D. Chatbot-Based Movie Recommender System Using POS Tagging. *Building of Informatics, Technology and Science* (*BITS*). 2022. Vol. 4, No. 2. P. 624-630. URL: https://doi.org/10.47065/bits.v4i2.1908. (дата звернення 25.01.2024).

20. Hindawi Publishing Corporation, Advances in Artificial Intelligence archive, USA : журнал. — 2009. 1—19 с.

21. Machine Learning. URL: https://reposhub.com/dotnet/machine-learning-and-data-science/dotnetmachinelearning.html. (дата звернення: 12.12.2023).

22. What is ML.NET and how does it work? URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work. (дата звернення 05.01.2024).

23. Announcing ML.NET. URL: https://devblogs.microsoft.com/dotnet/announcing-ml-net-0-3/#onnx-section. (дата звернення 05.01.2024).

24. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. URL: http://download.tensorflow.org/paper/whitepaper2015.pdf. (дата звернення 05.01.2024).

25. ONNX Runtime for inferencing machine learning models. URL: https://azure.microsoft.com/en-us/blog/onnx-runtime-for-inferencing-machine-learning-models-now-in-preview/. (дата звернення 05.01.2024).

26. Welcome to pytorch-accelerated’s documentation! URL: https://pytorch-accelerated.readthedocs.io/en/latest/. (дата звернення 05.01.2024).

27. Get started guide for developers on Azure. URL: https://docs.microsoft.com/en-us/azure/guides/developer/azure-developerguide#what-is-azure. (дата звернення 06.01.2024).

28. About Azure Bot Service. URL: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overviewintroduction?view=azure-bot-service-4.0. (дата звернення 06.01.2024).

29. Azure Monitor overview. URL: https://docs.microsoft.com/en-us/azure/azure-monitor/overview. (дата звернення 06.01.2024).

30. What are Azure Cognitive Services? URL: https://docs.microsoft.com/en-us/azure/cognitive-services/welcome. (дата звернення 07.01.2024).

31. What is conversational language understanding? URL: https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/overview#example-usage-scenarios. (дата звернення 25.01.2024).

32. What is Azure Table storage? URL: https://docs.microsoft.com/en-us/azure/storage/tables/table-storage-overview. (дата звернення 05.01.2024).

33. HTML. URL: https://developer.mozilla.org/ru/docs/Web/HTML. (дата звернення 11.02.2024).

34. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS. (дата звернення 11.02.2024).

35. Prepare data for building a model. URL: https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/prepare-data-ml-net. (дата звернення 12.01.2024).

36. Unified Modeling Language (UML) | Activity Diagrams. URL: https://www.geeksforgeeks.org/unified-modelinglanguage-uml-activity-diagrams/. (дата звернення 06.01.2024).

37. The Cold Start Problem for Recommender Systems. URL: https://yuspify.com/blog/cold-start-problem-recommendersystems/. (дата звернення 25.01.2024).

ДОДАТОК А

Лістинг коду розбиття набору даних на тестову та навчаючу множини

DataProcessor.cs

//функції LoadMovies, LoadRatings та LoadData

```
private List<Movie> LoadMovies(string filePath, bool hasHeader)
      {
         var data = File.ReadAllLines(filePath);
         List<Movie> movies;
         if (hasHeader)
         {
            movies = new List<Movie>(data.Length - 1);
            data = data.RemoveAt(0);
         }
         else
         {
            movies = new List<Movie>(data.Length);
         }
         foreach (var line in data)
         {
            var split = line.Split("");
            if (split.Length == 1)
            {
               split = split[0].Split(',');
            }
            else
            {
               if (split.Length > 3)
               {
                  split[1] = line.Substring(split[0].Length + 1,

                        line.Length - split[0].Length - split[^1].Length - 1);
               }
               split[0] = split[0].Substring(0, split[0].Length - 1);

               split[2] = split[^1].Substring(1);
            }
            movies.Add(new Movie { Id = float.Parse(split[0]), Title = split[1], Genres =
split[2].Split('|') });
         }
         return movies;
      }
      private void LoadRatings()
      {
         var dataPath = Path.Combine(Environment.CurrentDirectory, "Data", "ratings.csv");
         var ratingsData = _mlContext.Data.LoadFromTextFile<UserRating>(dataPath, hasHeader: true,
```

```
separatorChar: ',');
        _ratings = _mlContext.Data.CreateEnumerable<UserRating>(ratingsData, false).ToList();
    }

    public (IDataView training, IDataView test) LoadData()
    {
        var movieRatings = Ratings.Join(Movies, rating => rating.MovieId, movie => movie.Id,
         (rating, movie) => movie.Genres.Select(genre => new MovieRating
         {
            UserId = rating.UserId,
            MovieId = movie.Id,
            MovieTitle = movie.Title,
            MovieGenre = genre,
            Rating = rating.RatingValue
         })).SelectMany(movieRating => movieRating);

        var data = _mlContext.Data.LoadFromEnumerable(movieRatings);
        var split = _mlContext.Data.TrainTestSplit(data, 0.2);
        var training = split.TrainSet;
        training = _mlContext.Data.Cache(training);
        return (training, split.TestSet);
    }
```

ДОДАТОК Б

Лістинг Python-коду створення та навчання моделі FM

```
torch==1.10.0
torchmetrics==0.6.0
pytorch-accelerated==0.1.7
from pathlib import Path

import numpy as np
import pandas as pd
from statsmodels.distributions.empirical_distribution import ECDF
import matplotlib.pyplot as plt
!wget http://files.grouplens.org/datasets/movielens/ml-25m.zip
dataset_path = Path('ml-1m')
users = pd.read_csv(
    dataset_path/"users.dat",
    sep="::",
    names=["user_id", "sex", "age_group"],
    encoding='latin-1',
    engine='python'
)

ratings = pd.read_csv(
    dataset_path/"ratings.dat",
    sep="::",
    names=["user_id", "movie_id", "rating", "unix_timestamp"],
    encoding='latin-1',
    engine='python'
)

movies = pd.read_csv(
    dataset_path/"movies.dat", sep="::", names=["movie_id", "title", "genres"],
    encoding='latin-1',
    engine='python'
)
def get_last_n_ratings_by_user(
    df, n, min_ratings_per_user=1, user_colname="user_id", timestamp_colname="unix_timestamp"
):
    return (
        df.groupby(user_colname)
        .filter(lambda x: len(x) >= min_ratings_per_user)
        .sort_values(timestamp_colname)
        .groupby(user_colname)
        .tail(n)
        .sort_values(user_colname)
    )
def mark_last_n_ratings_as_validation_set(
    df, n, min_ratings=1, user_colname="user_id", timestamp_colname="unix_timestamp"
```

```
):

    df["is_valid"] = False
    df.loc[
        get_last_n_ratings_by_user(
            df,
            n,
            min_ratings,
            user_colname=user_colname,
            timestamp_colname=timestamp_colname,
        ).index,
        "is_valid",
    ] = True

    return df
train_df = ratings_df[ratings_df.is_valid==False]
valid_df = ratings_df[ratings_df.is_valid==True]
len(valid_df)
from torch.utils.data import Dataset

class UserItemRatingDataset(Dataset):
    def __init__(self, df, movie_lookup, user_lookup):
        self.df = df
        self.movie_lookup = movie_lookup
        self.user_lookup = user_lookup

    def __getitem__(self, index):
        row = self.df.iloc[index]
        user_id = self.user_lookup[row.user_id]
        movie_id = self.movie_lookup[row.title]

        rating = torch.tensor(row.rating, dtype=torch.float32)

        return (user_id, movie_id), rating

    def __len__(self):
        return len(self.df)
import math
from sklearn.metrics import mean_squared_error, mean_absolute_error

predictions = np.array([median_rating]* len(valid_df))

mae = mean_absolute_error(valid_df.rating, predictions)
mse = mean_squared_error(valid_df.rating, predictions)
rmse = math.sqrt(mse)

print(f'mae: {mae}')
print(f'mse: {mse}')
print(f'rmse: {rmse}')
```

ДОДАТОК В

Лістинг коду ініціалізації бота AdapterWithErrorHandler.cs

```
using Microsoft.Bot.Builder.Integration.ApplicationInsights.Core;
using Microsoft.Bot.Builder.Integration.AspNet.Core;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace MovieRecommenderBot {
   public class AdapterWithErrorHandler : BotFrameworkHttpAdapter {
      public AdapterWithErrorHandler(IConfiguration configuration,
         ILogger<BotFrameworkHttpAdapter> logger,
         TelemetryInitializerMiddleware telemetryInitializerMiddleware) :
         base(configuration, logger)
      {
         Use(telemetryInitializerMiddleware); OnTurnError =
         async (turnContext, exception) => {
            logger.LogError($"Exception caught : {exception.Message}");
            await turnContext.SendActivityAsync("Sorry, it looks like something went wrong."); };
      } }
}
```

ДОДАТОК Г

Лістинг коду обробників дій чат-бота Bot.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;

namespace MovieRecommenderBot.Bots {
   public class Bot<T> : ActivityHandler where T : Dialog {
      protected readonly Dialog Dialog;
      protected readonly BotState ConversationState;
      protected readonly BotState UserState;
          private readonly IStatePropertyAccessor<string> _userNameStateProperty;
          private readonly IStatePropertyAccessor<bool> _needToAskStateProperty;

      public Bot(ConversationState conversationState, UserState userState, T dialog) {
         ConversationState = conversationState;
         UserState = userState;
         Dialog = dialog;
         _userNameStateProperty = userState.CreateProperty<string>("UserName");
         _needToAskStateProperty = conversationState.CreateProperty<bool>("NeedToAsk");
      }

      protected override async Task OnMembersAddedAsync(
         IList<ChannelAccount> membersAdded,
         ITurnContext<IConversationUpdateActivity> turnContext,
         CancellationToken cancellationToken)
      {
         if (turnContext.Activity.ChannelId == "webchat")

         foreach (var member in membersAdded) {
            if (member.Id != turnContext.Activity.Recipient.Id) {
               await SendWelcomeMessageAsync(turnContext, cancellationToken); }
         } }
```

```
    protected override async Task OnConversationUpdateActivityAsync(
        ITurnContext<IConversationUpdateActivity> turnContext,
        CancellationToken cancellationToken)
    {
        await base.OnConversationUpdateActivityAsync(turnContext, cancellationToken);

        if (turnContext.Activity.ChannelId == "webchat" &&
            turnContext.Activity.MembersAdded?
                .FirstOrDefault(account => account?.Name == "MovieRecommenderBot")!= null) {
            await SendWelcomeMessageAsync(turnContext, cancellationToken); }
    }

    protected override async Task OnMessageActivityAsync(
        ITurnContext<IMessageActivity> turnContext,
        CancellationToken cancellationToken)
    {
        var message = turnContext.Activity.Text;

        if (await _needToAskStateProperty.GetAsync(turnContext, () => true, cancellationToken)) {
            await _userNameStateProperty.SetAsync(turnContext, message, cancellationToken);

            await _needToAskStateProperty.SetAsync(turnContext, false, cancellationToken);

            var reply = MessageFactory.Text($"Hi, {message}!");
            await turnContext.SendActivityAsync(reply, cancellationToken); }
        await Dialog.RunAsync(turnContext,
            ConversationState.CreateProperty<DialogState>(nameof(DialogState)),
cancellationToken);
    }

    public override async Task OnTurnAsync(ITurnContext turnContext,
        CancellationToken cancellationToken = default)
    {
        await base.OnTurnAsync(turnContext, cancellationToken);
        await ConversationState.SaveChangesAsync(turnContext, false, cancellationToken); await
        UserState.SaveChangesAsync(turnContext, false, cancellationToken);
    }
```

```
private async Task SendWelcomeMessageAsync(
    ITurnContext turnContext,
    CancellationToken cancellationToken) {
    await turnContext.SendActivityAsync(
        MessageFactory.Text("Hello and welcome to Movie Recommender!"),
        cancellationToken);

    await _needToAskStateProperty.SetAsync(turnContext, true, cancellationToken);

    var reply = MessageFactory.Text("Please, enter your name to login");

    await turnContext.SendActivityAsync(reply, cancellationToken); }
    }}
```

ДОДАТОК Д

Лістинг коду операцій з профілем користувача UserStorage.cs

```
using System.Collections.Generic; using
Microsoft.Azure.Cosmos.Table;
using MovieRecommender.DataModels;
using Newtonsoft.Json;

namespace MovieRecommender {
   public class UserStorage {
      private const string ConnectionString = "<secret>";
      private const string TableName = "Ratings"; private
      const string PartitionKey = "Users";
      private readonly CloudTable _tableClient;

      public UserStorage() {
         var storageAccount = CloudStorageAccount.Parse(ConnectionString);
         CloudTableClient tableClient = storageAccount.CreateCloudTableClient(new
TableClientConfiguration());
         _tableClient = tableClient.GetTableReference(TableName);
         _tableClient.CreateIfNotExists();
      }

      public void Create(string userId, IList<Rating> ratings) {
         var entity = new UserRatingEntity(userId) { Ratings = ratings };
         _tableClient.Execute(TableOperation.InsertOrReplace(entity));
      }

      public void Update(string userId, IList<Rating> ratings) {
         var retrieveOperation =
            TableOperation.Retrieve<UserRatingEntity>(PartitionKey, userId);

         var result = _tableClient.Execute(retrieveOperation); if
         (result.Result is UserRatingEntity entity)
         {
            var currentRatings = (List<Rating>)entity.Ratings;
            currentRatings.AddRange(ratings);
            entity.Ratings = currentRatings;
```

```
            _tableClient.Execute(TableOperation.Replace(entity));
        } }

    public bool UserExists(string userId) {
        var retrieveOperation =
            TableOperation.Retrieve<UserRatingEntity>(PartitionKey, userId);
        var result = _tableClient.Execute(retrieveOperation);
        return result.Result is UserRatingEntity;
    }

    public IEnumerable<Rating> GetUserRatings(string userId) {
        var retrieveOperation =
            TableOperation.Retrieve<UserRatingEntity>(PartitionKey, userId);
        var result = _tableClient.Execute(retrieveOperation);
        return (result.Result as UserRatingEntity)?.Ratings;
    }

    private class UserRatingEntity : TableEntity {
        public UserRatingEntity(){}
        public UserRatingEntity(string userId)
            :base(UserStorage.PartitionKey, userId) {}
        public string RatingsInternal { get; set; }

        [IgnoreProperty]
        public IList<Rating> Ratings {
            get => JsonConvert.DeserializeObject<IList<Rating>>(RatingsInternal); set
            => RatingsInternal = JsonConvert.SerializeObject(value);
        } }

} }
```

ДОДАТОК Е

Лістинг коду для отримання списку рекомендованих фільмів Predictor.cs

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Accord.Math.Distances;
using Microsoft.ML;
using MovieRecommender.DataModels;

namespace MovieRecommender.Services {
   public class Predictor {
      private readonly string _userRatingsPath; private
      Dictionary<int, double[]> _userRatings; private
      readonly DataProcessor _dataProcessor; private
      readonly MLContext _mlContext; private readonly
      ITransformer _model;

      public Predictor(MLContext mlContext, ITransformer model, DataProcessor dataProcessor) {
         _mlContext = mlContext; _model =
         model; _dataProcessor =
         dataProcessor;
         _userRatingsPath = _dataProcessor.UserRatingsPath; }

      public Dictionary<int, double[]> UserRatings {
         get {
            if (_userRatings == null) {
               _userRatings = new Dictionary<int, double[]>(); var
               data = File.ReadAllLines(_userRatingsPath);
               foreach (var str in data)
               {
                  var split = str.Split();

                  _userRatings[int.Parse(split[0])] = split.Skip(1).Select(double.Parse).ToArray(); }
            }
            return _userRatings; }
      }

      private int FindSimilarUserId(double[] currentUserRatings) {
```

```
    double maxSimilarity = 0.0; int
    userId = 0;
    var cosine = new Cosine();
    foreach (var userRating in UserRatings) {
        var similarity = cosine.Similarity(userRating.Value, currentUserRatings); if
        (similarity > maxSimilarity)
        {
            maxSimilarity = similarity;
            userId = userRating.Key;
        } }
    return userId; }

public IEnumerable<Recommendation> PredictTop5(IEnumerable<Rating> ratings) {
    ratings = ratings.ToList();
    var similarUserId = FindSimilarUserId(_dataProcessor.GetUserAllMoviesRatings(ratings)); var
    predictionEngine = _mlContext.Model
        .CreatePredictionEngine<MovieRating, MovieRatingPrediction>(_model); var
    watchedMoviesIds = ratings.Select(r => r.MovieId).ToList();
    return _dataProcessor.Movies.Select(movie => new Recommendation {
            Movie = movie,
            Prediction = predictionEngine.Predict(
                new MovieRating
                {
                    UserId = similarUserId,
                    MovieId = movie.Id
                })
        })
        .Where(moviePrediction => moviePrediction.Prediction.PredictedLabel
                            && !watchedMoviesIds.Contains((int)moviePrediction.Movie.Id))
        .OrderByDescending(moviePrediction => moviePrediction.Prediction.Score) .Take(5);
} }
```