ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
на ступінь вищої освіти магістр

на тему: «РОЗРОБКА МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З КОРИСТУВАЧАМИ НА БАЗІ МЕТОДІВ МАШИННОГО НАВЧАННЯ»

Виконав: студент 6 курсу, групи ПДМ-61
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Герасимчук М. В.
(прізвище та ініціали)

Керівник Бондарчук А. П.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерії програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного
забезпечення
Негоденко О.В.
__ _____ 20__року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Герасимчук Максим Володимирович

1. Тема роботи: «Розробка моделі інформаційної системи для взаємодії з користувачами на базі методів машинного навчання».
Керівник роботи Бондарчук Андрій Петрович, д.т.н., проф., директор Навчально-наукового інституту інформаційних технологій, затверджені наказом вищого навчального закладу від №122 12.10.2022р.

2. Строк подання студентом роботи 31.12.2022 р.

3. Вихідні дані до роботи:
   1. Рейтингові дані веб-сервісу MovieLens.
   2. Сервіси Azure Cognitive Services.
   3. Azure Bot Framework.
   4. Система розпізнавання природньої мови LUIS.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   1. Аналіз наявних рекомендаційних систем.
   2. Дослідження можливостей обраних засобів реалізації.
   3. Створення можелі рекомендаційної системи.
   4. Проектування та розробка бот-застосунку.

5. Графічна частина роботи представлена на 12 слайдах презентації:

1. Існуючі методи рекомендаційних систем.
2. Приклади існуючих рекомендаційних систем.
3. Схема створення моделі машинного навчання.
4. Діаграма діяльності процесу навчання моделі.
5. Оцінка результатів навченої моделі.
6. Діаграма діяльності та схема бот-додатку.

6. Дата видачі завдання «14» жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури	14.10.2022	Викон
2.	Підготовка обраних засобів реалізації	20.10.2022	Викон
3.	Створення та навчання моделі	25.10.2022	Викон
4.	Створення та розгорнення бот-застосунку	13.11.2022	Викон
5.	Вступ, висновки, реферат	15.11.2022	Викон
6.	Оформлення текстової частини	17.11.2022	Викон
7.	Оформлення графічної частини	17.12.2022	Викон
8.	Розробка презентаційної частини	22.12.2022	Викон
9.	Захист магістерської роботи	17.01.2023	

Студент _____ Герасимчук М.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бондарчук А.П.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська робота, 65 с., 32 рис., 26 джерел.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, РЕКОМЕНДАЦІЙНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, НАБІР ДАНИХ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ML.NET, ПРОБЛЕМА ХОЛОДНОГО СТАРТУ, БОТ-ДОДАТОК, ПРИРОДНЯ МОВА, AZURE, ХМАРНІ ТЕХНОЛОГІЇ.

Об'єкт дослідження – процес взаємодії з користувачами за допомогою рекомендаційної системи.

Мета роботи – покращення взаємодії з користувачами на основі розробленої моделі з використанням методів машинного навчання.

Методи дослідження – аналіз алгоритмів для побудови рекомендаційних систем, моделі машинного навчання для рекомендаційних систем, проектування, розробка та інтеграція бот-додатку із рекомендаційною системою.

Результатами є модель машинного навчання для рекомендаційної системи фільмів користувачам на основі їх уподобань і бот-додаток, що вміє розпізнавати природню мову користувача, інтегрований із рекомендаційною системою фільмів.

Область застосування – у сфері послуг, на інформаційних порталах, стрімінгові сервіси, соціальні мережі, тощо, для отримання рекомендацій в ході взаємодії з бот-додатком.

Робота пройшла апробацію на двох конференціях, а саме на XV Науково-технічній конференції «Сучасні інфокомунікаційні технології», та на Науково-практичній конференції «Проблеми комп’ютерної інженерії». Також за результатами дослідження опубліковано статтю у фаховому виданні Зв’язок.

ЗМІСТ

ВСТУП

Станом на сьогодні, левову частку нашого життя вже займають алгоритми та моделі навчання, у складі будь-яких інформаційних систем, і ця частка з кожним днем лише збільшується. Надскладні дослідницькі проекти, які розроблюються у сферах, від інформаційних технологій до біології та медицини, сфери дозвілля та надання різноманітних онлайн послуг, навіть розважальні додатки та ігри. Рекомендаційні системи є саме однією із цих сфер застосування машинного навчання, що доволі сильно інтегрована у суспільство практично на всіх рівнях. Сервіси онлайн-комерції, де на базі вподобань користувачів їм пропонується та чи інша веб-сторінка, до реклами саме тих продуктів, над якими останнім часом користувач задумувався, чи переглядав в інтернеті під свої потреби. Це все заслуги сучасних рекомендаційних систем, які щоденно розвиваються.

Рекомендаційні системи – це програми або ж цілі комплексні системи, які здатні робити прогнози щодо того, які продукти, наприклад, фільми, музика або товари в магазині сподобаються, або мають шанси того, що вони сподобаються користувачам. Конкретна інформація про користувача, його уподобання в якомусь продукті або категоріях товарів, різні соціально-демографічні характеристики, а також характеристики самого продукту, які можуть бути отримані найрізноманітнішим шляхом являються вихідними даними для таких систем.

Сьогодні рекомендаційні системи стали невід'ємною частиною багатьох онлайн-сфер. Вони присутні у таких галузях, як електронна комерція, соціальні мережі, стримінгові сервіси тощо. Це надзвичайно прибутковий інструмент, як в матеріальному або ж фінансовому плані, так і в інформаційному, надаючи велику кількість інформації про користувача, для дослідження соціальної поведінки. Ну і очевидно, що це дуже гарний спосіб виділитися серед значної конкуренції на сьогочасному ринку.

В свою чергу, реалізація рекомендаційних систем відбувається за

допомогою алгоритмів, що ґрунтуються на алгоритмах та моделях машинного навчання. Раніше, рекомендаційні системи на базі машинного навчання були обмежені конкретними мови програмування, як наприклад Python, та використанням специфічних бібліотек. Проте наразі вже є можливості застосовувати або створювати моделі машинного навчання, ґрунтуючись на мові програмування, яка є більш зручною для її розробника, або має функціональні переваги у конкретних ситуаціях. Однією із найпопулярніших мов програмування наразі є C#, котра використовується для розробки великої кількості різноманітних типів програмного забезпечення, під широкий спектр потреб. В результаті пошуків, було ML.NET як платформу для розробки, що задоволняэ всі необхідні потреби. Це середовище з відкритим вихідним кодом від компанії Microsoft, що розроблене для створення, навчання та оцінки моделей машинного навчання.

Боти – це програми, призначені для рішення завдань користувача через текстовий або графічний інтерфейс користувач-комп’ютер, де дії виконуються послідовно за запрограмованим сценарієм. Ключова відмінність ботів від інших програм полягає в універсальності, сумісності з різними платформами та системами. Проте головною перевагою є можливість імітувати собою людину, використовуючи голосові сервіси типу Text-to-Speech, чигенерацію текстових відповідей. Завдяки цьому досягається більш високий рівень спілкування з користувачами. Завдяки активному розвитку усіх цих сервісів стає складно під час спілкування, відрізняти віртуального помічника, від реальної людини-оператора. Здебільшого боти використовуються для автоматизації якихось циклічних, або рутинних завдань, таких як збір інформації, автоматизації підтримки системи тощо, що зменшує безпосередню участь людини у процесах. Людям психологічно закладено звикати, а згодом втомлюватися від монотонних дій, що негативно відображається як у фізичному плані, коли порушується концентрація та сповільнюється швидкість реакції, так і в моральному плані, коли падає настрій та зростає рівень стресу, що, тим самим лише погіршує ситуацію у фізичному плані, тоді як бот здатен виконувати ті ж

самі завдання без будь-якої для себе шкоди, при тому виконувати точніше, та швидше.

Бот-додатки стають все популярнішими завдяки своїй зручності та здатності взаємодіяти з користувачами як людина. Вони можуть надавати допомогу користувачам у режимі реального часу, що допомагає швидше досягти цілі. Наприклад, бот-додаток із рекомендаційною системою може бути інтегрований на сайті, та допомагати користувачам вибирати фільми. Користувачу не потрібно витрачати час на пошук серед каталогів – він може просто задати боту питання природньою мовою, і рекомендаційна система вибере найбільш відповідні фільми.

Саме через всі вищезазначені переваги та стрімкий ріст популярності, для інтеграції з розробленою рекомендаційною системою було обрано бот-додаток. Боти можуть бути використані не лише у якихось встановлених на комп'ютері користувальницьких додатках по типу Skype, Telegram, Discord чи інших, що, як правило, завжди під рукою, так і на сайтах, що ще більше спрощує їх розповсюдження через відсутність необхідноті щось встановлювати. Інтегрований із сайтом, бот із рекомендаційною системою надає змогу спростити користувачу пошук якогось фільму, просто звернувшись до програми із запитанням з приводу назви фільму, опису чи якогось із жанрів, і рекомендаційна система підбере найбільш підходящі йому кінокартини.

Метою роботи є побудова інформаційної системи з моделю машинного навчання, що генерує рекомендації щодо перегляду кінокартин користувачам на основі їхніх переваг, проектування і розробка бот-додатку з використанням хмарних технологій, вміння розпізнавати мову користувача, та інтеграція рекомендаційної системи до бот-додатку. Мовою розробки було обрано C#.

Були поставлені наступні задачі, а саме:

1) Аналіз наявних рекомендаційних систем.
2) Дослідження можливостей обраних засобів реалізації.
3) Створення моделі рекомендаційної системи.
4) Проектування та розробка бот-додатку.

Об'єктом дослідження є процес взаємодії з користувачами через бот-додаток.

Методи дослідження, застосовані в даній роботі, включають покращення взаємодії з користувачами за допомогою рекомендаційної системи на основі розробленої моделі з використанням методів машинного навчання.

1 АНАЛІЗ НАЯВНИХ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Огляд рекомендаційних систем у поточних реаліях

Для того, щоб користувачі найбільш ймовірно досягали своєї мети через використання програми, наприклад перегляду фільму, важливо, щоб продукт надавав допомогу користувачеві у виконанні їхніх потреб. Такий сервіс повинен розуміти потреби клієнтів і рекомендувати їм варіанти, які найбільш відповідають заданим критеріям. Користувачі більш ймовірно повернуться до сайту або сервісу, якщо вони змогли досягти своїх цілей з допомогою нього, не витрачаючи багато часу та зусиль.

Рекомендаційні системи розроблені для розв'язування вищеописаних завдань і широко використовуються різноманітними сервісами, брендами та компаніями. Візьмемо як приклад сферу електронної комерції. Платформа електронної комерції Rozetka надає категорії товарів такі як "Також вас може зацікавити" та "Купують разом" після того, як користувач перейшов на сторінку конкретного товару. Цей підхід допомагає збільшити прибуток організації. Це як приклад використання рекомендаційної системи у сфері електронної комерції і дана система специфічна саме для онлайн-магазинів. Такі системи відносять рекомендації лише до того товару, який користувач обрав чи обрав помилково, тому перехід на сторінку товару ще не означає, що користувачу сподобається цей товар.

Сервіс MovieLens - це онлайн-платформа, що пропонує своїм користувачам рекомендації фільмів, враховуючи їхні смаки та оцінки інших фільмів. Обрано було цей сервіс саме через її особливості, а саме:

1. Сервіс MovieLens дає користувачам можливість відкривати нові фільми, які сподобаються їхнім смакам, за допомогою постійно оновлених рекомендацій.

2. Налаштування рекомендацій у сервісі MovieLens враховує не тільки улюблені жанри фільмів, але і оцінки користувачів до фільмів, які вони

переглянули раніше.

3. Рекомендації, які користувач отримує в сервісі MovieLens, є актуальними і відповідають реальним оцінкам фільмів, відповідно до їхніх смаків і попередньо оцінених фільмів.

Чат-боти - це програми, що створюють інтерактивний контент, який дозволяє користувачам знаходити інформацію, отримувати консультації та інші послуги на веб-сайті або в месенджері. Сучасні чат-боти можуть розпізнавати текстові повідомлення від користувачів і відповідати на них у більш природному і розумному стилі, ніж раніше. Таким чином, чат-боти надають зручний і швидкий спосіб отримання інформації та різних послуг без необхідності використовувати спеціальні програми або відкривати веб-сайти. І саме тому, чат-боти стали невід'ємною частиною інтернету і важливим інструментом для електронного бізнесу чи інших сфер діяльності.

Вирішено було спроектувати бот-додаток, що інтегрується з рекомендаційною системою фільмів, заснованою на машинному навчанні, який дозволяє комбінувати особливості рекомендаційних систем та зручність та поширеність бот-додатків. Середовище ML.NET найкраще підходить для навчання та оцінювання моделей. Дані для навчання моделі не обов'язково потрібно створювати від нуля - є можливість використання існуючих даних та моделей, які можна редагувати за специфічні потреби, використовувати як прототипи або вихідні точки для створення нової моделі.

Ще однією з переваг зазначеного середовища є його здатність експортувати модель у формат TensorFlow чи ONNX та використовувати її у додатках не тільки на платформі .NET, але і в інших середовищах, хмарних сервісах з різними мовами програмування. Однією із проблем, що зустрічається у рекомендаційних системах, є так званий холодний старт, який може бути рішеним шляхом пошуку існуючого, активного користувача зі схожим набором вхідних даних, на якому була навчена модель, що знімає необхідність у постійному перенавчанні моделі і є великою перевагою у порівнянні з іншими рекомендаційними системами.

Azure Bot Service – це платформ, яка дозволяює створювати, вбудовувати та оперувати чат-ботами у різних месенджерах і веб-сайтах, а також платформа містить сервіс Bot Framework для безпосередньо розробки бот-застосунку. Ці сервіси зручні у використанні та мають можливість інтегрорувати розроблювані додатки на веб-сайти.

Для того, щоб бот-застосунок міг розпізнавати природню мову користувача, а не лише запрограмовані командні набори, було вирішено інтегрувати сервіс розпізнавання мови LUIS.

В результаті, бот-додаток з обраними сервісами може бути інтегрований як на веб сайт у вигляді невеликого вікна чату, так і в будь-який додаток, наприклад, WhatsApp, Telegram або Viber, що дозволяє не обмежувати користувача в рамках певного сервісу, та надає йому можливість використовувати послуги бота там, де йому зручніше за все.

Рекомендації будуть надаватися у формі списку з п'яти найбільш відповідних фільмів, які можна оцінити, при цьому із наступних рекомендацій вже оцінені фільми будуть виключені, щоб уникнути повторення. Для нових користувачів передбачено базові рекомендації у вигляді списку з двадцяти найпопулярніших фільмів, які були відібрані за середнім арифметичним всіх оцінок, проставлених користувачами. Після оцінки деяких з цих фільмів, система рекомендацій зможе створити персоналізовані рекомендації, основані на навченій моделі машинного навчання.

1.2 Типи рекомендаційних систем

Основна ідея рекомендаційних систем полягає в тому, що якщо користувачі обирають схожі товари при купівлі в інтернет-магазин, слухають подібну музику або дивляться одні й ті ж самі фільми, то ймовірність того, що користувачі і надалі будуть вести себе подібно – доволі висока. Тобто, рекомендаційної системи оцінюють вподобання користувачів щодо товарів, сервісів або іншої продукції, та роблять прогноз щодо того, що іншим

користувачам це теж може бути до вподоби.

Рекомендаційні системи проводять аналіз двома різними способами:

- На скільки продукт подобається користувачам, засновуючись на тому, що людям з схожими перевагами також сподобаються певні речі.
- На скільки продукти популярні серед користувачів, незалежно від наявності або відсутності у них схожих переваг.

На основі вищезазначених підходів постають основні типи рекомендаційних систем, а саме колаборативна фільтрація, фільтрація на основі змісту та гібридна фільтрація.

Фільтрація на основі змісту є аналізом додаткової інформації у форматі “користувачы та продукти” або “користувачі або продукти”, які були раніше вже оцінені користувачами. Головна ідея цього методу полягає у прийому доступних характеристик об’єкта, через які видно взаємодії користувачів з об’єктом, та побудові на основі цих даних моделі. Результатом цієї операції є утворення так званих груп взаємодії. Наприклад, фільми із жанрами «бойовик», «спорт» або «жахи» більше подобається чоловічій аудиторії, тоді як фільмам із жанрами «комедія», «психологія» та «романтика» надає перевагу жіноча частина аудиторії.

Колаборативна фільтрація є одним із методів направленого рекомендаційного системного аналізу, який використовує інформацію про попередні вибори та переваги користувачів для визначення рекомендацій для них. Цей метод переважає переваги користувачів, що мають схожі уподобання, і намагається знайти схожість між їх попередніми виборами і рекомендувати нові продукти, які можуть їм сподобатися.

Основна перевага цього методу - то, що чим більше користувачів взаємодіють, тим точнішими будуть рекомендації, оскільки їхні дії допомагають уточнювати переваги користувачів. Однак, одним з недоліків колаборативної фільтрації є те, що при старті роботи системи у нових користувачів може не бути рекомендацій, що називається "холодним стартом". Також, новий продукт

може не мати рекомендацій, так як у користувачів не було взаємодій з ним.

Для вирішення проблеми, є наступні способи:

- пропонувати популярні продукти відповідним користувачам, або ж пропонувати товари з найбільшою кількістю взаємодій;
- згенерувати список із випадково обраних товарів, та запропонувати його користувачеві;

Загалом, існує два основні підходи колаборативної фільтрації: колаборативна фільтрація на базі пам’яті, та колаборативна фільтрація, заснована на моделі.

Підходи колаборативної фільтрації, засновані на пам'яті, запам'ятовують взаємодії користувачів із продуктами, а потім використовують цю інформацію для знаходження подібностей між користувачами. Один із таких підходів - підхід на основі нормалізації. Він полягає у нормалізації взаємодій користувачів із продуктами, перетворенні їх у оцінки і побудові різноманітних матриць співвідношень.

Підходи колаборативної фільтрації, засновані на моделі, спробують знайти правила, що описують взаємодії користувачів із продуктами, і використовують ці правила для генерування рекомендацій. Один із таких підходів - підхід, заснований на матричному розкладанні.

Рекомендаційні системи з гібридною фільтрацією - це спеціальні алгоритми, які поєднують у собі фільтрацію на основі змісту та колаборативну фільтрацію, щоб створити якомога більш точні рекомендації. Це дозволяє аналізувати інформацію про попередні оцінки користувачів та прогнозувати рейтинги, які вони можуть поставити для різних об'єктів, або ж вибрати для конкретного користувача, k найкращих об'єктів із вибірки. Гібридна фільтрація потребує більше ресурсів, але може запропонувати більш точні результати.

1.3 Проблема «Холодного старту»

Проблема холодного старту або як її ще називають проблема нового

користувача – це неспроможність системи надати базові рекомендації для нового користувача, що як правило відбувається через через брак необхідної інформації, яка власне і дає змогу формувати рекомендації.

У рекомендаційних системах «холодний старт» означає, що система не має спроможності правильно оцінити нового користувача, та скласти можливі результати. Проблема холодного старту розгалуджується на холодний старт для продукту, та, в свою чергу - холодний старт для користувача.

Проблема холодного старту для продукту виникає у випадку, коли нові товари, які були додані на веб-сайт, мають недостатньо взаємодії з користувачами, і через це не мають сформованої логіки для рекомендації цих товарів користувачам. Для алгоритму колаборативної фільтрації це виливається у неможливість рекомендувати товар саме через те, що алгоритм базується на взаємодіях користувачів та товарів.

Якщо продукт не матиме достатню кількість взаємодій з різними користувачами, для можливості сформувати базову систему надання рекомендацій, тоді в такому випадку система не буде знати, коли можна буде пропонувати його користувачеві. З цього випливає наступне: чим новий продукт має взаємодій, тим ефективніше система зможе складати точні персоналізовані рекомендації.

Проблема холодного старту для продукту, характерна для систем, які в своєму роді представляють собою онлайн магазини, аукціони, тематичні сайти тощо. Як правило, на подібних сайтах продукти будуть додатково сортовані на основі дати публікації, де вище будуть виводитися свіжі продукти чи оголошення. Це збільшує ймовірність того, що користувачі будуть більше взаємодіяти саме з новими продуктами, формуючи цим самим необхідні дані для створення персоналізованих рекомендацій, які вже будуть таргетовано поширюватися по користувачам, як для тих хто взаємодіяв з цими продуктами, так і для наступних користувачів зі схожими потребами.

Холодний старт для користувача виникає тоді, коли відвідувач щойно

зареєструвався в системі або на веб-сайті. Такий користувач ще не надав ніяких відомостей про себе, не має історії пошуку товарів, а також не має ніяких взаємодій з товарами. В такому випадку рекомендаційна система не може відпрацьовувати на користувачеві, та створювати для нього персональні рекомендації.

В такому випадку, якийсь час рекомендаційна система повинна мати можливість, надавати користувачеві рекомендації, які були сформовані не на використанні минулих дій та історії користувача, щоб уникнути помилок при формуванні рекомендацій, чи формування неправильних ассоціативних ланцюжків для логіки персональних рекомендацій.

Одна із тактик для роботи з новими користувачами – це здобуття від користувача інформації, як базової, такої як вік, стать або регіон проживання, до інформації про їхні вподобання, які жанри вони люблять, а які уникають, цим самим закріплюючи якісь початкові дані про користувача. Так формується початкова точка, від якої система може почати свою роботу по створенню рекомендацій. Інформація може здобуватися як поза сайтом, коли система отримує доступ до профілю користувача в різних соц-мережах, якщо звісно він під час реєстрації обирав пункт, який створює профіль із прив’язкою до соціальної мережі, так інформація може отримуватися під час самої реєстрації, коли користувач вказує основну та додаткову інформацію про себе.

Як перший крок вирішення проблеми холодного старту для користувача, є використання стратегії, яка заснована на популярності продукту. З усіх продуктів робиться вибірка, де головним критерієм буде оцінка від інших користувачів. Тобто формується певний рейтинг товарів.

При цьому допускається варіація рейтингів. Наприклад може бути сформований глобальний рейтинг товарів – де продукти йдуть по градації оцінки за весь час, та може бути сформований рейтинг, де береться вибірка товарів, градація яких йде за оцінкою, але лише за останній час, щось накшталт категорій типу «Популярно зараз», варіації яких можна бачити в разноманітних онлайн магазинах.

Наступним кроком йде звуження кола інформації, що буде відображатися новому користувачеві. Наприклад відображати лише найбільш рейтингові товари, які беруться із сформованих раніше рейтингів.

Після цього користувач вже може оцінювати запропоновані йому товари, де оцінка буде базуватися на різноманітних факторах. Цими факторами може бути клік по товару, тривалість часу, проведеного на сторінці товару, додавання товару в закладки або навіть кошик. Кількість факторів можна вар'ювати взалежності від технічної можливості платформи, що є доволі гнучким фактором при сучасних інформаційних технологіях.

В результаті, після виконання даних кроків, за деякий проміжок часу від реєстрації нового користувача, система вже може зібрати базову інформацію, якої буде достатньо для формування початкових рекомендацій, які вже будуть згодом покрауватися під час подальших взаємодій користувача з іншими продуктами.

У нашому випадку ситуація виглядає аналогічним чином. Коли користувач вперше використовує сервіс або бот-додаток, то даних про нього, його історію перегляду, пошуку та рекомендацій не існує. Невідомо, які фільми або жанри йому подобаються, які фільми користувач вже переглянув або тільки збирався подивитися. У цьому випадку рекомендаційна система не може робити якісь прогнози по користувачу через брак початкових даних для формування рекомендацій. Основна стратегія роботи з новим користувачем в такому випадку – це під час реєстрації, надати користувачу можливість надати певну базову інформацію про себе, свої смаки та вподобання тощо, для створення базового профілю, щоб система початкові вхідні дані для роботи з цим користувачем. Діаграма діяльності даної стратегії наведена нижче на рисунку 1.1. Таким чином після входу нового користувача до програми повинен бути запит на надання деяких початкових даних через форму анкети, після заповнення якої буде представлений список найбільш популярних фільмів для початкової оцінки.

Рисунок 1.1 – Діаграма діяльності з новим користувачем

З набору даних MovieLens складається перелік фільмів, по яким здійснюється проходження системою, після чого відбувається групування та сортування за суммарним рейтингом. Отриманий список зберігається для подальшого використання, щоб надалі для нових користувачів не повторювати все спочатку. Достатньо буде лише періодично оновлювати рейтинговий список фільмів. Як приклад, на рисунку 1.2 представлений рейтинговий список фільмів.

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 1.2 – Вибірка фільмів з найбільшим рейтингом

Головними полями являються унікальний ідентифікатор фільму, його назва, та жанри фільму, записані та розділені чере спецсимвол «|».

Після того, як користувачу буде виведений список із двадцяти найпопулярніших фільмів, буде запропоновано оцінити ці фільми. Статистично відомо, що найбільш рейтингові фільми на одному сайті, швидше за все будуть високо оцінені і на інших майданчиках, тож доволі часто у різних списках популярних фільмів, буде багато спільних кінострічок. Це наштовхує на те, що під час оцінки фільмів, користувач швидше за все вже міг бачити якісь із цих фільмів, і тому зможе їх оцінити. Таким чином, користувач оцінює представлені

фільми, виставляючи оцінки від одного до п’яти. Оцінювати всі двадцять фільмів не є обов’язковим. Користувач після оцінювання п’ятого фільму може пропустити подальшу оцінку, якщо забажає, і цього буде достатньо, щоб рекомендаційна система могла зробити початковий прогноз, та створити перші персональні рекомендації щодо уподобань для користувача.

Розв’язання проблеми холодного старту, дозволяє для вирішення поставлених задач використовуати алгоритми колаборативної фільтрації.

1.4 Алгоритми колаборативної фільтрації

Колаборативна фільтрація можна поділити на два основні види, колаборативну фільтрацію відносно користувачів, та колаборативна фільтрація відносно відносно продукту. Колаборативна фільтрація відносно користувачів забезпечує рекомендації, засновані на порівнянні схожих переваг користувачів, незалежно від того, яку інформацію ми маємо про їхній профілі. Колаборативна фільтрація відносно продуктів, з другого боку, забезпечує рекомендації на основі схожості продуктів з тими, що вже сподобалися користувачеві.

Найбільш поширеною є колаборативна фільтрація щодо користувачів, оскільки дуже часто інформації про користувачів або недостатньо, або взагалі немає, тож зручніше використовувати тільки наявні дані про продукти.

Першочергово, для колаборативної фільтрації необхідно знайти найбільш схожу по вподобанням до вихідного користувача, людину. Другим кроком буде оцінка уподобань знайденого користувача, щоб передбачити ймовірну оцінку, яка може бути проставлена користувачем до відповідного продукту. Якщо оцінка буде високою, це означає, що продукт може бути рекомендований вихідному користувачеві.

Рейтинги можуть бути представлені у вигляді двухвимірної матриці, де кожен рядок представляє собою користувача, а кожен стовпець — певний фільм, як наприклад в ситуації, де користувач дає оцінку фільмам. Так як не всі користувачі могли бачити певні фільми, та не всі користувачі оцінюють фільми

після перегляду – дані в більшості своїй будуть розрідженними. Дану матрицю можна також розглядати як похідну матриць меншої розмірності, тобто використати підхід матричного розкладання, або ж факторизацію. Але постає проблема, адже при цьому включення якихось додаткових характеристик, таких як жанр, рік випуску, головні актори тощо, неможливе, і в результаті матриця може бути представлена у вигляді матриці користувачів та фільмів.

Машини факторизації являють собою покращену версія матричного розкладання. Тоді як даний підхід дозволяє імітцвати більшість модель взаємодії, через використання ознак об’єктів, тоді як матричне розкладання – це просто конкретний випадок машини факторизації. В даному випадку, взаємодія користувача з об'єктом подається у вигляді вектора, отриманого під час прямого унітарного кодування, при цьому кожен перетворений рядок в матриці буде мати тільки одну одиницю у векторі, що означає взаємодію одного користувача та одного об'єкта. Потім додаються додаткові ознаки, наприклад жанр фільму, рік випуску тощо, які також перетворюються або нормалізацією векторів, або за допомогою того ж самого унітарного кодування. Таким чином, за допомогою машин факторизації відбувається оцінка взаємодії у розріджених даних і їх факторизація. Це означає, що дані для області взаємодії одного користувача і об'єкта, дають змогу також оцінити параметри для пов'язаних із ними взаємодій.

Як альтернатива – можна використовувати машини факторизації зі специфікою поля, що також э способом реалізації колаборативної фільтрації. Полями тут виступають незалежні змінні, такі як, наприклад, жанр фільму, а значення в цих полях, наприклад жанр комедія, або жанр бойовик, то вони звуться категоріальними значеннями.

Кожній ознаці у машинах факторізації властиво мати лише один вектор, котрий визначає його взаємодію з іншими ознаками. Як приклад, можна взяти взаємодію користувача з жанром фільму комедія, та якимось конкретним головним актором. Машина факторизації з урахуванням специфіки поля розбиває один вектор на кілька – взаємодію користувача із жанром, та взаємодію користувача із актором. Таким чином утворюються нові вектори, такі

як взаємодія користувача із жанром комедія, взаємодія користувача із автором, взаємодія жанру із автором тощо.

Таким чином, підумувавши можна сказати, що машини факторизації, зберігаючи ефективність звичайних машин факторизації, можуть представляти взаємодії на детальнішому рівні, при тому використовуючи менше ознак, що, особливо для розріджених наборів даних великого об’єму, та із великою кількістю характеристик буде значно краще та ефективніше.

1.5 Метрики якості бінарної класифікації

Завдання бінарної класифікації полягає в тому, що необхідно спрогнозувати та визначити, якій фільм може бути рекомендований користувачу, чи яка ймовірність того, що рекомендованій фільм сподобається користувачеві. Ймовірності того, що фільми можуть бути високо оцінені, складаються для кожного користувача, де потім, з якогось порогово значення, після якого фільм може вважатися хорошим, створюються рекомендації для користувача.

З усього вищезазначеного випливає, що машина факторизації з урахуванням специфіки поля дає змогу вирішити задачу із підготовкою даних, а саме перетворенням ознак завдяки прямому унітарному кодуванню або нормалізації векторів, та перетворенням фінальних рейтингів в бінарний тип даних, де буде зазначено лише те, чи може фільм бути рекомендований, чи ні.

2 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ОБРАНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Середовище ML.NET

ML.NET – це середовище з відкритим вихідним кодом, розроблене для навчання, створення та оцінки моделей машинного навчання.

Машинне навчання в ML.NET є центральним елементом, що дозволяє навчати нові користувальницькі моделі, а також імпортувати попередньо навчені моделі. Також є можлиість експорту моделі у формат файту ONNX, для можливості використовувати модель в інших проектах та середовищах, відмінних від .NET.

ML.NET підтримується операційною системою Windows при наявності встановленного .NET Framework, а також системами Linux і macOS зі встановленним .NET Core.

Платформа надає кілька основних алгоритмів машинного навчання для вирішення широкого спектру завдань, а саме :

- Регресія – використовується для прогнозування значення по набору пов'язаних ознак. Наприклад, прогнозування зміни ціни на товар виходячи з деяких основних його показників.
- Класифікація або категоризація – містить у собі бінарну та багатокласову класифікації. Дає змогу вирішити, до якого з двох або більше класів, або категорій належить об'єкт. Можна використовувати для визначення типу об’єкта на фото, наприклад марки машини, або ж для поділу відгуків на сайті на позитивні та негативні.
- Кластеризація – використовується для групування об'єктів у кластери по подібним ознакам та характеристикам.
- Виявлення аномалій – використовується для вирішення таких завдань як виявлення операцій, що потенційно є шахрайськими, або

створення навчальних шаблонів у сфері кібербезпеки.

Починається процес створення моделі з підбору початкового набору даних, які використовуватимуться для навчання моделі, та відправляються у середовище. Там, залежно від обраного алгоритму машинного навчання, дані перетворюються на необхідний тип. Далі відбувається навчання моделі та її оцінка. Навчання моделі це доволі комплексна та ресурсомістка задача, тож необхідно мати або дуже потужні характеристики комп'ютера, або ж використовувати потужні обчислювальні кластери, наприклад Google Cloud Platform, Amazon Web Services або ж Azure Cloud Services.

Нижче, на рисунку 2.1 зображено схему процесу створення моделі в ML.NET та можливості її подальшого використання.

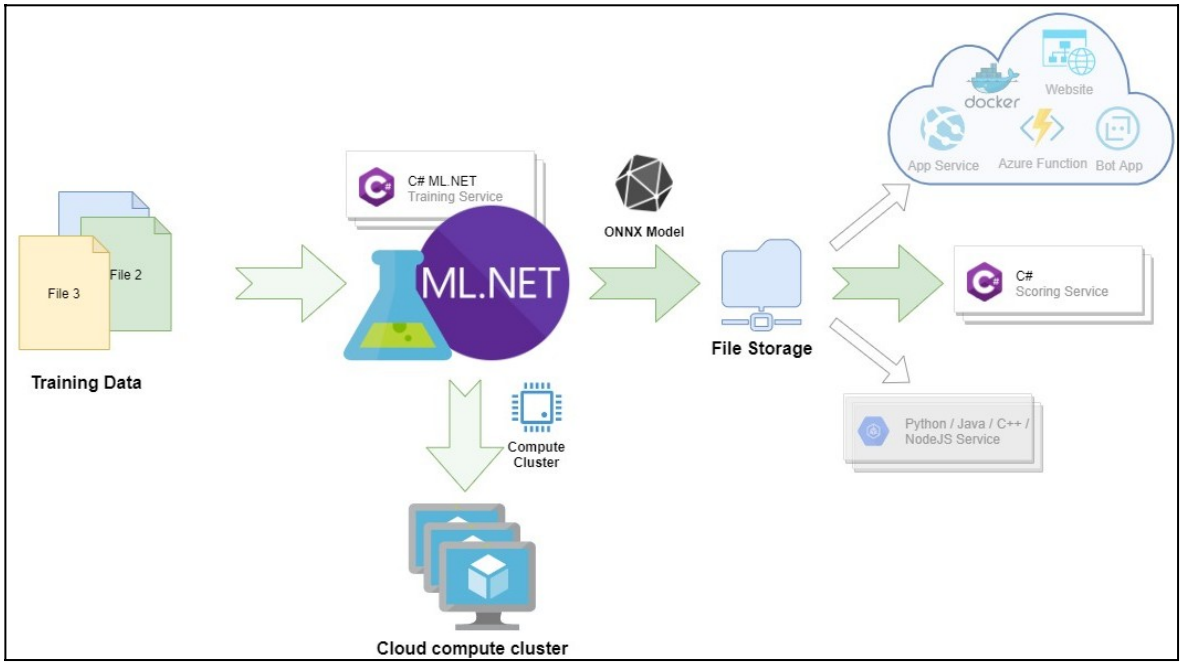


Рисунок 2.1 – Схема створення моделі в ML.NET та її використання

Як вже згадувалося, ML.NET дозволяє імпортувати готові моделі, а також експортувати навчену модель у відповідні формати TensorFlow та ONNX.

TensorFlow — бібліотека машинного навчання із відкритим вихідним кодом, розроблена компанією Google. Дана бібліотека використовується для вирішення завдань класифікації образів через побудову та навчання нейронної мережі.

ONNX – це стандартний формат для представлення моделей машинного навчання, який також має відкритий вихідний код. Перевага даного формату полягає у його сумісності, що дозволяє зберігати та експортувати навчені моделі у даному форматі на одних платформах, та використовувати їх на великій кількості інших платформ.

Експортовані моделі ONNX можна не лише переносити та використовувати поміж платформами. Також моделі можна використовувати в хмарних сервісах, так як переважна більшість наразі вже має підтримку даного формату, а якщо і з’являються нові сервіси, то вони також орієнтуються на один із найпоширеніших форматів моделей.

Таким чином, навчену в ML.NET модель, можна використовувати для створення прогнозів на різних пристроях, операційних системах, платформах – словом будь-де, де підтримується ONNX Runtime механізм виводу для моделей машинного навчанняв ONNX.

Процес машинного навчання в ML.NET починається з MLContext – об’єкта, котрий містить інші каталоги для завантаження і збереження даних, функцій перетворення і навчання даних, а також компонентів операцій моделі. Кожен об'єкт каталогу має свої методи для створення різних типів цих компонентів. Наприклад, каталог DataOperationsCatalog призначений для завантаження, кешування і збереження даних, за перетворення даних перед навчанням відповідає TransformsCatalog, ModelOperationsCatalog для збереження, завантаження моделі та створення прогнозів.

На рисунку 2.2 представлений процес побудови моделі машинного навчання у вигляді діаграми діяльності.

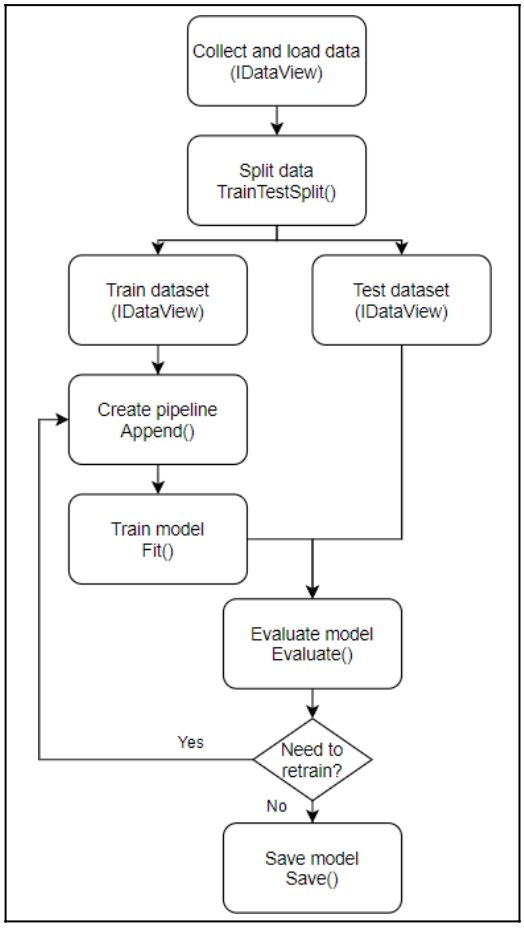


Рисунок 2.2 – Процес навчання моделі

Спочатку відбувається збір та завантаження даних для навчання в об’єкт IDataView, з подальшим розподілом даних на навчальний та тестові набори у заданоу користувачем, співвідношенні. Далі створюється конвеєр із включенням до нього даних, які були перетворені у необхідний формат, з обраними ознаками, а також обраний алгоритм машинного навчання. Потім відбувається власне сам процес навчання моделі, який ініціюється через виклик команди конвеєра, Fit. Після відбувається оцінка моделі, та за необхідності повторення попередніх кроків. Коли оцінка моделі відповідає необхідним критеріям, модель зберігається у бінарному форматі для подальшого використання. Процес використання моделі відображений на рисунку 2.3 у вигляді діаграми діяльності.

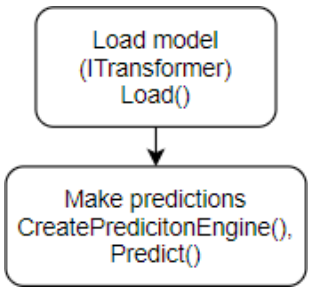


Рисунок 2.3 – Процес використання моделі.

Як видно із наведеної діаграми, щоб використати модель необхідно лише два кроки, це завантаження в об’єкт ITransformer, та етап прогнозування, який включає в себе виконання методу CreatePredictionEngine, що на виході дає нам об’єкт PredictionEngine. Цей об’єкт дає змогу робити прогнози на екземплярі тестових даних. І як завершальний крок – це виконання методу Predict, куди власне і передаються необхідні для прогнозування тестові дані, які використовує PredictionEngine.

2.2 Сховище таблиць Azure Table storage

Для будь якої програми необхідне сховище, в якому можна зберігати опрацьовувані дані, інформацію по користувачам, тимчасову інфу тощо.

Сховище таблиць Azure дозволяє зберігати нереляційні структуровані дані, або структуровані дані NoSQL, в хмарному сховище без необхідності заздалегідь створення структури БД.

Головними перевагами даного сервісу є швидкість доступу та ресурсоефективність. Також, перевагою є те, що через відсутність схем табличного сховища, дані можна адаптовувати в процесі розвитку додатку та його потреб.

Для наборів даних у форматі однієї таблиці, без зовнішніх ключів та складних міжелементних з’єднань, найкраще всього підходить саме сховище таблиць Azure, яке до того ж може бути денормалізоване для швидкого доступу до потрібної інформації за допомогою бібліотек .NET через запити LINQ та

протокол OData.

Для доступу до сервісу Azure Table storage необхідно оформити підписку Azure та Storage account, створений на порталі Azure. Створений профіль зображений на рисунку 2.4.

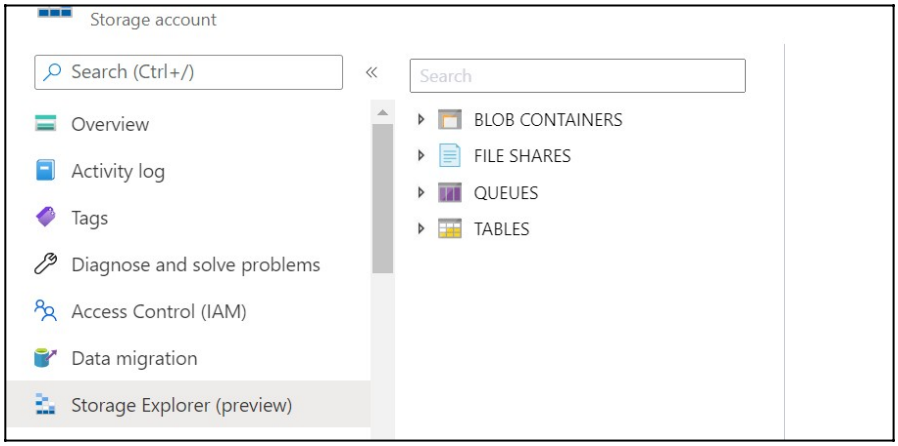


Рисунок 2.4 – Профіль користувача на порталі Azure

Для початку роботи з Azure Table storage, необхідно на порталі у розділі Access keys взяти спеціальний рядок підключення та записати його у файл програми. Також, для можливості роботи сервісу в додатку, спершу необхідно встановити спеціальний пакет Microsoft.Azure.Cosmos.Table, що створює об'єкт класу CloudTableClient, який є відповідальний за виклик та виконання всіх операції з таблицями та їхнім управлінням.

Щоб визначити суть, яка буде представляти об'єкт, що зберігається в таблиці, необхідно створити клас, що успадковується від класу TableEntity. Ключем тут буде використовуватися ім'я користувача, а ключем мітки всіх користувачів – виступає значення "Users". Це дозволяє швидко ідентифікувати в таблиці конкретного користувача. Збережені в таблиці, властивості сутностей із доступними властивостями класу, повинні мати змогу підтримувати як зчитування, так і запис значень. У класі UserRatingEntity, що відповідає за сутність оцінок користувачів така властивість лише одна – це властивість RatingsInternal, що має строковий тип і отримує своє значення через

серіалізацію списку JSON об’єктів, які належать класу Rating, що містить у собі властивості Id_Movie — тобто унікальний ідентифікатор фільму, та змінну RateVal — яка представляє собою число від одного до п'яти, виставлене до відповідному фільму.

Всі операції з даними здійснюється через метод Execute, що є об'єктом класу CloudTableClient, де в якості параметра передається об'єкт класу TableOperation. Метод InsertOrReplace використовується для додавання в таблицю інформації про нового користувача, або ж цей метод можна використовувати для оновлення інформації про вже існуючі рейтинги.

2.3 Azure Bot Service і Azure Bot Framework

Боти – це програми, за допомогою яких, як правило через дручний для користувача інтерфейс, можна вирішувати найрізноманітніші завдання. Через активний розвиток та все більшу інтеграцію бот-програм у суспільство, взаємодія з ботом стає все більш схожою на взаємодію з людиною, з використанням не запрограмованих ключ команд, а звичайною природньою мовою. Робота з файлами, використання та робота з базамми даних, підключення різних API тощо, це все також входить у сферу діяльності ботів. Боти корисні у в тому плані, що вони дозволяють економити інтеллектуальні, часові та фізичні ресурси шляхом автоматизації процесів, які їм потрібні здійснити в ході використання програми.

Розрізняють три основні види ботів. Це чат-боти, коммерційний боти та шкідливі-боти.

Чат-бот – найбільш поширений тип ботів, головне завдання якгого, це виконання запрограмованих задач через ведення діалогу з користувачем. Як правило, це відбувається у форматі діалогу, де користувач ставить системі питання, а система надає відповіді, задаючи зустрічні, уточнюючі питання. Дані боти використовуються в різноманітних сферах. Як правило, це служби підтримки, системи пошуку інформації, прогноз погоди, система для пошуку

роботи або пошуку співробітників тощо.

Комерційні боти приваблюють клієнтів і роблять розсилку новин та акційних пропозицій компаній.

Шкідливі ж роботи застосовуються для шахрайства з метою заробітку, дезинформації або отримання конфіденційної інформації користувача. Відомі також випадки, коли боти застосовувались для кординації мережевих атак на різноманітну інфраструктуру.

Azure – це хмарна платформа, на якій можливо розмістити існуючі програми і спростити розробку нових додатків. Azure інтегрує хмарні послуги, необхідні для розробки, тестування, розгортання і управління додатками, використовуючи ефективність хмарних обчислень. Головний портал Azure в вою чергу, надає можливість керувати усіма підключенимими сервісами, налаштовувати та відстежувати стан використаних ресурсів. Також, для можливості використання на інших платформах, системах та пристроях, сервіси Azure надають REST API.

Для створення, тестування, управління бот-застосунках в єдиному середовищі використовуються сервіси Azure Bot Service та Bot Framework, які надають засоби для вищезазначених дій, а також можливість опублікувати його в Azure. Бот також може бути підключений до таких інормаційно-комунікаційних систем, як Viber, Skype, Telegram тощо. Bot Framework виконує більшу частину роботи, необхідної для відправлення та отримання повідомлень з усіх цих різних платформ, в свою чергу бот отримує опрацьований та нормалізований потік незалежно від кількості каналів до яких він підключений.

2.4 Azure Cognitive Services

Azure Cognitive Services – це спеціальне API, призначене для використання при створенні інтелектуальних додатків без безпосереднього використання штучного інтелекту. Когнітивні сервіси розділені на різні категорії. Це додавання API пошуку Bing для забезпечення додатком доступу

до пошукової мережі та інформції. Це сервіси перетворення мови в текст і навпаки, розпізнавання природньої мови за допомогою створених сценаріїв. Також Azure Cognitive Services мають можливість надавати сервіси розпізнавання та ідентифікації фотографій та відео, так званий «комп'ютерний зір».

2.5 LUIS – Сервіс розпізнавання природньої мови

Language Understanding (LUIS) – це хмарний сервіс, який за допомогою штучного інтелекту, на основі тексту на природній мові, розпізнає сенс тексту та окремих фраз, та прогнозує можливе продовження тексту. Представляє собою LUIS діалоговий додаток, який взаємодії з користувачем природньою мовою. Приклади клієнтських додатків включають чат-ботів, меседжери або цифрові версії популярних настільних програм, де розпізнавання мови незамінне через варіативність можливостей користувача.

Процес використання LUIS починається з того, що додаток користувача надписає текст повідомлення користувача в API LUIS як запит. Сервіс LUIS застосовує моделі машинного навчання до отриманого тексту, і повертає відповідь у форматі JSON. Відповідь містить текст запиту та найбільш ймовірний варіант наміру, та можливо деякі додаткові дані. Далі додаток використовує відповідь, та робить прогноз щодо того, як виконувати запит.

Основою моделі LUIS є так звані наміри – тобто категорії введеного тексту. Для того, щоб сервіс міг розпочати процес навчання, йому необхідно мати якісь початкові дані. Цими вхідними даними слугує початковий список прикладів фраз.

3 СТВОРЕННЯ МОДЕЛІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

3.1 Огляд використаних даних

Як набір даних для моделі машинного навчання рекомендаційної системи фільмів було використано зібраний набір фільмових рейтингових даних із веб-сайту MovieLens дослідницької лабораторії GroupLens, що з 1992 року реалізувала безліч дослідницьких проектів, в таких областях, як:

- рекомендаційні системи;
- онлайн-спільноти;
- мобільні технології;
- електронні бібліотеки;

Набір даних про фільми збирався з січня 1995 року та до вересня 2018 року, та представляє собою більш ніж двадцять п’ять мільйонів об’єктів з п'ятизірковим рейтингом, які представляють собою оцінку близько шестидесяти тисяч фільмів, яку поставили приблизно триста тисяч користувачів.

Всі файли з даними представлені у форматі CSV, в якому роздільником даних є кома, а дані, які вже містять її в собі, для уникнення помилок, поміщені у подвійні лапки.

Всі користувачі, кожен з яких, задля уникнення витіку даних, представлені анонімними ідентифікаторами, та оцінили як мінімум один фільм.

Всі фільми, які мають оцінку користувача або тег, включені до набору даних, яким являється фійл movies.csv, структура якого представлена на рисунку 3.1.

Інформація про кожен фільм складаэться з унікального ідентифікатора, назви та жанр\ів фільма. Кожен фільм має один, але як правило більше, жанрів, які розділені символом "|". Як приклад, можуть бути наступні жанри: Комедія, Бойовик, Романтика, Аніме, Анімація, Документальний фільм, Фентезі, Жахи, Містика, Триллер, Наукова фантастика.

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children...
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (199...	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

Рисунок 3.1 – Структура файлу movies.csv

Структура даних рейтингів представляє собою ідентифікатор користувача, ідентифікатор фільму, оцінку фільма користувачем та часовий штамп, UNIX-формат часу, або ж кількість секунд від 1.01.1970 року. Вся інформація про рейтинги міститься у файлі ratings.csv. Файл та його структура зображені на рисунку 3.2.

userId	movieId	rating	timestamp
1	2840	3.0	1256677500
1	2986	2.5	1256677496
1	3020	4.0	1256677260
1	3424	4.5	1256677444
1	3698	3.5	1256677243
1	3826	2.0	1256677210
1	3893	3.5	1256677486
2	170	3.5	1192913581
2	849	3.5	1192913537
2	1186	3.5	1192913611
2	1235	3.0	1192913585
2	1244	3.0	1192913551

Рисунок 3.2 – Структура файлу ratings.csv

Ідентифікатор фільму також можливо використовувати для посилання на інші джерела з даними про фільм. Наприклад, якщо як авторизований користувач, виконати запит на сервіс https://movielens.org, додавши в кінці /movies/ плюс ідентифікатор фільму, наприклад 218, то в результаті запиту

https://movielens.org/movies/218 отримаємо фільм та дані про нього. Приклад зображений на рисунку 3.3.

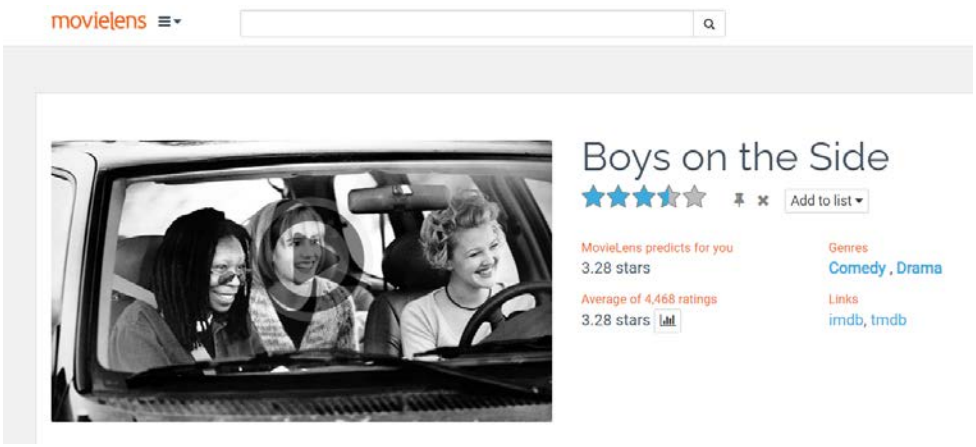


Рисунок 3.3 – Інформація про фільм на сайті https://movielens.org

Як вже згадувалося, в даній роботі використовуються два файли: ratings.csv та movies.csv. Ці файли завантажуються та поєднуються в єдиний набір даних через методи LoadRatings і LoadMovies відповідно, які реалізовані в класі DataProcessor, результат роботи який, зображений нижче на рисунку 3.4.

userId	movieId	rating	title	genres
1	307	3.5	Three Colors: Blue (Trois coule...	Drama
1	481	3.5	Kalifornia (1993)	Drama|Thriller
1	1091	1.5	Weekend at Bernie's (1989)	Comedy
1	1257	4.5	Better Off Dead... (1985)	Comedy|Romance
1	1449	4.5	Waiting for Guffman (1996)	Comedy
1	1590	2.5	Event Horizon (1997)	Horror|Sci-Fi|Thriller
1	1591	1.5	Spawn (1997)	Action|Adventure|Sci-Fi|Thriller
1	2134	4.5	Weird Science (1985)	Comedy|Fantasy|Sci-Fi
1	2478	4.0	iThree Amigos! (1986)	Comedy|Western
1	2840	3.0	Stigmata (1999)	Drama|Thriller

Рисунок 3.4 – Інформація про фільми та їх рейтинги

Далі, для даних про фільми необхідне розв’язання задачі з денормалізації жанру. Під час вирішення задачі, кожен рядок, що містить список жанрів, в свою чергу, розбивається на окремі рядки. Приклад денормалізації жанру представлений на рисунку 3.5.

userId	movieId	rating	title	genres
4	1	4.0	Toy Story (1995)	Adventure
4	1	4.0	Toy Story (1995)	Animation
4	1	4.0	Toy Story (1995)	Children
4	1	4.0	Toy Story (1995)	Comedy
4	1	4.0	Toy Story (1995)	Fantasy

Рисунок 3.5 – Результат денормалізації жанру

3.2 Створення та навчання моделі

3.2.1 Підготовка даних

Після завантаження дані діляться на навчальний і тестовий набори, де 75% тренувальних даних, та 25% тестових даних, та через вбудовану функцію TrainTestSplit відбувається поділ на набори.

Як правило, під час обробки даних, алгоритми навчання можуть їх завантажувати і циклічно проходити по ним велику кількість разів, для точнішої обробки даних та виключення деяких помилок зчитування, що в результаті породжує необхідність застосовувати кешування, для зменшення кількості звернень до диску. Це збільшує загальну ефективність, швидкість навчання, та зменшує відсоток зношення жорсткого диску. Через вбудовану функцію Cache відбувається кешування навчальних даних, та проводиться одразу після поділу даних.

У машинному навчанні стовпці, використані для прогнозування, називаються ознаками, а стовпці з поверненим прогнозом називаються міткою. У даному випадку , ознаками будуть виступати ідентифікатор користувача, назва і жанр фільму, а міткою буде виступати виставлений рейтинг.

Алгоритми машинного навчання ML.NET вимагають того, щоб вхідні дані або об'єкти знаходилися в одному числовому векторі, що призводить до необхідності перекодування значень міток.

Категоріальні дані є найбільш поширеним типом даних та представляють собою скінченне число категорій. Наприклад, категоріальними даними

виступають жанри фільмів. Дані повинні бути зіставлені з деяким числовим значенням для того, щоб їх можливо було використовувати для створення моделі машинного навчання.

Пряме унітарне кодування зіставляє кінцевий набір значень до цілих чисел, які в двійковому поданні мають значення один або нуль. Це зручно, коли немає необхідності зберігати упорядковування, тож не важливо, у якому порядку стоять жанри фільмів.

В результаті, всі стовпці з ознаками були спочатку перетворені за допомогою прямого унітарного кодування, а потім із декількох стовбців, поєднані в один, під назвою Features. Стовпець з рейтингом було перетворено в елементи з булевим типом даних, які були перекодовані при умові, що якщо рейтинг фільму більше за 3,5, то в такому випадку, фільм може бути рекомендований, при рейтингу нижче – не може бути рекомендований.

Описані вище перетворення були реалізовані з допомогою вбудованою функції OneHotEncoding для прямого унітарного кодування і CustomMapping – довільного відображення даних, що визначається класом CustomAction, для перетворення мітки до типу bool.

3.2.2 Навчання моделі

Після вибору алгоритму машинного навчання з каталогу, який надає середовище ML.NET, відбувається виклик методу Fit з передачею їй в якості параметра отриманого раніше набору тренувальних даних.

Всередині кожного каталогу є спеціальні розширюючі методи для додавання етапів обробки даних у навчальний конвеєр. Раніше до конвеєра були додані методи для довільного відображення даних і прямого унітарного кодування, а також об'єднання кількох ознак в один стовпець Features. В останню чергу, в конвеєр імпортується обраний алгоритм навчання, а саме машини факторизації з урахуванням специфіки поля. Клас FieldAwareFactorizationMachine представляє собою сам алгоритм для реалізації та знаходиться в каталозі алгоритмів бінарної класифікації.

Після створення об'єктів у конвеєрі дані можна використовувати для навчання моделі. Функція Fit після свого виконання, повертає навчену модель, яка являє собою об'єкт типу, що реалізує інтерфейс ITransformer. Тобто Модель перетворює вхідні дані в прогнози.

3.3 Оцінка отриманих результатів

Щоб мати змогу створити найбільш ефективну модель, потрібно після завершення її навчання, оцінити її продуктивність за тестовими даними, щоб при незадовільних результатів, не було пізно підкорегувати модель, дані або алгоритм, для отримання кращого результату. Для оцінки використовується функція Evaluate, яка дозволяє виміряти різні метрики для навченої моделі. Отримані метрики будуть залежати від того, яке завдання машинного навчання було поставлене до виконання.

Для створення прогнозів на тестовому наборі даних, викликається метод Transform, в який передається навчена модель машинного навчання. Таким чином дані спочатку підготовлюються в необхідному форматі, після чого формуються тестові прогнози.

Як тільки прогнози отримані, метод Evaluate оцінює модель, яка порівнює спрогнозовані значення з мітками тестового набору даних і повертає метрики по тому, наскільки ефективно працює модель.

В результати, всі отримані метрики повертаються як об’єкт класу CalibratedBinaryClassificationMetrics. Це відбувається через те, що об’єкти машини факторизації з урахуванням специфіки поля розміщені в каталозі бінарної класифікації. Всі доступні метрики представлені рисунку 3.6. Для зручності та можливості подальшого порівняння, всі метрики експортуються в файл eval_metrics_log.txt.

```
Evaluation Metrics
Accuracy: 0,78
Area Under Roc Curve: 0,86
Area Under Precision Recall Curve: 0,86
F1 Score: 0,78
Negative Precision: 0,78
Negative Recall: 0,77
Positive Precision: 0,77
Positive Recall: 0,78
Confusion matrix: TEST POSITIVE RATIO:  0,5006 (7530500,0/(7530500,0+7511742,0))
Confusion table
          ||=====================
PREDICTED || positive | negative | Recall
TRUTH     ||=====================
 positive || 5 910 246 | 1 620 254 | 0,7848
 negative || 1 741 051 | 5 770 691 | 0,7682
          ||=====================
Precision ||   0,7725 |   0,7808 |
```

Рисунок 3.6 – Метрики навченої моделі

Протягом процесу створення та навчання моделі вона зберігається в пам'яті середовища та доступна під час роботи програми. Однак коли програма припиняє своє виконання, модель перестає бути доступна. Тому, для подальшого використання моделі у інших проектах, додатках чи середовищах, модель необхідно зберегти через виклик методу Save. В результаті виконання методу, в робочій директорії проекту залишиться навчена модель, запакована у формат zip, для зручності переміщення та подальшого використання в інших додатках.

4 ПРОЕКТУВАННЯ ТА РОЗРОБКА БОТ-ЗАСТОСУНКУ

4.1 Проектування користувальницької програми

Взаємодія користувача та рекомендаційної системи відбувається наступним чином.

Спочатку користувач вводить своє ім’я. Це може бути як звичайне ім’я, так і вигаданий псевдонім або якесь слово, так званий логін або нікнейм. Після відправки логіну, система перевіряє, чи є такий користувач в базі. Якщо такий користувач існує – користувач потрапляє у головне меню, де він може викликати три основні команди: отримати рекомендації, викликати довідку, та вийти з програми. Команди можуть бути сформовані звичайним текстом, як у випадку, якщо користувач розмовляв з живою людиною. Якщо користувач обирає перший варіант, то система спочатку завантажить дані з його профілю: за раніше створеними рейтингами система визначає вподобання. Далі йде пошук користувача зі схожими вподобаннями, та після аналізу, зіставляється список рекомендацій, які найбільш ймовірно можуть сподобатися користувачеві. Всі ці кроки актуальні також і для нового користувача, за винятком того, що у випадку нового користувача, фільми будуть сформовані лише базуючись на вподобаннях знайденого користувача, так як у цільового користувача ще не має сформованого профілю. Далі система просить оцінити фільми, рейтингові дані з яких зберігаються в його профіль, і наступного разу допоможуть в складанні точніших рекомендацій. Якщо обраний другий варіант, із виведенням довідки, то користувачеві виводиться приклад запиту, за допомогою якого він може сформувати запит на отримання рекомендацій. Якщо ж обраний третій — здійснюється вихід із програми. Для нового користувача також доступний другий та третій варіанти, відрізняється лише початкові кроки для формування списку рекомендацій, та проходження початкового заповнення профілю, яке складається зі списку питань щодо вподобань користувача. Після такого анкетування, користувач може вводити раніше згадані команди, та

алгоритм дій повторюється.

Для того, щоб відобразити послідовність взаємодій рекомендаційної системи та користувача, було створено діаграма діяльності, яку наведено на рисунку 4.1.

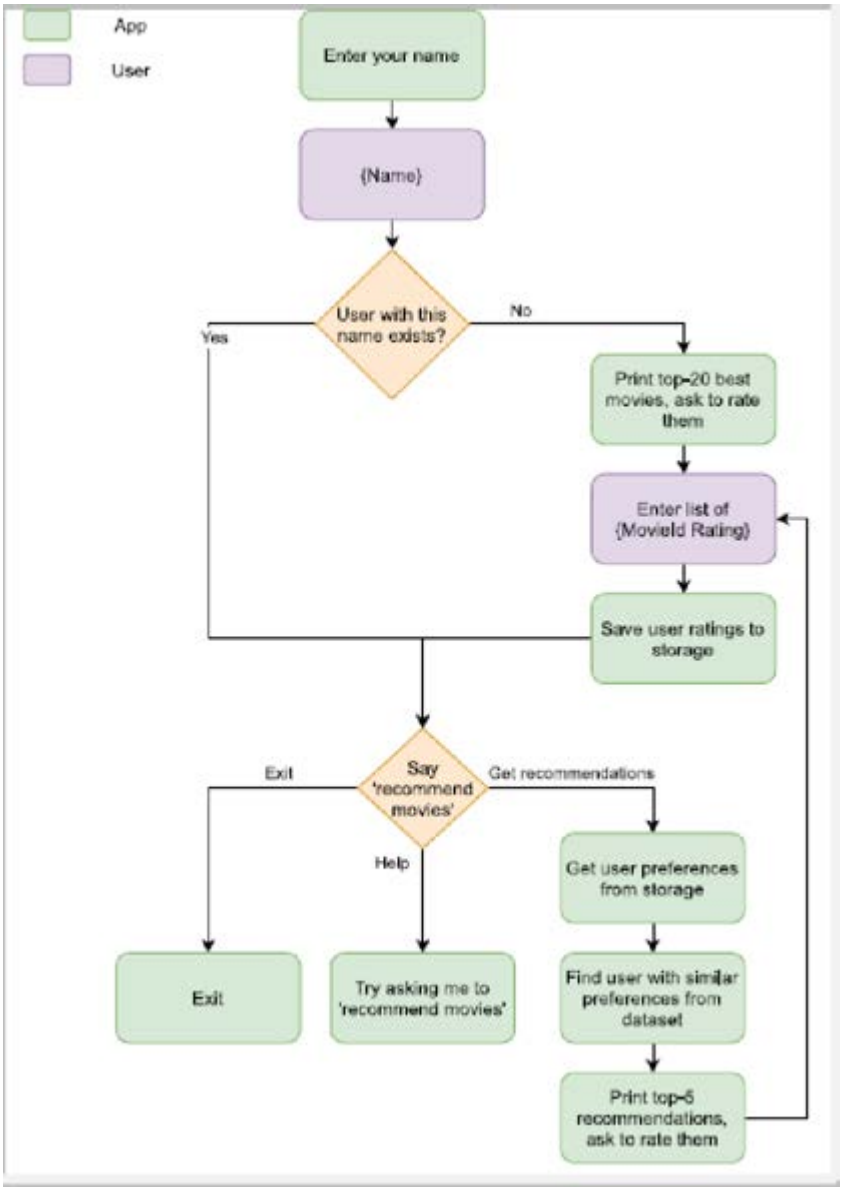


Рисунок 4.1 – Діаграма діяльності взаємодій користувача із рекомендаційною системою

Оскільки набір даних MovieLens містить інформацію про надзвичайно велику кількість фільмів, ще більшу кількість даних про користувачів, то зберігання вектора з рейтингами на всі фільми в матриці уподобань, де, у вектор

записується поставлене користувачем значення, або ж записується 0 при умові, якщо значення відсутнє, буде надзвичайно ресурсовитратно, як у вигляді великого розміру файлів, так і по кількості операцій читання\запису з диску, що в свою чергу доволі сильно зношує накопичувачі. Незважаючи на те, що запис у файл буде здійснюватися один раз під час підготовки даних, зчитування необхідних даних з файлу буде необхідно під час входу користувача до програми і запиті їм рекомендацій, а довге очікування для користувача неприпустимо.

Для вирішення проблеми, було вирішено групувати фільми за так званою «схожістю», в якості показника якого було обрано жанр фільму. В результаті групування, для кожного користувача, розмірність вектору на виході була не в кілька тисяч, а всього лише близько два десятки, що дорівнює більшості жанрів фільмів які існують. Якщо користувач вже оцінив якийсь фільм, то поставлений йому рейтинг сумується в компонент обраного вектора, який відповідає певному жанру кінострічки. При цьому якщо фільм має не один, а кілька жанрів, то рейтинг підсумовується до кожного з цих компонентів.

На виході є матриця, рядки якої – це вподобання користувачів, а стовпці – вектор оцінок користувачів для кожного жанру. Очікується, що матриця буде розрідженою, адже багато користувачів можуть не оцінювати фільми які переглянули по різним причинам. Для отримання ступеня схожості, найбільш ефективно буде використання косинусної подібності.

Потенційно різні шкали рейтингу фільмів, які можуть використовуватися з різних джерел, можуть впливати на результати оцінки. Тому щоб запобігти неточностям, використовується косинусна подібність та нормалізація векторів. Це дозволяє отримати вектори, чиї компоненти є речовими числами, що перебувають в діапазоні від 0 до 1. Отримана матриця переваг зберігається у файлі user_ratings.csv. Кожен рядок починається з ідентифікатора користувача, а далі наведено його вектор переваг.

Користувачам, що раніше використовували додаток, не потрібно проходити анкетування, адже вони вже поставили рейтинги фільмам з переліку

двадцяти найкращих у системі при реєстрації. Нові користувачі, отримавши перелік з двох десятків найкращих фільмів та проставивши їм рейтинги, в результаті вже надали достатньо інформації для того, щоб рекомендаційна система могла скласти прогнози, і вони можуть перейти до наступного кроку.

При розпочатому процесі рекомендації, спочатку завантажується інформація про оцінки фільмів, які користувач виставив раніше, з бази даних Azure Table storage. Для цього використовується метод Retrieve, який належить класу TableOperation, що дозволяє знайти перелік рейтингів фільмів за допомогою ключа (імені або логіну користувача). Ця інформація необхідна для того, щоб запустити процес прогнозування.

Для управління профілем користувача був створений клас UserProfile, який використовує об'єкт класу UserStorage для безпосереднього взаємодії з Azure Table storage. Отриманий список рейтингів надсилається до методу PredictTop5 класу Predictor, який був створений для реалізації функціоналу, напряму пов'язаного безпосередньо з прогнозуванням.

В методі PredictTop5 виконується пошук ідентифікатора користувача, що найбільш схожий на необхідного, що визначається на основі схожості уподобань у фільмах.

У методі PredictTop5 виконується перетворення рейтингів користувача на вектор переваг, який визначається так само, як це було зроблено для кожного користувача в наборі даних MovieLens під час створення файлу user_ratings.csv. Якщо фільм отримав оцінку від користувача, то вектору присвоюється значення рейтингу, інакше – нуль, після чого відбувається нормалізація вектору. Далі йде визначення косинусної подібності для отриманого вектора та кожного вектора із файлу user_ratings.csv.

У методі PredictTop5 використовується платформа Accord.NET та її бібліотека Accord.Math для обчислення косинусної подібності двох векторів через структуру Cosine і її метод Similarity. У результаті знаходиться користувач, чий вектор переваг найбільш схожий на вектор переваг вихідного користувача. Далі для цього користувача обчислюється список рекомендацій,

який виглядає як вибірка із п'яти фільмів, які найбільш ймовірно сподобаються вихідному користувачу, які він ще не оцінив, за допомогою раніше навченої моделі ML.NET.

Локально збережена модель може бути використана у інших програмах. Для її завантаження слід використати функцію Load, передаючи шлях до zip-файлу з навченою моделлю як параметр. За необхідності, з файлу movies.csv завантажуються більш детальні дані про фільми.

Для роботи з прогнозами використовується об'єкт класу PredictionEngine з універсальними параметрами, що вказують на тип тестових даних MovieRating та на тип прогнозу MovieRatingPrediction. Для здійснення прогнозу використовується метод Predict, який приймає вхідні дані як аргумент.

Для виведення найкращих п'яти фільмів для користувача, метод Predict викликається в циклі для кожного фільму з файлу movies.csv, та створюється об’єкт класу MovieRating, що ініціалізований ідентифікаторами користувача та фільму, і робиться прогноз для цього об'єкту. Після цього об'єкти сортуються за спаданням властивості Score у об'єкті Prediction отриманого прогнозу, та вибираються п'ять перших елементів з тих фільмів, що користувач ще не дивився.

Після виведення рекомендованих фільмів користувачу надається можливість оцінити їх шкалою від одного до п'яти. Це допомагає розширити базу даних системи із відомостями про уподобання користувача, що покращує точність подальших прогнозів. Основною перевагою створеної рекомендаційної системи є те, що чим частіше користувач заходить у додаток та запитує нові рекомендації, а потім надає зворотну відповідь системі щодо сподобання або несподобання запропонованих фільмів, тим точніші стануть рекомендації у наступний раз.

4.2 Розробка та інтеграція бот-додатку

На рисунку 4.2.1 показана фінальна архітектура бот-додатку, в якій

відображені сам бот, та залежні зв'язки із зовнішніми сервісами та службовими компонентами.

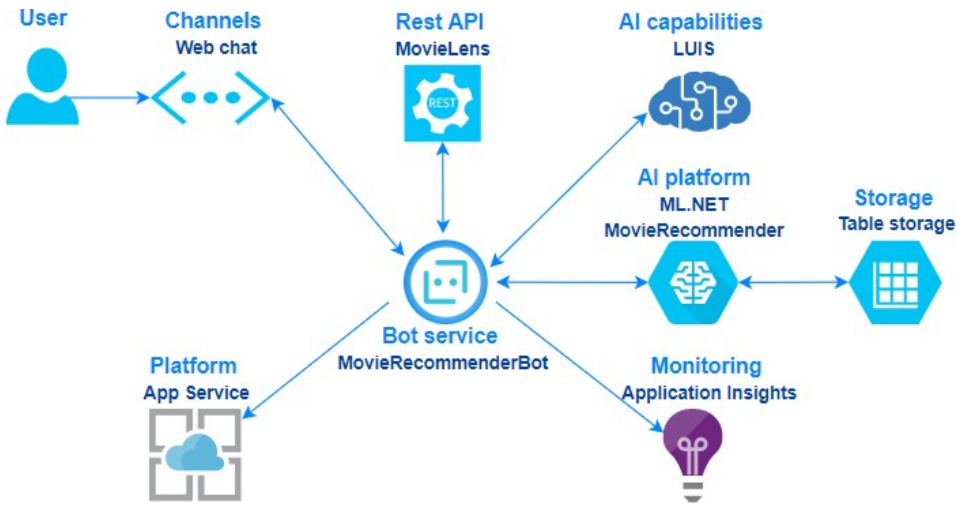


Рисунок 4.2.1 – Загальна схема бот-додатку

Бот-додаток розгортається та виконується за допомогою Azure App Service, та відправляє дані телеметрії в компонент App Insights, які можуть бути використані для аналізу роботи програми та відслідковування стабільної роботи.

Взаємодія компонентів бота проходить таким чином:

1. Користувач не підключається напряму до боту, а з ним взаємодіє через канал, такий як Web Chat, Email, Skype тощо.

2. Бот - це веб-сервіс, що надає API /api/messages для отримання вхідних повідомлень від користувача та відповідей на них.

3. За допомогою каналу і бот-додатків, користувач може запитати рекомендації.

4. Бот використовує сервіс розпізнавання природної мови LUIS для аналізу тексту запиту, а потім інтегрується з рекомендаційною системою, яку навчив ML.NET.

5. Для зберігання інформації про уподобання користувача (виставлені рейтинги) використовується Azure Table storage.

Бот завантажує постери фільмів за допомогою MovieLens API для

інтерактивного відображення рекомендацій. Бот є додатком, який користувачі можуть використовувати для обміну текстом, графікою (наприклад, інтерактивними картками або зображеннями) або мовою у розмовному режимі. Azure Bot Service використовує сервіс Bot Framework для передачі інформації між додатками користувачів, які також називаються каналами, і ботом.

Протягом розмови люди зазвичай взаємодіють, по черзі здійснюючи свій «хід». В сервісі Bot Framework поняття «ходу» використовується для опису вхідної дії користувача, що ініційується при зверненні до бота, і відповідної дії, що відправляється ботом. Таким чином, «хід» представляє собою пару дій: дія користувача і відповідна дія бота. Таким чином, "хід" - це процес обробки дії, що відбувається у розмові. Дія надходить у вигляді HTTP-запиту, що представлений у вигляді JSON-об'єкту, і передається до класу адаптера бота, який є унаслідуваним класом BotFrameworkHttpAdapter. Клас AdapterWithErrorHandler, реалізує код адаптера. Адаптер ініціалізує контекст "ходу" та додає до нього інформацію про саму дію, включаючи відправника, одержувача, дату відправлення, ідентифікатор каналу та інші необхідні дані. Ініціалізований контекст "ходу" передається до класу робота для подальшої обробки.

Клас бота, який є унаслідуваним класом ActivityHandler і реалізує інтерфейс IBot, включає у себе обробники дій, які визначені в ньому, які наслідуються з методу OnTurnAsync, який є базовим обробником, що ще має назву "обробник ходу". Вся обробка дій у розмові починається з виклику цього методу, а потім, в залежності від типу отриманої дії, викликається індивідуальний метод обробки. Наприклад, коли користувач відправить будь-яке повідомлення, він цим самим ініціював дію, тож базовий обробник, тобто наш “обробник ходу” перенаправить цю дію до методу AsyncOnMessageActivity, який відповідає за обробку відправлених повідомлень. Дія, яка генерується при додаванні нового користувача до діалогу, передається в метод AsyncOnMembersAdded, який відповідає за обробку дій, що стосуються додавання нових користувачів.

У бота, що підключається до каналу Web Chat, необхідно перевизначити метод AsyncOnConversationUpdateActivity, який викликається, коли користувач додається або видаляється з діалогу. Якщо користувач додається, то викликається метод AsyncOnMembersAdded, який, в свою чергу, викликає метод AsyncSendWelcome класу Bot. Цей метод виводить вітальне повідомлення для користувача і запитує ім'я (рисунок 4.2.2).

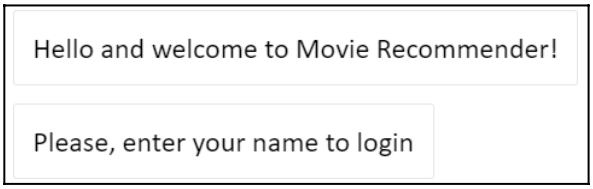


Рисунок 4.2.2 – Процес авторизації в бот-застосунку

Діалоги слугують для того, щоб керувати розмовою з користувачем і виконувати певні завдання у певному порядку. Можна викликати діалоги у відповідь на повідомлення користувача, відповідь на деякі зовнішні події, або інші діалоги, або на низку інших вбудованих функцій, які надає бібліотека діалогів.

Каскадні діалоги дозволяють боту отримувати інформацію за кроками і передавати її на наступні кроки, що дозволяє спрямувати розмову з користувачем і виконувати певні завдання у порядку, як в нашому випадку:

1. Бот отримує ім'я користувача і виводить вітання, та перевіряє, чи вже є дані про користувача у системі. Якщо дані вже є, бот переходить до наступного кроку. Якщо ні, бот завантажує двадцять найбільш популярних в системі фільмів і виводить їх (рисунок 4.2.3).

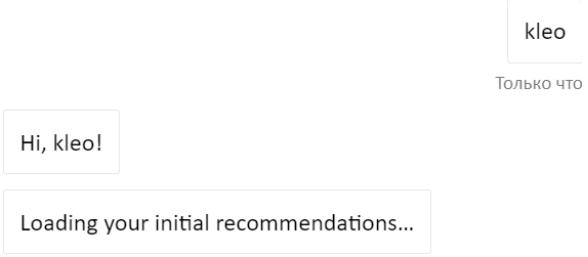


Рисунок 4.2.3 — Початкові повідомлення бот-застосунку

2. Якщо користувач є новим у системі, бот очікує, що користувач оцінить запропоновані фільми. Для цього викликається альтернативний діалоговий ланцюг (рисунок 4.2.4). Якщо користувач не хоче оцінювати фільми, бот переходить до наступного кроку.

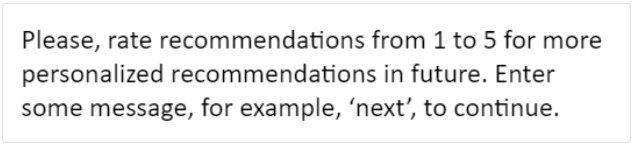


Рисунок 4.2.4 – Запит оцінити запропоновані фільми

3. Якщо користувач є новим у системі, бот зберігає виставлені рейтинги у системі. Бот також виводить сповіщення із прикладом можливого запиту на команду отримання будь-яких рекомендацій, що зображено на рисунку 4.2.5.

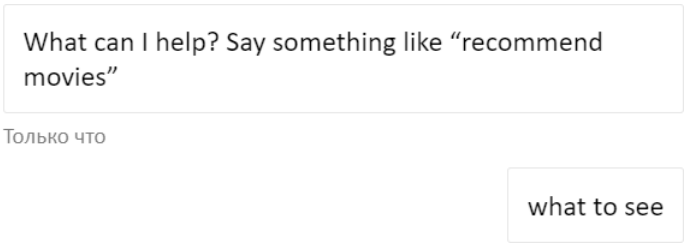


Рисунок 4.2.5 – Запит вибору подальшої дії

4. Якщо користувач зробив запит на рекомендації, бот виводить рекомендації користувачу (рисунок 4.2.6), а також виводить повідомлення з проханням оцінити рекомендовані фільми після виведення рекомендацій.

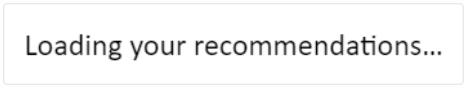


Рисунок 4.2.6 – Повідомлення о завантаженні рекомендацій

5. Бот зберігає виставлені рейтинги у системі та пропонує свої послуги

користувачу знову.

Діалоги введення є способом запитувати у користувача деяку інформацію і отримувати відповідь. Бібліотека діалогів надає набір методів, які можуть бути використані для різних типів введення. В нашему розробленому додатку ми використовуємо текстовий запит TextPrompt разом із об'єктом PromptOptions. У цьому об'єкті є поле Prompt, яке містить питання для користувача. Користувач має відповісти будь-яким текстовим повідомленням.

Сервіс Bot Framework надає форматовані картки, які допомагають удосконалити досвід користування користувачами. Це можливо через надання їм інформації у вигляді списку або каруселі інтерактивних карток, а не просто текстових повідомлень. Це дозволяє користувачам отримувати інформацію в більш зручному форматі.

Клас Activity представляє собою дію в діалозі з користувачем та містить властивість Attachments, яка зберігає масив об'єктів Attachment. Ці об'єкти Attachment можуть бути форматованими картками і файлами мультимедіа, які відображаються у повідомленні. Це допомагає розширити можливості бота для надсилання інформації користувачам у більш зручному форматі.

Для зручності виведення інформації про популярні фільми, використано функціонал створення та форматування адаптивних карток. Цей тип карток може містити різні типи медіа-контенту, такі як текст, аудіо, зображення, кнопки та поля для введення, що відображаються новим користувачам або користувачам, яким виведено список рекомендацій. Для роботи з адаптивними картками, необхідно встановити пакет AdaptiveCards, що є NuGet-пакетом. Адаптивна картка представляє собою об'єкт JSON, який відповідає певній схемі.

Для можливості виведення інформації щодо фільму, було розроблену схему, розміщену у файлі MCard.json, який містить назву фільму, список жанрів і зображення постера. Схема recommends_card_items.json схожа на MCard.json, але також включає інформацію про рекомендований фільм, в тому числі ймовірність того, що користувач вподобає фільм, базований на основі його попередніх переваг. Все це зображено на рисунках 4.2.7. Для можливості

оцінювати фільми за шкалою від одного до п’яти користувачами, був створений відповідний компонент рейтингової системи. Обидві схеми, MCard.json і recommends_card_items.json, в своїй структурі містять елемент SplitByColumn, який ділить область картки на стовпці, що в свою чергу дозволяє елементам розташовуватися поруч. Картка складається з п'яти стовпців, де кожен із стовпців містить постер, а також об'єкт типу Action.Submit, який відправляє запрограмовану подію боту. При натисканні на стовпець, боту приходить об'єкт JSON, який містить в тілі об’єкту, ідентифікатор фільму, відповідної картки, та значення рейтингу, що відповідає номеру колонки - від одного до п'яти.

б)

Рисунок 4.2.7 – Адаптивні картки про фільм або рекомендації

Для того, щоб відобразити список форматованих карток у вигляді каруселі (рисунок 4.2.8), необхідно встановити властивість AttachmentLayout об'єкта Activity значення Carousel (Карусель). Це дозволить відображати список форматованих карток у вигляді каруселі, котра може бути гортана користувачем.

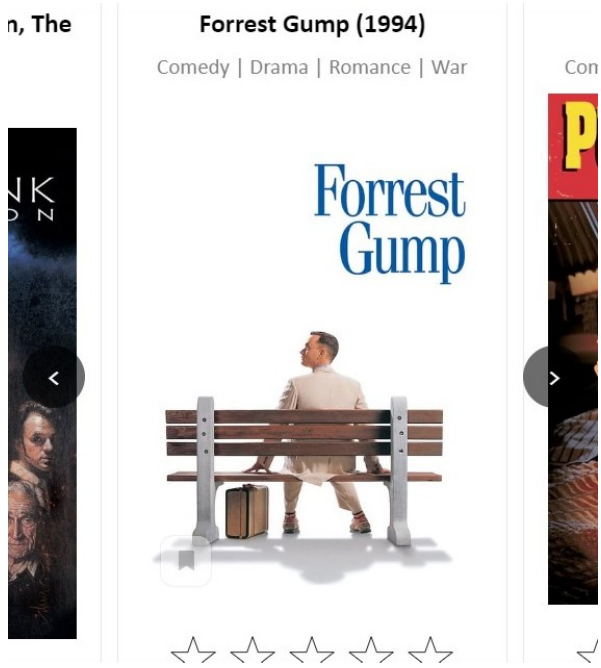


Рисунок 4.2.8 – Карусель форматованих карток

Для того, щоб скористатися сервісом LUIS, необхідно:

1. Зареєструватися на порталі luis.ai та створити нову програму LUIS.
2. Перейти в вкладку "Manage" та розділ "Versions", натиснувши кнопку "Import" та вибравши "Import as JSON" зі списку можливих опцій.
3. Вибрати файл LuisReccomend.json, який містить чотири можливих наміри користувача: GetAnyReccomends, Help, Cancel, та None.
4. Навчити програму LUIS.
5. Опублікувати програму LUIS у робочому середовищі.

Рисунок 4.2.9 містить скріншот з порталу LUIS, який показує результати для пунктів 1-3.

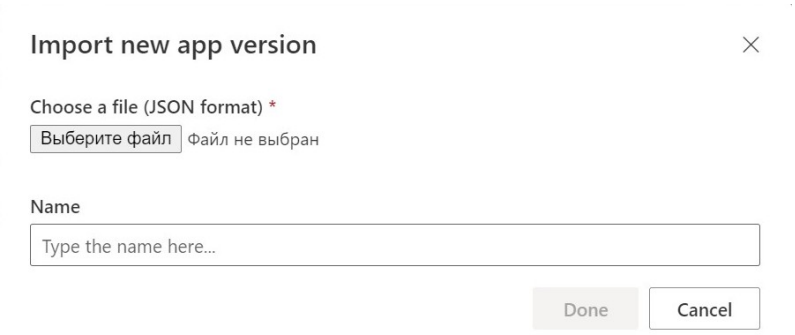


Рисунок 4.2.9 – Імпорт програми на порталі LUIS

Навчання сервісу розпізнавання мови LUIS дозволяє розпізнавати природну мову користувача. Після навчання на результати треба перевірити, за допомогою тестових висловлювань. Якщо результати не такі, як очікувалося, то треба змінити і покращити додаток LUIS та повторити процес навчання і тестування. Це ітераційний процес, який можна запустити на порталі LUIS. Слід пам’ятати, що без щонайменше хоча б одного висловлювання, початок навчання буде неможливий. Наприклад, для наміру GetAnyReccomends можна використовувати фрази, наприклад 'get any recommendations', 'what I can to see', 'please, can you make prediction for me' і так далі, що завгодно в подібному контексті.

Необхідні дії для навчання програми на порталі LUIS:

1. Обрати необхідний додаток на сторінці всіх додатків користувача.
2. У обраному додатку натиснути кнопку Train.

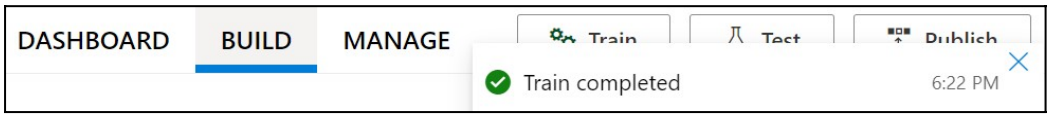


Рисунок 4.2.10 — Успішне завершення навчання

Після того, як програма була створена, навчена та протестована, її необхідно зробити доступною для клієнтської програми, опублікувавши, виконавши наступні кроки:

3. Натиснути кнопку Publish (рисунок 4.2.11).

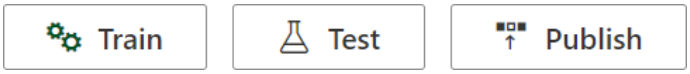


Рисунок 4.2.11 – Верхня панель на порталі LUIS

4. Вибрати параметри для публікації (рисунок 4.2.12) і натиснути кнопку "Done".

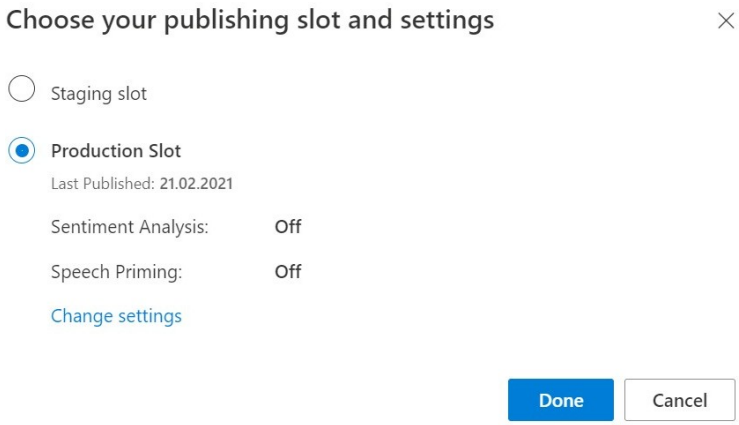


Рисунок 4.2.12 – Обрання типу публікації програми

Для того, щоб отримати доступ до програми LUIS, яка опублікована на порталі https://www.luis.ai, необхідно:

1. Відкрити опубліковану програму LUIS і перейти на вкладку Manage у верхній панелі.

2. Скопіювати значення App ID на вкладці Settings.

3. Скопіювати значення Primary Key та Location, які знаходяться на вкладнці Azure Resources.

4. Додати раніше скопійовані значення AppId і APIKey у рядки файлу налаштувань під назвою appsettings.json.

Встановлення NuGet-пакету Microsoft.Bot.Builder.AI.Luis дасть змогу використати LUIS у бот-додатку. Для взаємодії застосунку з встановленим

пакетом, створено клас RecommenderRecognizeService, який також запитує сервіс LUIS, викликаючи метод AsyncRecognizeService Після цього, бот може отримати доступ до сервісу LUIS, та отримувати інформацію, яка була додана у файл налаштувань програми appsettings.json.

LUISGen є інструментом для створення намірів та сутностей, які визначаються у моделі LUIS. Файл LuisReccomend.cs, що створений за допомогою цієї утиліти, містить в собі опис класу LuisReccomend, який використовується для збереження результатів розпізнавання повідомлення користувача під час виклику методу AsyncRecognizeService<LuisReccomend> з головного діалогу, визначеного у класі MainDialog.

Після того, як сервіс LUIS для бота був повністю налаштований та підключений, його роботу можна протестувати за допомогою додатку Bot Framework Emulator. Для цього можна ввести різні повідомлення у емуляторі і перевірити, чи вірно LUIS розпізнає наміри у них. Наприклад, якщо сервіс LUIS розпізнав намір GetAnyReccomends - отримання рекомендацій, то для даного користувача будуть генеруватися рекомендації на основі раніше навченої моделі. Сервіс LUIS був навчений відповідати на різні ймовірні запити користувача для отримання рекомендацій. Нижче наведено приклад рекомендацій, які можуть бути отримані від сервісу LUIS. Приклад наведено на рисунку 4.2.13

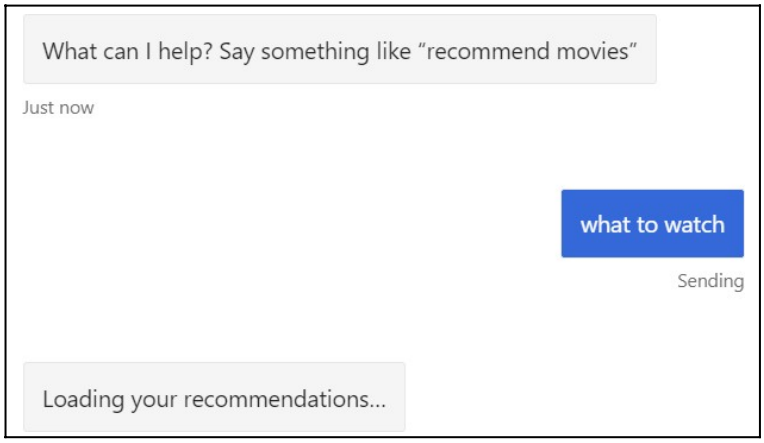


Рисунок 4.2.13 – Розпізнавання додатком LUIS наміру користувача

Після того, як бот надав рекомендації, він знову пропонує свої послуги, та

користувач може вибрати між отриманням нових рекомендацій або закінчення діалогу. Якщо LUIS розпізнає відмову, то діалог завершується (Рисунок 4.2.14).

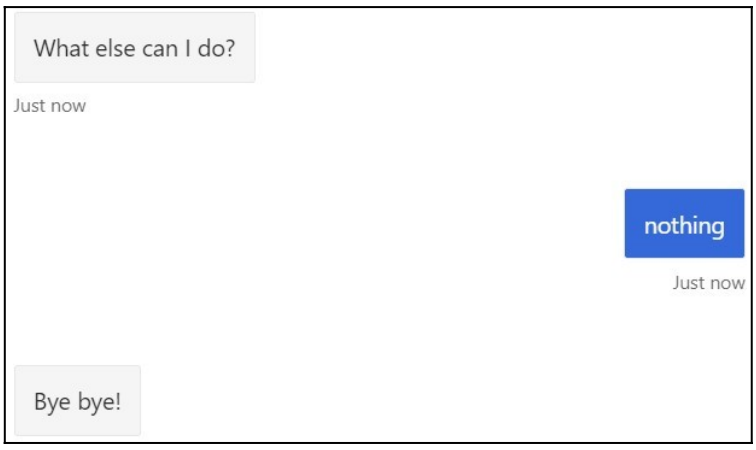


Рисунок 4.2.14 – Відмова користувача від повторного використання послуг бота

Через навчену модель LUIS є можливість отримати допомогу у бота. Для цього користувачу виводиться приклад запиту у вигляді підказки, якою можна скористатися, щоб отримати рекомендації. Наприклад, користувач може скористатися фразою "Можеш надати мені рекомендації?" або "Я бажаю отримати рекомендації", або просто "Допоможи мені". (Рисунок 4.2.15).

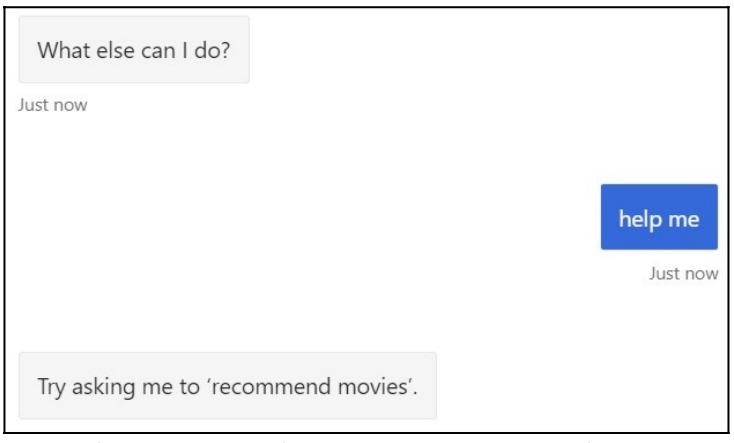


Рисунок 4.2.15 – Повідомлення бота в випадку розпізнавання наміри Help

У разі, якщо фраза користувача не відповідає жодному із раніше перерахованих намірів, LUIS її розпізнає як намір "None" і бот відповідно

виведеться повідомлення. Наприклад, якщо користувач введе фразу "Привіт, як ти?", бот може відповісти "Дякую, у мене все добре. Чи є у вас якісь спеціальні питання або бажання?" або інше відповідне повідомлення. (рисунок 4.2.16).

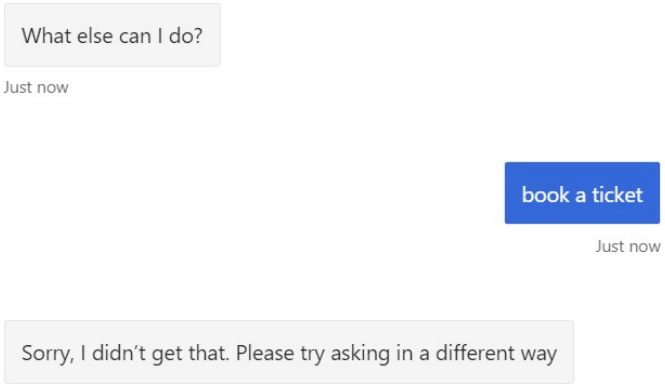


Рисунок 4.2.16 – Поведінка бот-застосунку при нерозпізнаванні наміру

Бот може виводити фільми у зручному для користувача, інтерактивному форматі, який включає в собі картку з назвою фільму, а також його постер. Для цього підключається додаток до MovieLens API. У проекті створюється інтерфейс IMPostService та реалізується його класом MoviePosterService. Після цього, цей сервіс реєструється через процес впровадження залежностей. В конструктор класу MoviePosterService впроваджуються сервіс IHttpClientFactory, який допомагає налаштувати та створювати екземпляр HttpClient для надсилання запитів до API, а також об'єкти IOpt<UrlOpt> та IOpt<LogOpt>, які, для підключення до API, містять у собі всі необхідні дані, які витягуються з конфігураційного файлу appsettings.json.

Коли бот потребує рекомендацій фільмів або популярних фільмів для нового користувача, він викликає метод AsyncGetPostLink у сервісі IMPostService, щоб отримати пряме посилання на зображення постера. Спочатку, бот використовує ідентифікатор фільму, щоб отримати інформацію про фільм у вигляді JSON об'єкта з MovieLens API, в тому числі шлях до постера. Потім бот формує посилання на постер, яке складається з адреси TMDb - популярної бази даних фільмів, розміру зображення та шляху до постера, який

був отриманий раніше. Це дозволяє отримати доступ до зображення постера фільму і відобразити його у боті.

Перед розгортанням бот-застосунку на Azure, потрібно спочатку зібрати проект у режимі випуску. Для цього можна використати Visual Studio і вибрати "Release" зі списку конфігурацій рішення. Після цього, щоб розгорнути застосунок з командного рядка Windows, потрібно встановити Azure CLI (Command Line Interface). Цей інтерфейс командного рядка допоможе розгорнути бот-застосунок на Azure із командного рядка.

На порталі Azure було створено сервіс програми, який можна розгорнути в Visual Studio як будь-який інший веб-додаток. Для цього необхідно натиснути правою кнопкою миші на проект, вибрати "Publish", потім "New" і з меню вибрати "Azure". З порталу Azure завантажуються групи ресурсів та сервіси додатків. Потім необхідно вибрати створені раніше "res_movie_bot_groups" та "movie-recommender-bot" і завершити процес. Відкриється вікно налаштувань розгортання, де необхідно змінити режим розгортання на "Self-contained", для можливості використання ботом сторонніх бібліотек та ресурсів, у тому числі раніше навчену модель. Після цього після натискання “Publish”, залишається дочекатися кінця процес. У результаті бот-додаток стає доступний на порталі Azure, де є можливість, за допомогою вбудованого каналу веб чату, провести його тестування.

Канал - це з'єднання між ботом та комунікаційними програмами, такими як Skype, Viber, Telegram та інші. Перед тим, як бот може бути доступний через ці канали, його потрібно налаштувати через портал Azure. Сервіс Bot Framework, налаштований через Azure, підключає бота до каналів і з'єднує його з користувачами. Боту не потрібно знати про конкретні канали, всю роботу виконує сервіс, в тому числі його завдання - адаптувати повідомлення від бота до формату, що підтримується каналом. Bot Framework має змогу автоматично переформатовувати будь яке повідомлення, яке надіслано в якийсь із каналів, що містить у собі картку із зображенням та кнопками.

Бот з рекомендаційною системою містить багато різноманітних карток, в

тому числі з кнопками та зображеннями, але тільки деякі канали підтримують всі ці компоненти без небажаних змін. Тому для підключення був обраний канал Web Chat. Для налаштування аідключення до каналу чат-бота, необхідно виконати наступні дії::

1. Обрати раніше створеного MovieRecommenderBot, який підключатиметься до каналу.
2. У розділі Settings натиснути на Channels.
3. Натиснути на потрібний канал.

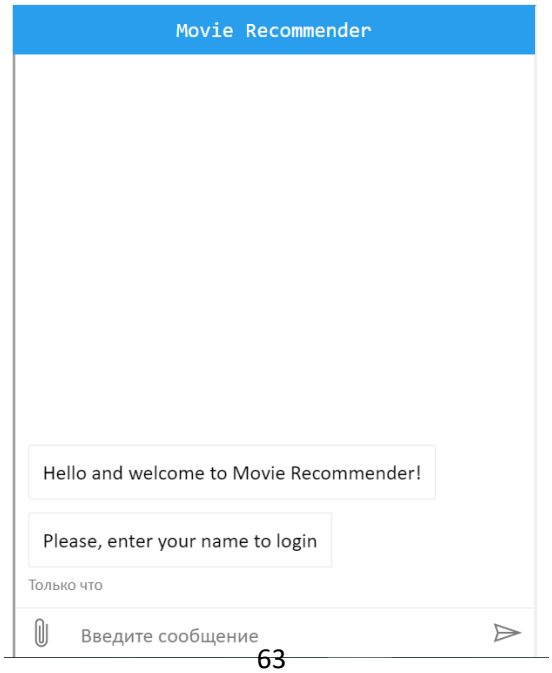
Канал Web Chat за замовчуванням, автоматично створюється при створенні бота, і тому він вже буде перерахований у списку каналів на вкладці доступних каналів, тож не потрібно виконувати зазначені вище кроки. Крім того, можна одразу ж перевірити бота, натиснувши Test in Web Chat у розділі налаштувань.

Процес налаштування для кожного каналу буде відрізнятися. Web Chat дає змогу через елементи управління, напряму запитувати бота на веб-сторінці. На порталі надані усі необхідні інструкції і дані, щоб вставити цей елемент на веб-сторінку, включаючи сгенерований секретний ключ для доступу та управлінню бот-застосунком.

Для можливості створення на будь-якому веб-сайті, віджету чату, у документі BotWebDial.html елемент div "обгортає" iframe, який фіксується до нижньої частини вікна браузера за допомогою CSS-стилів, а його висота встановлюється так, щоб початково чат не був видний. Токен бота вставляється в src атрибут, розміщений в iframe. Усередині створеного елемента div створюється ще один – що формує видимий користувачу, під час завантаження веб-сайту, блок із надписом "Movie Recommender" (рисунок 4.2.17).

Рисунок 4.2.17 — Вікно чату у згорнутому вигляді

Смужка у нижній частині екрану з написом "Movie Recommender" може буде відображена користувачу при відкритті сторінки. За допомогою JavaScript додається спеціальний обробник EventHandler, який реагує на натискання мишкою. При цьому, обробник перевіряє висоту основного div елемента: якщо вона дорівнює висотіелементу, то розгортає чат, інакше - сховує його. Таким чином, користувач може контролювати видимість чату на сайті (Рисунок 4.2.18).



63

Рисунок 4.2.18 — Вікно чату у розгорнутому вигляді

Для досягнення плавної появи віджету, використовується CSS-властивість transition для div елементу. Це дозволяє поступово змінювати висоту протягом 0,4 секунди, задаючи також параметр linear, який регулює рівномірну швидкість переходу. Таким чином, можна створити віджет, який можна розмістити на будь-якому веб-сайті, наприклад, для підбору рекомендованих фільмів за запитом користувача. Таким чином, користувачу не потрібно витрачати час на переміщення по каталогах, він може просто поставити питання на природній мові, і система знайде для нього найбільш відповідні фільми.

ВИСНОВОК

Результатом роботи являється розроблена модель машинного навчання для рекомендаційної системи фільмів, що базується на уподобаннях та перевагах користувачів, а також бот-додаток, який має змогу розпізнавати природну мову і інтегрований з рекомендаційною системою фільмів.

У ході виконання роботи були розглянуті, поставлені та вирішені наступні завдання:

1. Проведено огляд існуючих рекомендаційних систем, складені вимоги до додатку, що розробляється.
2. Розглянуто основні типи рекомендаційних систем: колаборативна фільтрація, фільтрація на основі змісту та гібридна фільтрація.
3. Для розв’язання задач з бінарної класифікації, було досліджено та обрано алгоритм машин факторизації, а саме їхню покрашену версію – машини факторизації з урахуванням специфіки поля.
4. Розглянуто можливості ML.NET для вирішення поставлених завдань машинного навчання. Також вивчено процес машинного навчання в ML.NET, а саме: завантаження та підготовка даних, розбиття їх на тренувальний і тестовий набори, створення конвейеру, і врешті-решт навчання і оцінка готової моделі.
5. З набору даних MovieLens завантажено та оброблено дані про фільми та їх рейтинги, та була здійснена підготовка даних для побудови моделі.
6. Створено та навчено модель для рекомендаційної системи фільмів, а також здійснено оцінку ефективність цієї моделі.
7. Вирішено проблему холодного старту рекомендаційної системи для користувача. Як складова холодного старту, була вирішена задача знаходження схожого за вподобаннями користувача за допомогою косинусної подібності.
8. Було вивчено та застосовано можливості бази даних Azure Table Storage для збереження рейтингів, які виставляють користувачі.
9. Спроектовано користувальницький додаток.
10. Проаналізовано функціонал та можливості Azure Cloud Services для

розробки бот-застосунку.

11. Спроектовано бот-додаток та його основні залежності.

12. Створений та підключений до бот-зстосунку LUIS додаток для розпізнавання мови користувача.

13. Розроблений бот-додаток для отримання рекомендацій фільмів з інтегровано. рекомендаційною системою, засновану на машинному навчанні.

14. Бот-додаток та віджет чату розгорнутий та підключений до Web Chat на порталі Azure.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Content-Boosted Collaborative Filtering for Improved Recommendations [Електронний ресурс] / Prem Melville, Raymond J. Mooney, Ramadass Nagarajan. – Режим доступу: https://www.aaai.org/Papers/AAAI/2002/AAAI02-029.pdf. (Дата звернення: 24.09.2022).

2. Machine Learning for .NET [Електронний ресурс]. – Режим доступу: https://reposhub.com/dotnet/machine-learning-and-data-science/dotnet-machinelearning.html. (Дата звернення: 25.09.2022).

3. Development of recommender systems using ML.NET [Електронний ресурс] / Bahrudin I Hrnjica, Denis Mušić, Selver Softic. – Режим доступу: https://www.researchgate.net/publication/340389519_DEVELOPMENT_OF_RECOMMENDER_SYSTEMS_USING_MLNET. (Дата звернення: 24.09.2022).

4. Introduction to recommender systems [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada. (Дата звернення: 28.09.2022).

5. An Intuitive Explanation of Field Aware Factorization Machines [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/an-intuitive-explanation-of-field-aware-factorization-machines-a8fee92ce29f. (Дата звернення: 28.09.2022).

6. E-Commerce Item Recommendation Based on Field-aware Factorization Machine [Електронний ресурс] / Peng Yan, Xiaocong Zhou, Yitao Duan. – Режим доступу: https://www.researchgate.net/publication/282846395_E-Commerce_Item_Recommendation_Based_on_Field-aware_Factorization_Machine. (Дата звернення: 04.10.2022).

7. The ultimate guide to binary classification metrics [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/the-ultimate-guide-to-binary-classification-metrics-c25c3627dd0a. (Дата звернення: 02.11.2022).

8. What is ML.NET and how does it work? [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/machine-learning/how-does-

mldotnet-work. (Дата звернення: 25.09.2022).

9. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems [Електронний ресурс]. – Режим доступу: http://download.tensorflow.org/paper/whitepaper2015.pdf. (Дата звернення: 18.11.2022).

10. Announcing ML.NET 0.3 [Електронний ресурс]. – Режим доступу: https://devblogs.microsoft.com/dotnet/announcing-ml-net-0-3/#onnx-section. (Дата звернення: 18.11.2022).

11. ONNX Runtime for inferencing machine learning models [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/en-us/blog/onnx-runtime-for-inferencing-machine-learning-models-now-in-preview/. (Дата звернення: 22.11.2022).

12. Unified Modeling Language (UML) | Activity Diagrams [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/. (Дата звернення: 16.11.2022).

13. MovieLens Latest Full [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/grouplens/movielens-latest-full. (Дата звернення: 14.10.2022).

14. Prepare data for building a model [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/prepare-data-ml-net. (Дата звернення: 16.10.2022).

15. Personality-Based Active Learning for Collaborative Filtering Recommender Systems [Електронний ресурс] / Mehdi Elahi, Matthias Braunhofer, Francesco Ricci, Marko Tkalcic. – Режим доступу: http://www.cp.jku.at/research/papers/elahi2013aixia.pdf. (Дата звернення: 06.09.2022).

16. The Cold Start Problem for Recommender Systems [Електронний ресурс]. Режим доступу: https://yuspify.com/blog/cold-start-problem-recommender-systems/. (Дата звернення: 06.08.2022).

17. MachineX: Cosine Similarity for Item-Based Collaborative Filtering

[Електронний ресурс]. – Режим доступу: https://dzone.com/articles/machinex-cosine-similarity-for-item-based-collabor. (Дата звернення: 06.09.2022).

18. What is Azure Table storage? [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/storage/tables/table-storage-overview. (Дата звернення: 07.08.2022).

19. 27 Incredible Chatbot Statistics [Електронний ресурс]. – Режим доступу: https://www.netomi.com/chatbot-statistics . (Дата звернення: 19.08.2022).

20. Get started guide for developers on Azure [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer–guide#what-is-azure. (Дата звернення: 19.08.2022).

21. About Azure Bot Service [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-4.0. (Дата звернення: 19.08.2022).

22. Azure Monitor overview [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/azure-monitor/overview. (Дата звернення: 23.09.2022).

23. What are Azure Cognitive Services? [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/cognitive-services/welcome. (Дата звернення: 25.09.2022).

24. What is Language Understanding (LUIS)? [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis. (Дата звернення: 25.10.2022).

25. Connect a bot to channels [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-manage-channels?view=azure-bot-service-4.0. (Дата звернення: 06.09.2022).

26. Use Search In Application Insights [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/azure-monitor/app/diagnostic-search. . (Дата звернення: 07.04.2022).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА
«РОЗРОБКА МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З КОРИСТУВАЧАМИ НА БАЗІ МЕТОДІВ МАШИННОГО НАВЧАННЯ»

Виконав: студент групи ПДМ – 61, Герасимчук Максим Володимирович

Керівник: д.т.н., проф., директор Навчально-наукового інституту інформаційних технологій Бондарчук А.П.

Київ - 2022

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: покращення взаємодії з користувачами на основі розробленої моделі з використанням методів машинного навчання.

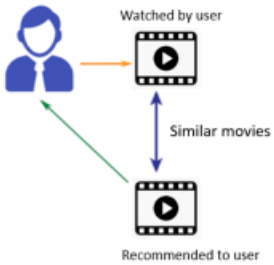
Об'єкт дослідження: процес взаємодії з користувачами.

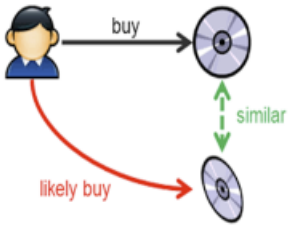
Предмет дослідження: рекомендаційні системи на базі методів машинного навчання.

ІСНУЮЧІ МЕТОДИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Content-Based метод
Переваги:
- Не потребує даних про інших користувачів.
- Працює окремо для кожного користувача

Недоліки:
- Необхідне професійне знання предметної області
- Обмежена в можливості розширення інтересів користувача

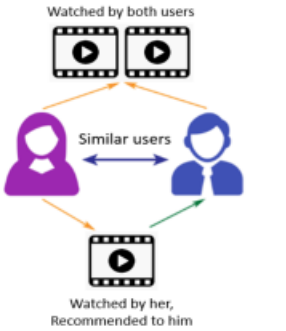




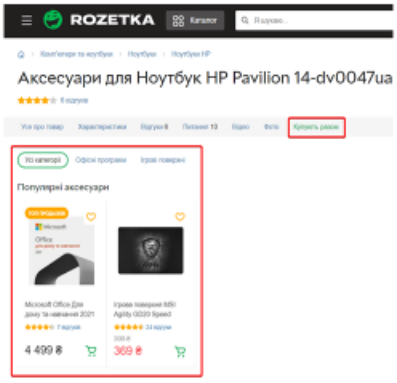
ІСНУЮЧІ МЕТОДИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Колаборативна фільтрація
Переваги:
- Не потрібні знання предметної галузі
- Розширення кола інтересів користувача
- Для навчання потрібно мімімум вхідних даних

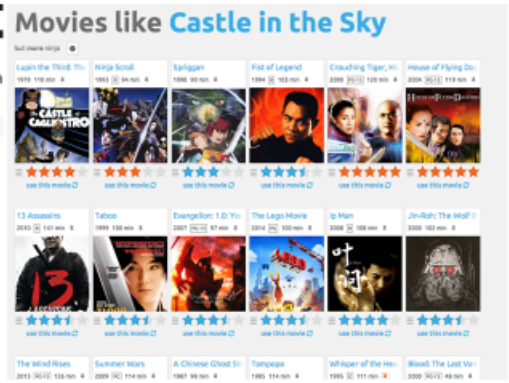
Недоліки:
- Проблема «холодного» старту
- Можлива необхідність додавати побічні функції



ПРИКЛАДИ ІСНУЮЧИХ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

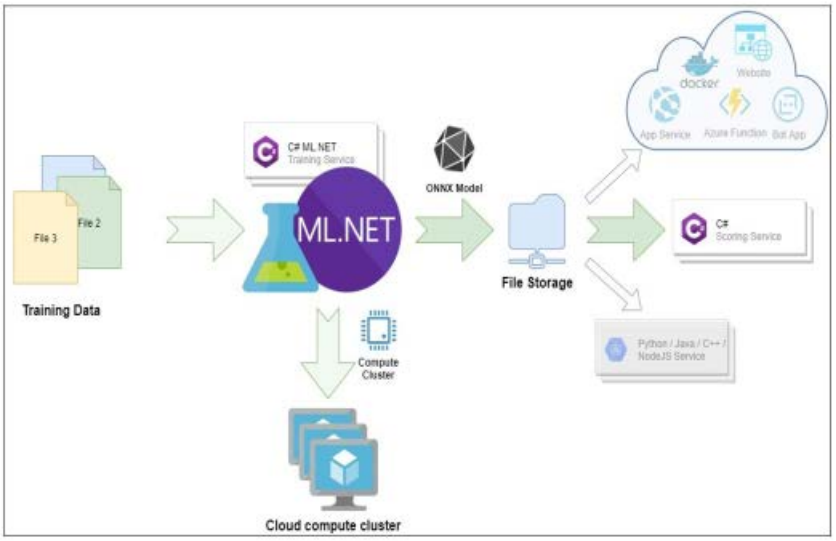


Використання рекомендаційних систем в інтернет магазині Rozetka

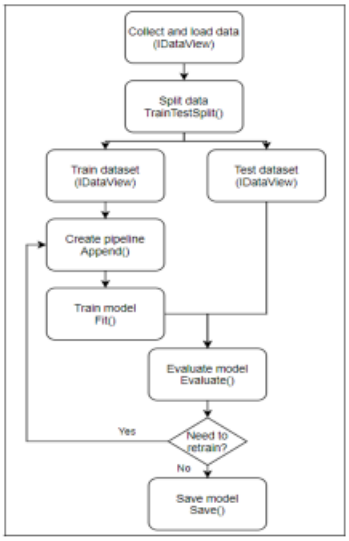


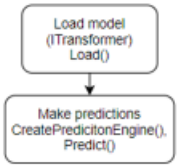
Сервіс рекомендації фільмів MovieLens

СХЕМА СТВОРЕННЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ



ДІАГРАМА ДІЯЛЬНОСТІ ПРОЦЕСУ НАВЧАННЯ МОДЕЛІ

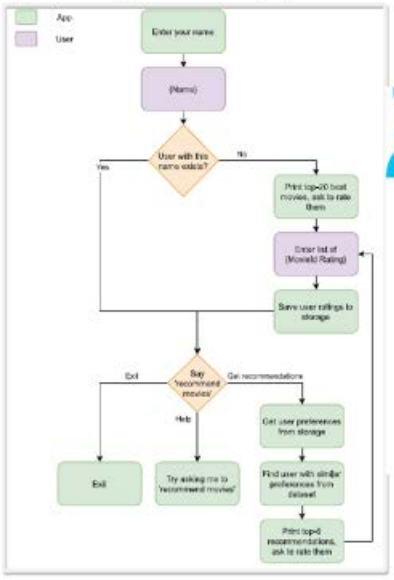


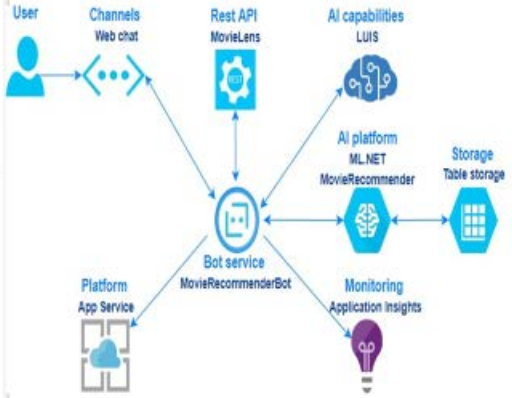


ОЦІНКА РЕЗУЛЬТАТІВ НАВЧЕНОЇ МОДЕЛІ

```
Evaluation Metrics
Accuracy: 0,78
Area Under Roc Curve: 0,86
Area Under Precision Recall Curve: 0,86
F1 Score: 0,78
Negative Precision: 0,78
Negative Recall: 0,77
Positive Precision: 0,77
Positive Recall: 0,78
Confusion matrix: TEST POSITIVE RATIO:  0,5006 (7530500,0/(7530500,0+7511742,0))
Confusion table
          ||======================
PREDICTED || positive | negative | Recall
TRUTH     ||======================
 positive || 5 910 246 | 1 620 254 | 0,7848
 negative || 1 741 051 | 5 770 691 | 0,7682
          ||======================
Precision ||   0,7725 |   0,7808 |
```

ДІАГРАМА ДІЯЛЬНОСТІ ТА СХЕМА БОТ-ДОДАТКУ





ВИСНОВКИ

1. Проведено аналіз існуючих рекомендаційних систем, складені вимоги до додатку, що розробляється.
2. Досліджено основні типи рекомендаційних систем а також алгоритми колаборативної фільтрації.
3. Створено і навчено модель для рекомендаційної системи фільмів. Здійснена оцінка ефективності навченої моделі.
4. Розглянуто основні типи бот-додатків. Вивчено можливості Azure Bot Service і Bot Framework – засобів для створення, тестування, розгортання і управління інтелектуальними ботами в єдиному середовищі.
5. Розроблено бот-додаток для отримання рекомендацій фільмів та інтегровано з раніше розробленою рекомендаційною системою на основі машинного навчання.
6. Бот-додаток розгорнутий на порталі Azure та підключений до каналу Web Chat, створений віджет чату для використання його на веб-сайтах.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Колумбет В.П., Сторчак К.П., Герасимчук М.В. Метод трансформації моделі організаційно-технічної системи в модель інформаційної системи / Зв'язок. №2 (155), 2022. С. 33.

Тези доповідей на конференціях:

1. Герамимчук М.В. Рекомендаційні системи як складова сучасних інфокомунікаційних технологій. // XV Науково-технічна конференція «Сучасні інфокомунікаційні технології» – Київ: ДУТ, 2022.
2. Герамимчук М.В. Аналіз основних типів рекомендаційних систем. // Науково-практична конференція «Проблеми комп'ютерної інженерії» – Київ: ДУТ, 2022.

ДЯКУЮ ЗА УВАГУ!