Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

Методи побудови рекомендаційних систем, заснованих на знаннях
(тема)

Виконав:
студент 2 курсу, групи СШІм–19–2
Беседін Ф.О.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Рябова Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

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

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бесединy Федору Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи побудови рекомендаційних систем, заснованих на знаннях

затверджена наказом університету від 29 03 20 21 р. № 390Ст.

2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 ___ р.

3. Вихідні дані до роботи Огляд та аналіз методів побудови рекомендаційних систем, заснованих на знаннях. Розробка прототипу рекомендаційної системи, заснованої на знаннях у предметній області кулінарних рецептів. Перелік використовуваних програмних засобів: ОС Windows 10, середовище розробки Jupyter Notebook, середовище розробки Google Colab. Технічне забезпечення: ноутбук зі встановленим програмним середовищем Jupyter Notebook

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної області; Постанова задач досліджень; Характеристика предметної області; Проблематика предметної області; Аналіз методів побудови рекомендацій, заснованих на знаннях; Розробка прототипу рекомендаційної системи Recipe Recommender, що рекомендує рецепти страв залежно від запитів користувача; Огляд використаного середовища та набору даних; Передобробка даних; Реалізація методів рекомендації; Демонстрація роботи системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1.1 – Рекомендаційна система MovieLens; Рисунок 1.2 – Рекомендаційна система Amazon.com; Рисунок 1.3 – Рекомендаційна система Netflix; Рисунок 1.4 – Приклад рекомендації із epicurious.com; Рисунок 1.5 – Приклад автоматично згенерованих колекцій програми Caviar; Рисунок 1.6 – Дослідження у предметній області рекомендаційних систем для кулінарних рецептів; Рисунок 2.1 – Прогнозування рейтингового значення для цільового рецепта; Рисунок 2.2 – Приклад графу рецептів для планування меню; Рисунок 3.1 – Цикл вирішення проблеми за допомогою CBR; Рисунок 4.1 – Опис колонок; Рисунок 4.2 – Представлення датасету; Рисунок 4.3 – Видалення зайвих колонок та перетворення колонки nutrition; Рисунок 4.4 – Створені колонки; Рисунок 4.5 – Фільтрація тегів рецепту; Рисунок 4.6 – Додані колонки; Рисунок 4.7 – Створення колонок двох типів; Рисунок 4.8 – Частина алгоритму рекомендації/бази знань; Рисунок 4.9 – Функція подібності; Рисунок 4.10 – Вхідні дані; Рисунок 4.11 – Зміна кількості рекомендацій при різних запитах

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 29.03.2021 | виконано |
| 2 | Аналіз предметної області | 30.03.2021 – 31.03.2021 | виконано |
| 3 | Постанова задач досліджень | 01.04.2021 – 02.04.2021 | виконано |
| 4 | Характеристика предметної області | 02.04.2021 – 03.04.2021 | виконано |
| 5 | Проблематика предметної області | 04.04.2021 – 05.04.2021 | виконано |
| 6 | Аналіз методів побудови рекомендацій на знаннях | 06.04.2021 – 07.04.2021 | виконано |
| 7 | Огляд використаного середовища та набору даних | 07.04.2021 – 10.04.2021 | виконано |
| 8 | Реалізація методів рекомендації | 11.04.2021 – 17.04.2021 | виконано |
| 9 | Перевірка роботи системи | 17.04.2021 – 18.04.2021 | виконано |
| 10 | Написання пояснювальної записки | 18.04.2021 – 19.04.2021 | виконано |
| 11 | Попередній захист | 14.05.2021 | виконано |
| 12 | Захист перед ЕК | 19.05.2021 | |

Дата видачі завдання 29 березня 20 21 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 97 с., 22 рис., 2 дод., 44 джерела.

МЕТОДИ ІНЖЕНЕРІЇ ЗНАНЬ, МЕТОДИ ПОДАННЯ ЗНАНЬ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, CASE-BASED REASONING, CONSTRAINT-BASED RECOMMENDATIONS.

Об’єкт дослідження – рекомендаційні системи, засновані на знаннях.

Мета роботи – аналіз та застосування методів побудови рекомендаційних систем, заснованих на знаннях в предметній області кулінарних рецептів.

Методи дослідження – методи системного аналізу, методи машинного навчання, методи інженерії знань, аналіз рекомендаційних систем, побудова прототипу рекомендаційної системи, заснованої на знаннях за допомогою інтегрованого середовища розробки Jupyter Notebook, мови програмування Python.

Предмет дослідження – методи побудови рекомендаційних систем, заснованих на знаннях.

В результаті проведених досліджень вирішено задачу порівняння та аналізу різноманітних методів побудови рекомендаційних систем, заснованих на знаннях, створено прототип. Дана розробка може бути корисно застосована великою кількістю користувачів у сфері кулінарії.

# РЕФЕРАТ

Пояснительная записка: 97 с., 22 рис., 2 прил., 44 источника.

МЕТОДЫ ИНЖЕНЕРИИ ЗНАНИЙ, РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ, СПОСОБЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ, CASE-BASED REASONING, CONSTRAINT-BASED RECOMMENDATIONS.

Объект исследования – рекомендательные системы, основанные на знаниях.

Цель работы – анализ и применение методов построения рекомендательных систем, основанных на знаниях в предметной области кулинарных рецептов.

Методы исследования – методы системного анализа, методы машинного обучения, методы инженерии знаний, анализ рекомендательных систем, построение прототипа рекомендательной системы, основанной на знаниях с помощью интегрированной среды разработки Jupyter Notebook, языки программирования Python.

Предмет исследования – методы построения рекомендательных систем, основанных на знаниях.

В результате проведенных исследований решена задача сравнения и анализа различных методов построения рекомендательных систем, основанных на знаниях, создан прототип. Данная разработка может быть полезно использована большим количеством пользователей в сфере кулинарии.

# ABSTRACT

Explanatory note: 97 p., 22 fig., 2 ann., 44 sources.

CASE-BASED REASONING, CONSTRAINT-BASED RECOMMENDATIONS, KNOWLEDGE ENGINEERING, KNOWLEDGE PRESENTATION, RECOMMENDER SYSTEMS.

The object of research is knowledge-based recommender systems.

The purpose of the work is the analysis and application of methods for building recommender systems based on knowledge in the subject area of culinary recipes.

Research methods – methods of system analysis, methods of machine learning, methods of knowledge engineering, analysis of recommender systems, construction of a prototype of a knowledge-based recommender system using an integrated development environment Jupyter Notebook, Python programming language.

The subject of research – methods of building recommender systems based on knowledge.

As a result of the conducted researches the problem of comparison and analysis of various methods of construction of the recommender systems based on knowledge is solved, the prototype is created. This development can be usefully applied by a large number of users in the field of cooking.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CB – на основі вмісту – content-based;

CBR – міркування на основі прецедентів – case based reasoning;

CF – колаборативна фільтрація – collaborative filtration;

CKB – база знань обмежень – constraint knowledge base;

KBRS – рекомендаційна система, заснована на знаннях – knowledge-based recommender system;

RS – рекомендаційна система – recommender system;

TF-IDF – Term Frequency-Inverse Document Frequency;

USDA – Міністерство сільського господарства США – United States Department of Agriculture.

# ВСТУП

На поточний момент ми живемо в епоху інформаційних технологій. Це час, коли в Інтернеті, де кількість варіантів незліченна, існує потреба у фільтруванні, визначенні пріоритетів та ефективному наданні відповідної інформації, щоб полегшити проблему перевантаження інформації, що створило потенційну проблему для багатьох користувачів Інтернету. Рекомендаційні системи вирішують цю проблему шляхом пошуку у великому обсязі динамічно генерованої інформації задля надання користувачам персоналізованого вмісту та послуг.

Рекомендаційні системи – одна з найуспішніших і широко поширених застосувань технологій машинного навчання. Рекомендаційні системи допомагають збільшити дохід бізнесу і допомагають покупцям купувати найбільш підходящий для них продукт. Вони широко використовуються в онлайн магазинах, музичних і кіно сервісах, картах і т.д.

Зараз існує три основних методи побудови рекомендаційних систем, найбільш новим і маловивченим із яких є метод, заснований на знаннях. Цей метод базується на чітких знаннях про асортимент товарів, уподобаннях користувачів та критеріях рекомендацій (тобто, який товар слід рекомендувати в якому контексті). Ці системи застосовуються у сценаріях, коли неможливо застосувати альтернативні підходи, такі як колаборативна фільтрація та фільтрація на основі вмісту, або для створення гібридної системи на їх основі.

Зважаючи на вищесказане, темою даної кваліфікаційної роботи обрано дослідження методів побудови рекомендаційних систем, заснованих на знаннях на прикладі предметної області кулінарних рецептів.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд і аналіз сучасного стану розгляду проблеми

Рекомендаційні системи мають на меті передбачити інтереси користувачів та рекомендувати товари, які, цілком ймовірно, є для них цікавими. Вони є одними з найпотужніших систем машинного навчання, які різноманітні компанії впроваджують для стимулювання продажів.

Дані, необхідні для рекомендаційних систем, походять від оцінок користувачів після перегляду фільму чи прослуховування пісні, неявних запитів пошукової системи та історії покупок або з іншої інформації про самих користувачів чи предмети. Такі сайти, як Spotify, YouTube або Netflix, використовують ці дані для того, щоб пропонувати списки рекомендацій – так звані щоденні мікси, або рекомендації щодо відео відповідно [1].

Важливим каталізатором розвитку таких систем із розповсюдженням інтернету у цьому плані є легкість, з якою інтернет дозволяє користувачам залишати відгуки про свої симпатії чи антипатії. Наприклад, розглянемо варіант такого постачальника контенту, такого як Netflix. У цьому випадку користувачі можуть легко надати зворотній зв'язок простим клацанням миші. Типовою методологією надання зворотного зв'язку є форма рейтингу, в якій користувачі вибирають числові значення з певної системи оцінки (наприклад, п'ятизіркової системи рейтингу), що визначає їх симпатії та антипатії до різних елементів.

Інші форми зворотного зв’язку не є настільки очевидними, але їх навіть простіше отримати у мережі. Наприклад, простий вчинок користувача, який купує або переглядає товар, може розглядатися як схвалення цього товару. Такі форми зворотного зв’язку зазвичай використовуються інтернет-продавцями, такими як Amazon.com, і збір даних цього типу здійснюється абсолютно без зусиль, що вимагаються від клієнта. Основна ідея

рекомендаційних систем, полягає у використанні різних джерел даних для виведення інтересів клієнтів. Суб’єкт, якому надається рекомендація, називається користувачем, а предмет, що рекомендується, називається товаром. Тому аналіз рекомендацій часто базується на попередній взаємодії між користувачами та предметами, оскільки минулі інтереси та схильності часто є хорошими показниками майбутнього вибору. Помітним винятком є випадок систем рекомендацій, заснованих на знаннях, в яких рекомендації пропонуються на основі визначених користувачем вимог, а не з минулої історії користувача [2].

Основний принцип рекомендацій полягає в тому, що існує суттєва залежність між активністю, орієнтованою на товари та орієнтованою на користувача. Наприклад, користувач, якого цікавить історичний документальний фільм, швидше за все зацікавить інший історичний документальний фільм чи освітню програму, а не бойовик. У багатьох випадках різні категорії предметів можуть виявляти суттєві взаємозв'язки, які можна використовувати для отримання більш точних рекомендацій. Альтернативно, залежності можуть бути присутніми при більшій деталізації окремих елементів, а не категорій. Ці залежності можна дізнатись на основі даних із матриці рейтингів, а отримана модель потім використовується для прогнозування цільових користувачів. Чим більша кількість оцінюваних елементів, доступних для користувача, тим легше робити надійні прогнози щодо майбутньої поведінки користувача. Для виконання цього завдання можна використовувати багато різних моделей навчання. Наприклад, колективна поведінка покупців або рейтинг різних користувачів може бути використана для створення груп подібних користувачів, які зацікавлені в подібних продуктах. Інтереси та дії цих груп можуть бути використані для надання рекомендацій окремим членам цих груп.

З точки зору користувача, рекомендації можуть допомогти покращити загальну задоволеність користувачів веб-сайтом. Наприклад, користувач, який

неодноразово отримує відповідні рекомендації від Amazon.com, буде більш задоволений досвідом і, швидше за все, знову використовуватиме цей сайт. Це може покращити лояльність користувачів та ще більше збільшити продажі на сайті. На кінці продавця процес рекомендацій може дати уявлення про потреби користувача та допомогти в подальшій налаштуванні взаємодії з користувачем. Також надання корисних пояснень користувачеві щодо того, чому рекомендується певний предмет часто є корисним. Наприклад, у випадку Netflix надаються рекомендації разом із раніше переглянутими фільмами. Далі буде розглянуто найбільш відомі рекомендаційні системи.

GroupLens була новаторською рекомендаційною системою, яка була побудована як дослідний прототип для рекомендації новин Usenet. Система збирала рейтинги від читачів Usenet і використовувала їх, щоб передбачити, чи сподобається стаття іншим читачам, перш ніж вони її прочитають. Деякі з найдавніших автоматизованих алгоритмів колаборативної фільтрації були розроблені в групою вчених, що створили GroupLens. Загальні ідеї, розроблені цією групою, були поширені і на інші налаштування продуктів, такі як книги та фільми. Відповідні системи рекомендацій називались BookLens та MovieLens (рисунок 1.1), відповідно. Окрім свого новаторського внеску у спільні дослідження фільтрації, дослідницька група GroupLens відзначилася випуском декількох наборів даних у перші роки роботи в цій галузі, коли набори даних були непросто доступними для порівняльного аналізу. Яскраві приклади включають три набори даних [3] із системи рекомендацій MovieLens. Ці набори даних послідовно збільшуються і містять $10^5$, $10^6$ та $10^7$ оцінок відповідно.

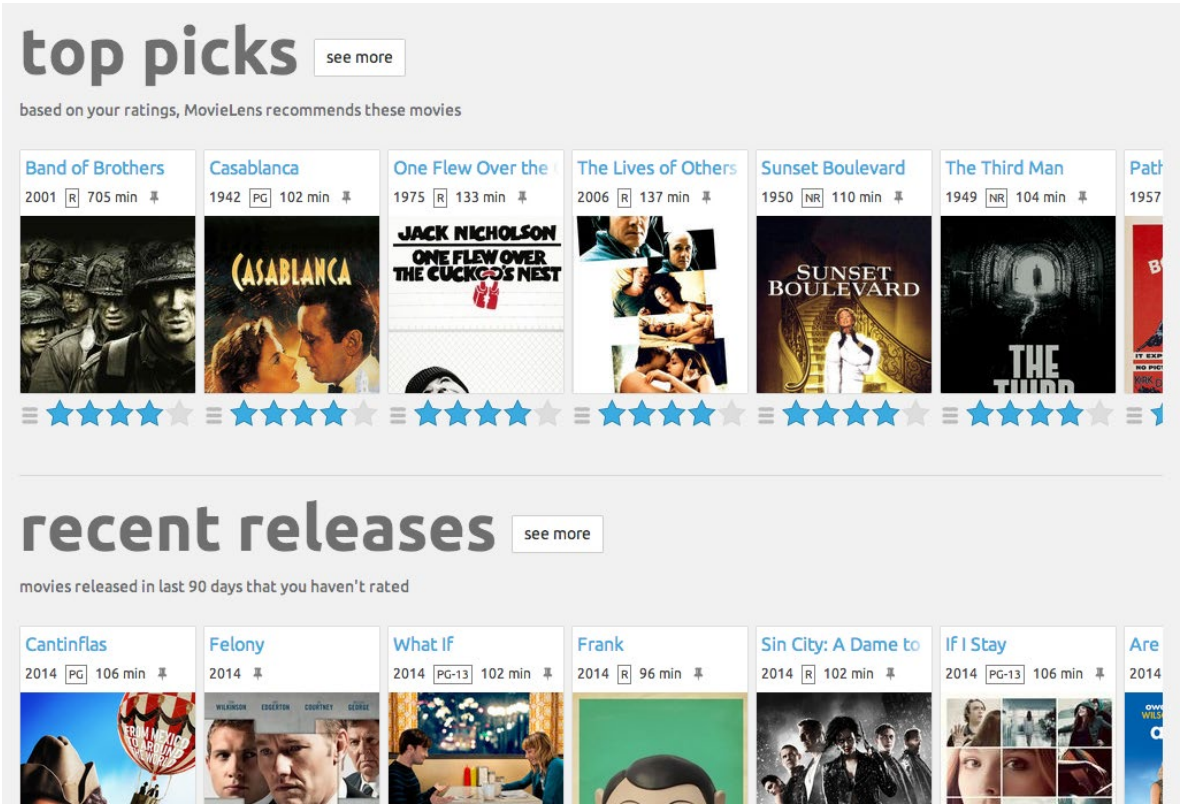


Рисунок 1.1 – Рекомендаційна система MovieLens

Amazon.com [4] також був одним із першопрохідців рекомендаційних систем, особливо в комерційному середовищі. У перші роки це був один з небагатьох роздрібних торговців, що усвідомив корисність цієї технології. Спочатку заснований як книжковий інтернет-магазин, цей бізнес розширився практично до усіх видів продукції. Зараз Amazon.com зараз продає практично всі категорії продуктів, такі як книги, компакт-диски, програмне забезпечення, електроніка тощо. Рекомендації на Amazon.com надаються на основі чітко вказаних рейтингів, поведінки покупців та поведінки веб-переглядачів. Рейтинг на Amazon.com визначається за 5-бальною шкалою, найнижчий рейтинг – 1 зірка, а найвищий – 5 зірок, що можна побачити на рисунку 1.2. Дані про покупки та перегляд, що відповідають замовнику, можна легко зібрати, коли користувачі входять в систему за допомогою механізму аутентифікації облікового запису, що підтримується Amazon. Рекомендації

також надаються користувачам на головній веб-сторінці сайту кожного разу, коли вони входять у свої облікові записи. У багатьох випадках надаються пояснення щодо рекомендацій. Наприклад, відношення рекомендованого товару до раніше придбаних товарів може бути включене до системного інтерфейсу рекомендатора. Поведінку покупця або перегляду користувача можна розглядати як тип неявного рейтингу, на відміну від явного рейтингу, який визначається користувачем. Багато комерційних систем, у тому числі Amazon, дозволяють гнучко надавати рекомендації як на основі явного, так і неявного зворотного зв'язку.


Рисунок 1.2 – Рекомендаційна система Amazon.com

Netflix була заснована як компанія з прокату цифрових відеодисків (DVD) із замовленням поштою [5] фільмів та телевізійних шоу, яка з часом була розширена до потокової доставки. В даний час основним бізнесом Netflix є забезпечення потокової доставки фільмів та телевізійних

шоу за передплатою. Netflix надає користувачам можливість оцінювати фільми та телевізійні шоу за 5-бальною шкалою. Крім того, дії користувача щодо перегляду різних предметів також зберігаються Netflix. Потім ці рейтинги та дії використовуються Netflix для надання рекомендацій. Netflix чудово справляється з наданням пояснень щодо рекомендованих елементів. У ньому явно наводяться приклади рекомендацій, заснованих на конкретних елементах, які спостерігав користувач. Така інформація надає користувачеві додаткову інформацію, щоб вирішити, дивитись певний фільм чи ні, це показано на рисунку 1.3. Надання значущих пояснень важливо для того, щоб надати користувачеві розуміння, чому їм може бути цікавим певний фільм. Цей підхід також збільшує для користувача можливість діяти відповідно до рекомендацій та покращує взаємодію з користувачем. Цей цікавий підхід може також допомогти покращити лояльність та утримання клієнтів.

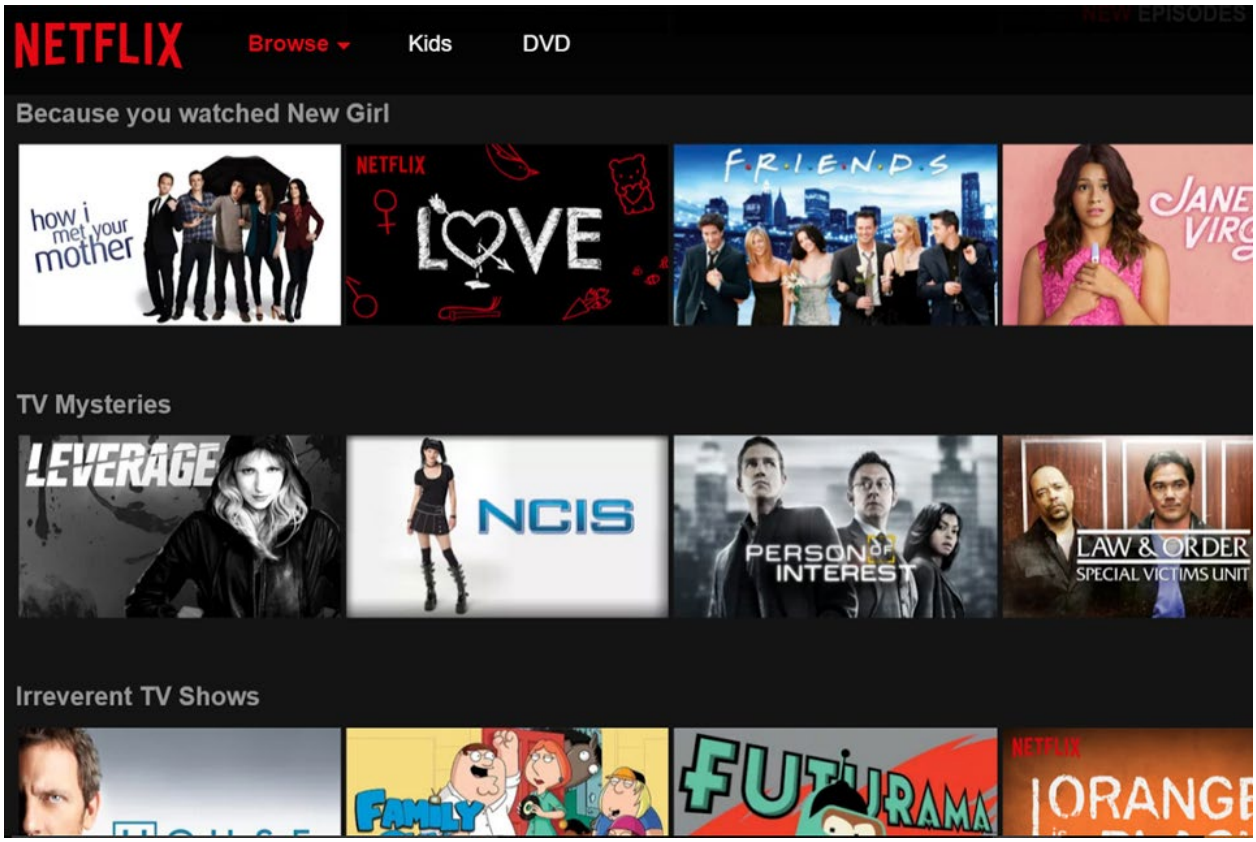


Рисунок 1.3 – Рекомендаційна система Netflix

У контексті предметної області кулінарії є багато представників рекомендаційних систем, побудованих за допомогою кількох методів, як власних так і гібридних. Ці методи та приклади систем будуть розглянуті у наступному підрозділі.

## 1.2 Класифікація та методи побудови рекомендаційних систем

Механізми рекомендацій в RS системах фільтрують продукти, які можуть бути цікаві конкретному клієнтові або які він купить, виходячи з його попередньої історії. Чим більше даних про клієнта є, тим точніше будуть рекомендації. Залежно від методів фільтрації виділяють три основних типи RS систем: засновані на колаборативній фільтрації (collaborative filtering); на змісті (content-based); на знаннях (knowledge-based).

Колаборативна фільтрація передбачає наявність матриці оцінок користувач-елемент. Ідея полягає в тому, щоб для кожного користувача знайти найбільш схожих «сусідів» і заповнити пропуски конкретного користувача, виважено усереднюючи рейтинги «сусідів». Колаборативна фільтрація є найпоширенішим підходом до формування рекомендацій, часто слугуючи так званою базовою лінією для порівняння з іншими іншими методами [6] Це не дивно, враховуючи, що це стандартна техніка, яка може працювати нестандартно, не вимагаючи високоякісних тегів або інших метаданих для елементів. Однак її найбільшим недоліком є проблема холодного старту для нових користувачів та предметів [7], у протидію якої деякі системи проводять опитування клієнтів на його смак та інтереси. Наприклад, додаток рецептів allrecipes.com просить користувачів оцінити щонайменше 3 із 51 категорій, в які входить кухня, основний інгредієнт, номер страви та корисність для здоров'я перш ніж вони зможуть користуватися додатком. Колаборативна фільтрація є хорошим підходом для сайтів з великою кількістю можливостей відстежувати відклик користувачів, таких як сайти рецептів, на яких

користувачі стежать і оцінюють внески до системи інших користувачів (наприклад, allrecipes.com, epicurious.com та cookpad.com), а також служби доставки їжі, де користувачі використовують рейтинги для вибору їжі та місця, де сайти підштовхують або стимулюють користувачів надати відгуки.

Ідея content-based підходу полягає в тому, щоб створити для користувача вектор його переваг в просторі предметів з історії дій користувача, і рекомендувати товари, близькі до цього вектору [1]. Рекомендаційні системи на основі вмісту також часто зустрічаються. Вони пропонують можливість зрозуміти смаковий профіль людини – їм подобаються такі категорії та кухні, як цитрусові, китайська їжа, середні спеції, але не телятина, – і подавати подібні страви. Така стратегія потенційно дозволить системі запам'ятати дієтичні обмеження від відстежених та оцінених предметів за умови достатньої кількості метаданих елементів. Є ризик негативного впливу на користувача у разі рекомендації їжу, яку він не може їсти. Рекомендаційні системи на б основі вмісту рідше мають можливість помилкової рекомендації, наприклад продукти харчування чи категорії, до яких ви не виявляли зацікавленості, але можуть не мати різноманітності та випадковості. Така особливість може бути особливо важливою рекомендаційних систем, оскільки люди не люблять їсти одне і те ж щодня, кухарі часто хочуть розширити сферу застосування та спробувати нові речі і що існує важлива сезонність інгредієнтів. Тільки тому, що в користувача не було взаємодії із продуктами з полуницею не виявляє недостатнього інтересу, вони, можливо, досі не були доступні на ринку. Але існують деякі підходи [8], [9] до впровадження різноманітності у рекомендаційних системах без втрати точності. У більшості випадків атрибути змісту застосовуються до продукту в цілому: кухня, харчування або пікантність страви. Однак рецепти – цікавий виняток. Вони являють собою суміш атрибутів рівня рецепта, а також атрибутів рівня інгредієнтів. Тобто

окремі інгредієнти можуть асоціюватися з пікантністю, поживністю або визначати домінуючий смак. Тобто рецепт – це вектор інгредієнтів, кожен з яких може мати відповідні ваги, мітки та атрибути. Рейтинг рецептів також можна передавати самим інгредієнтам [10]. На рисунку 1.4 представлено приклад рекомендацій щодо їжі на основі вмісту від epicurious.com. У ситуації холодного старту уподобання «холодний заварний сливовий холодний чай» призводить до набору чотирьох рекомендованих продуктів на основі сливи та холодного чаю.

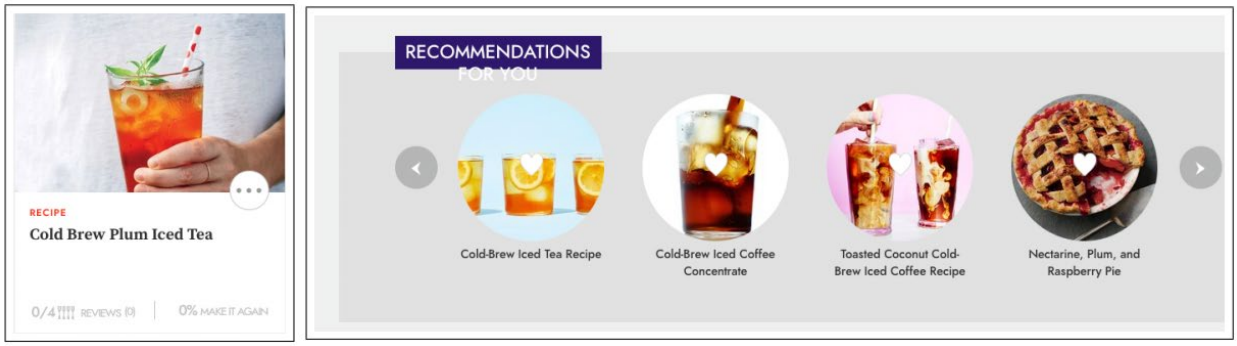


Рисунок 1.4 – Приклад рекомендації із epicurious.com

Враховуючи плюси і мінуси колаборативної фільтрації та рекомендацій, заснованих на вмісті, кілька досліджень спробували гібридний підхід [8], припустимо, що користувачі надають перевагу рецептам, але рецепт містить чіткі інгредієнти, кожен з яких має інгредієнт з прихованим фактором (у 50 вимірах). Факторизуючи матрицю інгредієнтів для користувачів (а не рецепти користувачів), вони створили набір значущих факторів, що відповідають групам, наприклад хліб, яловичина, напої та барбекю. Потім на профіль користувача орієнтується система щодо всіх інгредієнтів у всіх рецептах, які він оцінив, а рецепт-кандидат є вектором між факторами інгредієнтів, які можна класифікувати та рекомендувати за подібністю косинусів. Також було створено рекомендацію на основі вмісту, передавши рейтинги рецептів самим

інгредієнтам, зваженим однаково. Однак інгредієнт може бути частиною набору рецептів, які отримали неоднозначні оцінки, але цей інгредієнт не був причиною поганих оцінок. Щоб протистояти цьому, розглядаються лише позитивні оцінки інгредієнтів, які отримали неоднозначні оцінки. Потім спільна фільтрація використовується для прогнозування оцінки неоцінених інгредієнтів. Таким чином, цей гібрид перевершив як їх систему тільки на колаборативній фільтрації, так і на основі вмісту.

Вкладання (embeddings) слів розширюють векторну модель для обробки тексту. Незважаючи на те, що вони зазвичай не використовуються для харчових продуктів, у літературі є кілька цікавих прикладів. Назва їжі, опис страви або детальний рецепт – це всі текстові документи. Використовуючи ці вхідні дані, моделі вбудовування, такі як word2vec [10], можуть вивчати асоціації серед слів і створювати не тільки кластери та асоціації за допомогою яких створюються рекомендації, але й аналогії. Наприклад, є варіант системи, у якій автори [11] використовували word2vec у меню ресторану, перетворювали пункти меню у вектори за допомогою тієї ж моделі, усереднювали вектори для кожного слова у фразі пункту меню та знаходили найближчий тег категорії з таких наборів, як «піца», «пельмені» та «рамен». Після позначення пунктів меню вони використовували це для автоматичного генерування колекцій елементів, які потім можна рекомендувати користувачам, що відображено на рисунку 1.5.

Метрика подібності – це те, що дозволяє вам знаходити результати, які не зовсім відповідають пошуковому запиту. Для кожного властивості необхідно знати дві речі: відносну важливість властивості і функцію корисності, яка описує подібність між двома значеннями властивості. Ця функція може бути такою ж простою, як визначення різниці між двома значеннями. В цьому випадку вона була б лінійної і могла б описуватися як ціна (чим дешевше, тим краще). Однак для іншого параметра в ідеалі може

знадобитися повернення результату, максимально наближеного до запиту користувача.

І цю функцію, і вага параметра може вручну встановити фахівець в даній області, експерт. Однак їх також можна вивчити за допомогою відгуків користувачів.

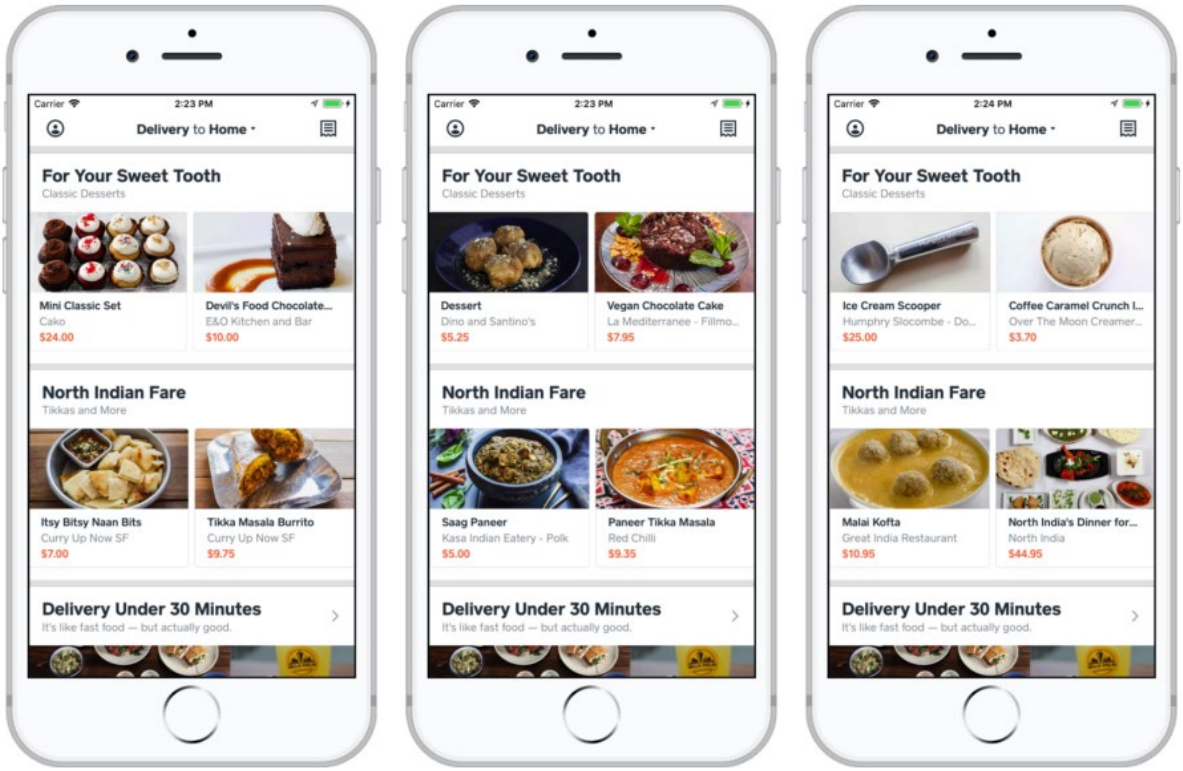


Рисунок 1.5 – Приклад автоматично згенерованих колекцій програми Caviar

Також необхідно прагнути до різноманітності результатів пошуку, щоб у користувача був вибір з безлічі елементів і можливість внесення в нього змін.

Поліпшення методів критики також може допомогти користувачеві постійно і швидко знаходити релевантні результати.

Основна небезпека в системах, заснованих на знаннях, полягає в тому, що користувач в кінцевому підсумку безцільно блукає по простору продукту і

не знаходить нічого цікавого. Пояснення рекомендацій і можливі конфлікти між вимогами можуть запобігти цьому.

Ще один простий спосіб поліпшити методи критики – запропонувати високорівневі зміни, що впливають на багато параметрів. Це і прискорює пошук, і позбавляє користувача від необхідності переводити потрібні зміни в налаштування параметрів низького рівня.

Більш складний метод – це динамічна критика. Динамічна критика намагається повернути найбільш підходящі варіанти засновані на поточних результатах. Це означає, що якщо ви шукаєте дешеве житло, але вже переглядаєте найдешевше житло з тими функціями, які вам потрібні, може не бути варіанту, який знизив би ціну і задовольнив всі ваші інші обмеження. Динамічна критика розглядає всі можливості для корегування вашого запиту і повертає найбільш часто зустрічаються шаблони, такі як «менше спалень, більш низька ціна». Шляхом пошуку загальних закономірностей у всіх результатах користувач може вибрати критику, яка дає багато результатів пошуку і, отже, багато варіантів.

## 1.3 Особливості методів побудови рекомендаційних систем, заснованих на знаннях

Шукати товари в Інтернеті – завдання непросте. Існує величезна кількість інформації, і в Інтернеті немає жодного консультанта, який би допоміг покупцям визначити, які товари краще відповідають їхнім уподобанням. Система рекомендацій, заснована на знаннях (Knowledge-Based Recommender System, KBRS), є наступним кроком у розвитку даного напрямку і видає рекомендації, засновані не на історії рейтингів користувача, а на конкретних запитах. KBRS може надати користувачеві можливість задати ряд правил або обмежень того, як повинні виглядати результат або приклад

елемента (об'єкта рекомендацій). Потім система переглядає свою базу елементів і повертає відповідні результати.

В області розробки KBRS виділяють два основних підходи: на основі прецедентів і на основі обмежень.

У рекомендаційних системах на основі прецедентів конкретні випадки вказуються користувачем як цільові точки або точки прив'язки. Метрики подібності визначаються в атрибутах елемента для вилучення елементів, схожих на ці точки. Метрики подібності часто ретельно визначаються в залежності від предметної області. Отже, показники подібності формують знання предметної області, які використовуються в таких системах. Повернуті результати часто використовуються як нові цільові прецеденти деякими інтерактивними модифікаціями користувачем. Наприклад, коли користувач бачить повернутий результат, який майже подібний до того, що він хоче, він може повторно виконати запит з цією метою, але з деякими атрибутами, зміненими на свій смак. В якості альтернативи, спрямована критика може бути вказана для видалення елементів з конкретними значеннями атрибутів, більшими (або меншими), ніж у конкретного елемента. Цей інтерактивний процес використовується, щоб направити користувача до завершальної рекомендації.

У системах на основі обмежень користувачі зазвичай вказують вимоги або обмеження (наприклад, нижні або верхні межі) для атрибутів елемента. Крім того, для відповідності вимогам користувача або атрибутам елемента використовуються правила, що залежать від предметної області. Такі правила можуть приймати форму доменних обмежень на атрибути елемента (наприклад, «Автомобілі до 1970 року не мають круїз-контролю»). Крім того, системи на основі обмежень часто створюють правила, що зв'язують атрибути користувача з атрибутами товару (наприклад, «Літні інвестори не вкладають кошти в продукти з надвисоким ризиком»). У таких випадках атрибути користувача також можуть бути вказані в процесі пошуку.

Залежно від кількості і типу повернених результатів у користувача може бути можливість змінити свої початкові вимоги. Наприклад, користувач може послабити деякі обмеження, якщо повертається занадто мало результатів, або додати додаткові обмеження, якщо повертається занадто багато результатів. Цей процес пошуку повторюється в інтерактивному режимі, поки система не приведе користувача до бажаних результатів.

В обох випадках система надає можливість користувачеві змінити свої визначені вимоги. Однак спосіб, як це робиться, відрізняється в обох випадках. У системах, що засновані на прецедентах, прецеденти (або випадки) використовуються як опорні точки для керівництва пошуком у поєднанні з метриками подібності, тоді як у системах, що базуються на обмеженнях, специфічні критерії, правила (або обмеження) використовуються для керівництва пошуком. В обох випадках представлені результати використовуються для модифікації критеріїв пошуку подальших рекомендацій. Системи, засновані на знаннях, отримали свою назву від того, що вони перетворюють різні типи знань про предметну область у вигляд обмежень, правил, метрик подібності та функцій корисності під час процесу пошуку. Наприклад, розробка метрики подібності або конкретного обмеження вимагає знань про конкретну предметну область, що є вирішальним для ефективного функціонування системи рекомендацій. Загалом, системи, засновані на знаннях, спираються на специфічні для предметної області джерела знань, порівняно із системами на основі вмісту та колаборативній фільтрації, які працюють із дещо схожими типами вхідних даних у різних доменах. Як наслідок, системи, засновані на знаннях є високо налаштованими, і їх непросто узагальнити в різних сферах. Однак ширші принципи, за допомогою яких здійснюється ця настройка, є незмінними між доменами. Ці принципи будуть розглянуті у третьому розділі.

Хоча KBRS за своєю суттю є інструментом, що вирішує конкретну проблему, вони можуть бути дуже корисними, особливо в поєднанні з іншими

формами рекомендаційних систем. KBRS можуть працювати в короткостроковій перспективі як рішення проблеми холодного запуску (відсутність необхідних початкових даних про користувача) і переключитися на колаборативну фільтрацію або систему на основі вмісту, коли будуть отримані достатні оцінки. Рекомендаційні системи, засновані на знаннях, також можуть бути цінними інструментами самі по собі, особливо коли простір елементів є складним або використовується нечасто.

Системи рекомендацій, засновані на знаннях, також можуть бути персоналізовані для окремих користувачів декількома способами.

Вивчення функцій корисності і подібності може бути персоналізоване для рекомендаційних систем на основі прецедентів і обмежень. Ви можете використовувати відгуки користувачів, щоб визначити важливість певних змінних. Ви можете відстежувати, які обмеження вибирають користувачі в своїх запитах, і пропонувати обмеження на основі загальних варіантів.

Критику можна персоналізувати, подивившись на комбінації змін в історії цього користувача. Це всього лише кілька ідей; існує безліч способів персоналізації рекомендаційних систем, заснованих на знаннях.

Системи рекомендацій, що базуються на знаннях, як правило, розроблені для предметних областей, в яких елементи мають велику кількість параметрів. У таких випадках бажано надати користувачеві більший контроль у процесі рекомендацій за допомогою специфікації вимог та інтерактивності. Системи рекомендацій, засновані на знаннях, можуть бути або системами, що базуються на обмеженнях, або можуть бути системами, що базуються на конкретних випадках. У системах, заснованих на обмеженнях, користувачі вказують свої вимоги, які поєднуються з правилами для конкретного домену для надання рекомендацій. Користувачі можуть додавати обмеження або послаблювати обмеження залежно від розміру результатів. У системах, що базуються на конкретних випадках, користувачі працюють із цілями та списками кандидатів, які ітеративно модифікуються в процесі критики. Для

отримання використовуються залежні від домену функції подібності, які також можна вивчити.

У контексті ж предметної області переважними методами є колаборативна фільтрація (CF) та засноване на вмісті (CB). Існує лише кілька спроб створення подібної системи на основі знань. Було реалізовано два підходи, у 2006 та 2012 (рисунок 1.6), які буде розглянуто у третьому розділі. У наступному ж буде проаналізовано предметну область, потенційних користувачів системи, та підходи що переважають.

| Food recommender systems | Papers | Recommendation approaches |
|---|---|---|
| Type 1: Considering user preferences | (El-Dosuky et al. 2012) | *Knowledge-based recommendation* |
| | (Freyne and Berkovsky 2010) | *CB, CF, Hybrid recommendation* |
| | (Freyne et al. 2011) | *CB, CF, Hybrid recommendation* |
| | (Svensson et al. 2000) | CF |
| | (Elahi et al. 2015) | *Matrix factorization* |
| | (Kuo et al. 2012) | *Graph-based recommendation* |
| Type 2: Considering nutritional needs of users | (Ueta et al. 2011) | *Goal-oriented recipe recommendation* which suggests the right type of nutrient to treat users' health problems |
| | (Aberg 2006) | *Hybrid recommendation (CB & CF), Constraint-based recommendation* |
| Type 3: Balancing between user preferences and nutritional needs of users | (Elsweiler et al. 2015) | *CB* |

Рисунок 1.6 – Дослідження у предметній області рекомендаційних систем для кулінарних рецептів

Модифікації запитів досягаються використанням критики. Критика може бути простою, складеною або динамічною. Системи, засновані на

знаннях, значною мірою базуються на вимогах користувачів і включають лише обмежену кількість даних із історії користувачів. Тому вони, як правило, ефективні при вирішенні питань холодного запуску. Недоліком цього підходу є те, що інформація з історії не використовується для «заповнення прогалин». В останні роки також були розроблені методи для включення більшої кількості персоналізації з використанням інформації з історії із сеансів користувачів.

## 1.4 Постановка задач дослідження

Метою даної кваліфікаційної роботи є дослідження методів побудови рекомендаційних систем у цілому, зокрема заснованих на знаннях.

З урахуванням сформульованої мети в даній атестаційній роботі необхідно вирішити низку таких задач:

– проаналізувати методи що переважають у рекомендаційних системах;

– виділити їх переваги та недоліки;

– провести аналіз предметної області, існуючих систем у її контексті; методів критики та метрик подібності, що використовуються в них;

– виділити потенційних користувачів;

– виділити обмеження, які можна до неї застосувати;

– на основі цього підібрати підходящий метод побудови рекомендаційної системи;

– провести аналіз методу, заснованого на прецедентах та методу; заснованого на обмеженнях;

– дізнатися, чи можливо реалізувати їх у контексті предметної області самостійно, чи використовуючи певну гібридну модель.

В результаті має бути побудовано прототип рекомендаційної системи, що зможе рекомендувати рецепти користувачам, виходячи із того, які вхідні дані користувачі будуть надавати системі.

# 2 ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ

## 2.1 Кулінарні рецепти

Харчування та дієта – це складні сфери, що створюють багато проблем для рекомендаційних технологій. Для надання рекомендацій необхідно зібрати тисячі продуктів/інгредієнтів. Крім того, оскільки продукти/інгредієнти, як правило, поєднуються між собою в рецепті, а не споживаються окремо, це експоненціально збільшує складність системи рекомендацій [6]. Крім того, системи рекомендацій їжі не лише рекомендують їжу, яка відповідає уподобанням користувачів, але й пропонують вибір здорової їжі, відстежують харчову поведінку та розуміють проблеми зі здоров’ям.

«Куди нам піти на обід?» або «Що нам їсти на вечерю?» це звичайні запитання, на які нам доводиться дуже часто відповідати. Хоча багато рекомендаційних систем намагалися лише відповідати уподобанням користувачів музичним, кіно або книжковим предметним областям, останнім часом вони також застосовуються в харчовій галузі, щоб дати надійні відповіді на вищезазначені питання. Наприклад, RecipeKey2 – це система рекомендування продуктів харчування, яка фільтрує рецепти на основі врахування улюблених інгредієнтів, наявної харчової алергії та описів продуктів (наприклад, типу їжі, кухні, часу приготування тощо), вибраних користувачами. Що стосується споживання їжі в наші дні, помітно, що спостерігається збільшення захворювань, пов'язаних із способом життя, таких як діабет та ожиріння, які є причиною багатьох хронічних захворювань. Цю проблему можна покращити, застосовуючи відповідний дієтичний режим [12]. У цьому контексті системи рекомендацій їжі також досліджуються як потенційний засіб, який допомагає людям живитись здоровіше [13]. Має сенс використовувати системи рекомендацій щодо їжі як частину стратегії зміни

поведінки користувачів у харчуванні. У цьому випадку системи, що рекомендують їжу, не тільки вивчають переваги користувачів щодо інгредієнтів та стилів харчування, але й вибирають здорову їжу, беручи до уваги проблеми зі здоров'ям, харчові потреби та попередню поведінку в їжі. Як згадувалося в [14] існує три типи систем, що рекомендують їжу. Системи першого типу рекомендують здоровіші рецепти або продукти харчування, які найбільш схожі на ті, що сподобались користувачеві в минулому. Другий тип систем рекомендацій рекомендує користувачам лише ті предмети, які були попередньо визначені працівниками охорони здоров'я. Третій тип формує рекомендації на основі врахування обох вищезазначених критеріїв з метою збалансування їжі, яка подобається споживачу із їжею, яку споживачу необхідно споживати. Усі ці типи рекомендаційних систем в основному розроблені для окремих користувачів. Також існує система, що представляє групові рекомендації, в яких продукти харчування споживають групи користувачів, а не окремі особи.

У галузі здорової їжі вивчення смаків користувачів визнається важливим обов'язковим кроком для пропонування страв, які сподобаються користувачам. Для цього проводяться дослідження, спрямовані на рекомендацію продуктів харчування або меню окремим користувачам на основі вивчення смаків користувачів. Більшість із них використовують популярні методи рекомендацій [6], [15], [16] та/або поєднують з різними методами з метою покращення якості рекомендацій [17], [18].

Найпростішою системою рекомендацій щодо харчування [15] з простим сценарієм є система, яка рекомендує лише окремі продукти харчування для користувачів. Автори використовують метод вилучення термінів TF-IDF (Term Frequency-Inverse Document Frequency) для створення профілю користувача та застосовують деякі обчислення для виявлення подібності між рецептом та профілем користувача. Крім того, у базу знань включено бази даних про здорові та стандартні продукти харчування, які були отримані з

Міністерства сільського господарства США (USDA). База знань – це онтологія доменів, що складається з класів, взаємозв’язків та екземплярів класів. Для отримання рекомендацій кожен користувач вручну оцінює продукти харчування певної категорії (наприклад, фрукти, овочі, м’ясо тощо) як відповідні чи невідповідні його інтересам. Після цього система обчислює схожість між продуктами харчування та раніше розрахованим профілем користувача. Якщо значення подібності перевищує заздалегідь визначений поріг, харчовий продукт рекомендується, інакше він ігнорується. В іншому дослідженні Freyne та ін. [6] використовують content-based алгоритм для прогнозування рейтингового значення для цільового рецепта на основі використання інформації про відповідні інгредієнти, включені в цей рецепт.

Рейтингове значення користувача для конкретного інгредієнта в цільовому рецепті обчислюється як сума рейтингових значень користувача для всіх інших рецептів, які містять інгредієнт, поділене на кількість рецептів, що містять цей інгредієнт.

Прогнозоване рейтингове значення користувача для цільового рецепта рахується на основі середнього значення всіх рейтингових значень усіх інгредієнтів включених до цього рецепту.

Користувачеві будуть рекомендовані рецепти з високим прогнозованим значенням рейтингу. Ілюстрація для прогнозування рейтингового значення для цільового рецепта представлена на рисунку 2.1.

Elahi та ін. [17] пропонують систему рекомендацій щодо їжі, використовуючи активний алгоритм навчання та факторизацію матриць. Це дослідження надає користувачам повну взаємодію із системою з метою збору довгострокових уподобань користувачів щодо рейтингу рецептів та тегів. Крім того, у процесі користування рекомендаційною системою, користувачі повинні надати короткочасні уподобання щодо інгредієнтів, які вони хочуть приготувати або включити в їжу. Потім система використовує обидва типи вподобань користувачів для вироблення рекомендацій. Довгострокові

переваги використовує модель прогнозування рейтингу факторизації матриці, розроблена для врахування як тегів користувачів, так і рейтингів. Кожен користувач та кожен рецепт змодельовані векторами, що відображають їхні приховані особливості. Рейтингове значення користувача для конкретного товару оцінюється шляхом обчислення продукту користувача та векторів елементів. З короткочасними уподобаннями система фільтрує рецепти відповідно до поточних уподобань користувачів. Користувачам рекомендуються рецепти із найвищими рейтинговими значеннями.

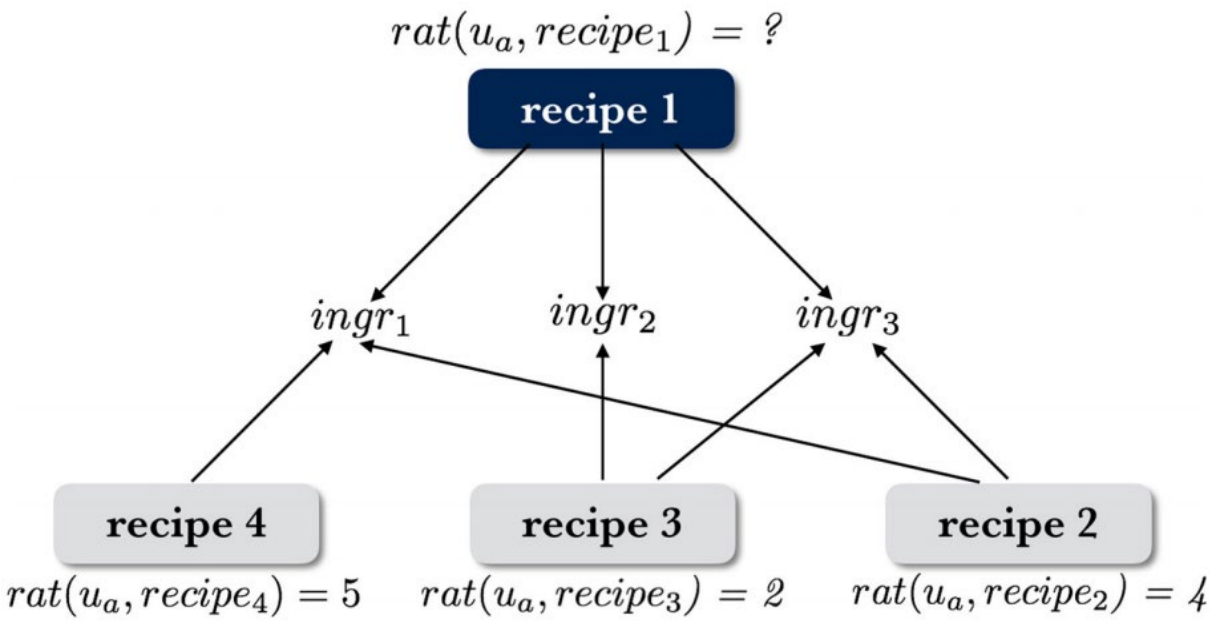


Рисунок 2.1 – Прогнозування рейтингового значення для цільового рецепта

У цьому прикладі прогнозоване значення буде представлене як

$$pred(u, recipe1) = \frac{rat(u, ingr_1) + rat(u, ingr_2) + rat(u, ingr_3)}{3} = \frac{((5+4)/2) + 2 + ((4+3/2))}{3} = 3.166 \quad (2.1)$$

Нещодавно деякі нові підходи були включені до систем, що

рекомендують їжу, наприклад, використання ярликів (тегів) для різних груп користувачів [16] алгоритмів активного навчання та розкладання матриць [17]. Зокрема, у роботі [16], автори розробляють інтернет-магазин харчових продуктів з метою запропонувати види їжі, які слід купувати користувачам. На основі рецептів, які користувачі обирали раніше, групи користувачів маркуються та називаються відповідно до їх змісту, наприклад, «Любителі м’яса», «Вегетаріанці», «Любителі спецій» тощо. Рекомендовані рецепти визначаються на основі трьох різних характеристики, обрані користувачами: групи користувачів, категорії продуктів харчування (наприклад, риба, східна, італійська, червоне м’ясо, курка) та інгредієнти (наприклад, рис, спагеті, карі, помідори). Користувачі вибирають рецепти зі списку рекомендацій та складають їх у кошик для покупок. Потім усі інгредієнти вибраних рецептів автоматично додаються до списку продуктів, який доставляється до порогу користувача. Крім того, для покращення соціальної взаємодії рецептів, до кожного рекомендованого рецепта додаються деякі додаткові функції (наприклад, середнє значення оцінки або коментарі інших користувачів).

Хоча більшість існуючих досліджень у галузі продуктів харчування зосереджуються лише на рекомендаціях щодо продуктів харчування чи рецептів, існує потреба в тому, щоб користувачі планували меню з поєднанням багатьох рецептів у повноцінні страви. З цією ідеєю Kuo та ін. [18] пропонують розумний механізм планування меню, який пропонує набір рецептів за допомогою графічного алгоритму. Спочатку будується ненаправлений графік рецептів, де кожен вузол є рецептом, що містить набір інгредієнтів, кожен край представляє взаємозв'язок між двома рецептами, а вага краю представляє відстань між двома рецептами (рисунок 2.2). Вага кожного краю, що з'єднує два різні рецепти, описує вартість меню, яке включає ці два рецепти. Чим менша вага, тим вища ймовірність того, що два рецепти будуть спільними в меню.

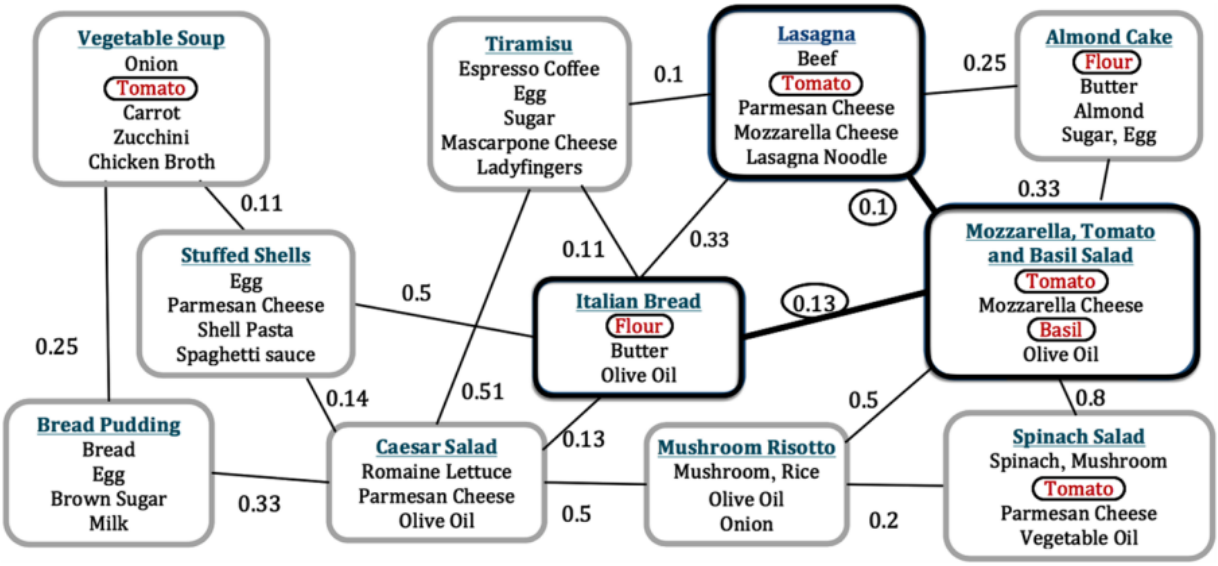


Рисунок 2.2 – Приклад графу рецептів для планування меню

Крім того, вартість меню також визначається як зважена сума ребер мінімального охоплюючого дерева на індукованому підграфі. На основі цього, план меню створюється шляхом вибору набору рецептів, який містить усі інгредієнти запиту (тобто інгредієнти, що необхідні користувачам), а вартість меню мінімальна.

Наприклад, з інгредієнтами запиту {помідор, борошно, базилік}, ми можемо знайти багато різних наборів рецептів, наприклад, {«Салат з моцарелою, помідорами та базиліком», «Лазанья», «Італійський хліб»}, {«Моцарела, помідори та базилік», «Лазанья», «Мигдаль торт»}, {«Моцарела, помідор та базилік», «Італійський хліб», «Салат зі шпинату»},тощо.

Однак перший набір {«Салат з моцарелою, помідорами та базиліком», «Лазанья», «Італійський хліб»} буде рекомендований користувачам, оскільки його загальна вартість меню мінімальна (тобто 0,23).

## 2.2 Обмеження, які можна застосувати до системи

Для роботи системи необхідно вирішити проблему задоволення обмежень, яка подібна до підходу, заснованого на параметрах [19], щоб запропонувати рецепт на основі врахування уподобань користувача. У цьому прикладі ми припускаємо, що змінні використовуються для представлення параметрів рецепта, таких як time, cost, energy, protein, allergies, disease, де: time (у хвилинах) – час підготовки рецепту, cost (у євро) – вартість рецепта, energy (у ккал) – харчова цінність рецепта, protein (у %) – відсоток білка, що міститься в рецепті, allergies – це набір алергічних інгредієнтів користувачів, а disease – проблеми зі здоров'ям користувачів. Кожна змінна має відповідне визначення домену, наприклад, dom(time) = [1..60]. Крім того, база знань CKB (Constraint Knowledge Base) включає обмеження, що використовуються для опису бази знань. Наприклад, time < 60 означає той факт, що час приготування рецепта повинен бути менше 60 хвилин.

Обмеження можуть бути жорсткими та м'якими. Жорсткі обмеження – (алергія = морепродукти ⇒ інгредієнти != морський краб) представляє знання про те, що якщо у користувача алергія на морепродукти, то морського краба не слід включати до рекомендованих рецептів.

М'які обмеження – для різноманітності рецептів не будуть обрані рецепти, що містять багато подібних до попередніх страв інгредієнтів (наприклад, яловичина та картопля сьогодні не будуть вибирати на вечерю, оскільки вони вже були спожиті в обід). Рецепти з високим прогнозованим рейтингом матимуть більшу ймовірність рекомендувати їх користувачам.

На основі цього формується набір користувацьких уподобань, який повинен узгоджуватися з CKB таким чином, щоб можна було знайти відповідне рішення.

Таким чином можна виділити наступний список змінних, їх доменів та

правил бази знань, що складатимуть обмеження системи:

а) змінні: вік користувача, національність, алергії, захворювання, вегетаріанство, інші обмеження (наприклад, релігійні), допустима кількість калорій, дієта, кулінарні навички, складність страви (кількість кроків для приготування страви), калорійність страви, харчова цінність страви, час на приготування, розмірність страви, кухня страви, номер страви, інгредієнти, теги;

б) домени: вік користувача – чисельна, національність – категорична, алергії – категорична, захворювання – категорична, вегетаріанство – двійкова, інші обмеження (наприклад, релігійні) – категорична, допустима кількість калорій – чисельна, дієта – складена (жири – чисельна, сахари – чисельна, натрій – чисельна, білки – чисельна, насичені жири – чисельна), кулінарні навички – категорична, складність страви (кількість кроків для приготування страви) – чисельна, калорійність страви – чисельна, харчова цінність страви – складена (жири – чисельна, сахари – чисельна, натрій – чисельна, білки – чисельна, насичені жири – чисельна), час на приготування – чисельна(хв), розмірність страви – категорична, кухня страви – категорична, номер страви – категорична(перша, друга, головна, салат, десерт), інгредієнти – категорична, теги – категорична (містить характеристики, що складно віднести до іншої категорії, наприклад святкова їжа);

в) правила:

1) «Якщо користувач вегетаріанець, то не рекомендувати йому страви, що не є вегетеріанськими»;

2) «Якщо користувач на дієті, не рекомендувати йому страви, калорійність яких перевищує задану користувачем»;

3) «Якщо в користувача певна алергія, не рекомендувати йому страви, що містить його алергіки»;

4) «Якщо користувач хоче приготувати певний вид страви, рекомендувати йому страви саме цього виду»;

5) «Якщо користувач хоче приготувати страву певної кухні, рекомендувати йому страви саме цієї кухні»;

6) «Якщо користувач хоче, щоб страва містила певні інгредієнти, рекомендувати йому страви найбільш подібні до тих, що містять усі інгредієнти, задані користувачем»;

7) «Якщо користувач має невеликий досвід у кулінарії, рекомендувати йому страви із невеликою кількістю кроків та часом приготування», тощо.

Більше інформації щодо змінних, їх доменів та повний список знань предметної області у вигляді правил представлено у розділі 4.

## 2.3 Користувачі системи та варіанти її використання

Їжа є важливою частиною дня кожного і має велике соціальне, культурне та релігійне значення. Однак для людей завжди було проблемою обрати, що саме їсти. По-перше, їх може просто вразити вибір [20]. У містах голодна пара може зіткнутися з безліччю місцевих ресторанів, які потрібно відвідати, та багатьма іншими для доставки їжі через Інтернет. Мандрівник у місті на конференції хоче вивчити найкращі місцеві страви. Куди їм їхати? Любитель кулінарії хоче спекти новий торт, але який? На вибір є тисячі! По-друге, люди можуть боротися з обмеженнями. Людям на дієті може знадобитися допомога, щоб знайти смачний і корисний вибір. Лікар хоче запропонувати найкращу неповторювану дієту, щоб допомогти своєму пацієнту відновитись після лікування та побічних ефектів, що залежать від дієти [21]. Мати запитує, що можна зробити, враховуючи, що «у мене в холодильнику яловичий фарш», «вишня зараз в сезоні», або «лимони у продажу»?

Враховуючи складність простору інгредієнтів, продуктів харчування та ресторанів, багаторазове використання, нюанси особистих уподобань та обмеження здоров’я та релігійних дієт, не дивно, що існує багато

рекомендаційних систем, що рекомендують їжу [22]. Зважаючи на це, можна виділити багато користувачів подібних систем. Це просто голодні люди, що можуть підібрати собі їжу за смаком та потім замовити цю їжу. Це ті, хто тільки почав готувати, і хоче знайти собі простий рецепт щоб набратися кулінарного досвіду. Це любителі кулінарії, що шукають рецепт для приготування страви власноруч. Це ті, хто слідує здоровому харчуванню, і може обрати собі здорові, поживні страви. Це ті, хто слідує дієти, і може обрати собі низькокалорійну, але ситну і корисну їжа. Це пацієнти, яким лікарі пропонують дотримуватися цієї дієти. Це алергіки чи ті, хто не може споживати певні продукти із інших причин, і можуть знайти тут страви, що задовольняють їх потреби.

Любителі кулінарії хочуть хороших рекомендацій щодо рецептів, які вони могли б приготувати. Зазвичай їм потрібні рекомендації, що враховують їх смак, але вони також можуть обмежити результат, встановлений такими аспектами, як кухня, час приготування, кількість інгредієнтів та складність приготування. Ключовим випадком використання, що ґрунтується на обмеженнях, є рекомендації, засновані на певному інгредієнті, наприклад, «що я повинен приготувати, враховуючи те, що у мене в холодильнику курячі стегна?» Є кілька випадків використання, не тільки те, що є в коморі, але й те, що доступне через сезонність чи ціну. Кухарі також можуть отримати рекомендації щодо додаткових інгредієнтів, «що добре поєднується з лососем?» або вони можуть отримати рекомендації за станом здоров’я [23]. Наприклад, користувач може захотіти оздоровити страву, замінивши інгредієнт інгредієнтом, який має подібний смаковий профіль, але має меншу калорійність, менше жиру тощо [24]. Багато людей мають певні дієтичні обмеження. Певною мірою це можна зробити за допомогою простого індивідуального фільтра, наприклад, «рекомендувати вегетаріанські страви». Інші випадки вживання здоров’я вимагають рекомендувати набір страв, тобто дієту (мається на увазі в широкому розумінні: види їжі, які людина звично

їсть) або тижневий план харчування.

## 2.4 Проблематика предметної області

Наявні дослідження рекомендаційних систем, що рекомендують їжу, відіграють вирішальну роль у підтримці користувачів у виборі дієти, яка відповідає інтересам та стану здоров'я. Ці дослідження використовують інформацію про профілі та рецепти користувачів, щоб отримати рекомендації щодо їжі. Було визначено, що на якість рекомендацій сильно впливає адекватність та точність інформації про користувачів, а також інформація про харчові продукти. Крім того, хоча деякі документи [19], [25] пропонують харчові рекомендації для вирішення проблем зі здоров’ям, пропозиції щодо зміни харчової поведінки, що є передумовою для підтримки здорового способу життя, все ще відсутні. Пояснення можуть допомогти користувачам більше довіряти рекомендаціям та заохочувати їх дотримуватися належних харчових звичок, однак включення пояснень до систем рекомендацій щодо їжі не викликало інтересів дослідників. Крім того, дослідження систем, що рекомендують їжу, в основному зосереджені на сценаріях для одного користувача, а не на групових сценаріях. До цього часу дослідження систем, що рекомендують групи в галузі здорової їжі, дуже обмежені. Існують дослідження, які пропонує деякі стратегії агрегування для формування рекомендацій щодо їжі для груп користувачів. У цьому розділі буде розглянуто проблеми дослідження в системах, що рекомендують їжу, та запропоновано деякі потенційні рішення.

Проблеми щодо інформації про користувача – це невизначеність харчової інформації від користувачів: для того, щоб робити рекомендації, система повинна збирати харчові потреби, рейтинги харчових продуктів / рецептів та інформацію про попередні страви від користувачів. Більша частина інформації надається лише завдяки постійній взаємодії з користувачами.

Однак насправді, реєструючи споживання їжі споживачами, не можна уникнути помилок, оскільки користувачі зазвичай забувають або дають неправильну інформацію про спожиту їжу. Хоча для вирішення цих проблем було запропоновано деякі системи, наприклад, FOODLOG, вони не можуть надати точну інформацію про спожиті страви, хоча вони можуть оцінити харчовий баланс між різними видами страв. Також проблемою є збір даних про рейтинг користувачів. Системам, що рекомендують їжу, потрібна інформація про уподобання користувачів, щоб рекомендувати подібні продукти харчування [26]. Цю інформацію можна зібрати, попросивши користувачів оцінити продукти / рецепти. Однак це не зручно, якщо система просить користувачів оцінити занадто багато елементів. Отже, виклик пов'язаний із цим: «Як зібрати достатню кількість рейтингів користувачів, заощадивши їх зусилля?». Крім того, подібно до ведення звітності про споживання їжі (як уже згадувалося вище), переконання користувачів дотримуватися рейтингу страв також стає ще однією проблемою для подібних рекомендаційних систем.

Для того, щоб розрахувати харчові рекомендації для користувачів, будь-якому алгоритму потрібна певна інформація. По-перше, це інформація про користувача (наприклад, оцінки «подобається», «не подобається», споживання їжі чи харчові потреби). Подібно до рекомендаційних систем в інших сферах, подібні системи в сфері харчування також стикаються з проблемою холодного старту, коли система використовується вперше. Цю проблему можна подолати, використовуючи інформацію про попередні страви користувачів, щоб обчислити подібність, а потім рекомендувати користувачам нові рецепти [26]. Однак це рішення вимагає багатьох зусиль користувача та зменшує бажання користуватися системою. По-друге це бази даних рецептів. Необхідно вирішити, скільки рецептів повинна мати система, та як зібрати точну харчову інформацію про рецепти. Кількість зібраних рецептів має бути достатньо великою, щоб задовольнити уподобання багатьох користувачів і

мати варіації рекомендованих рецептів, при цьому мінімізуючи час на створення рекомендацій. Це складна проблема, коли система намагається збалансувати різноманітність рекомендацій та час реакції системи. Hoxmeier та ін. [27] зазначають, що тривалий час відгуку викликає невдоволення користувачів, та це ще більше зменшує постійне використання системи. Помічено, що з одним і тим же продуктом харчування, якщо ми використовуємо різні способи його приготування, тоді ми отримаємо від нього різні харчові цінності. Тому дуже важко переконатися, що зібрані таблиці поживних речовин для харчових продуктів є точними, оскільки при порівнянні таблиці різних харчових цінностей харчових продуктів іноді вона повертає різні значення для тих самих харчових продуктів. Наприклад, харчова цінність селери у рецепті салату відрізняється від харчової цінності самого у рецепті картоплі фрі, оскільки готування з високою температурою змушує селеру втрачати велику кількість ефірної олії. Це означає, що кількість ефірної олії селери у рецепті картоплі фрі може бути нижчим, ніж у рецепті салату.

Також система має мати певний набір обмежень або правил. Додаток більшої кількості обмежень та правил у процесі рекомендацій покращить якість рекомендацій. Наприклад, користувачеві, який страждає на серцеві захворювання, система повинна рекомендувати меню з меншим вмістом жиру та солі. Більше того, дуже необхідно виявляти конфлікти між обмеженнями або правилами, які заважають алгоритмам рекомендацій знаходити рішення. Однак, завдяки великій базі даних (наприклад, тисячі продуктів / рецептів), перевірка обмежень / правил у базі даних призводить до негативних наслідків для продуктивності системи. Крім того, системи рекомендацій продуктів харчування повинні враховувати обмеження щодо доступності інгредієнтів, щоб допомогти користувачам заощадити гроші та запобігти поведінці харчових відходів. Проблема тут полягає в тому, як запропонувати їжу, яка відповідає ситуаціям зі здоров’ям та харчовим потребам користувачів, а також скористатися перевагами інгредієнтів, які зараз є в холодильнику. У цьому

випадку системи, що рекомендують, вимагають багато зусиль від користувачів, тому що користувачі повинні регулярно повідомляти про споживання всіх інгредієнтів, і це може заважати користувачам постійно використовувати систему.

У наш час багато людей страждають від проблем зі здоров'ям через невідповідні харчові звички [28]. Наприклад, деякі люди їдять занадто багато їжі порівняно з рівнем фізичної активності і поступово страждають ожирінням. Тоді як інші (наприклад, літні люди, які дотримуються дієти) вкрай обмежують споживання їжі, і це призводить до недоїдання. Тому однією з основних функцій рекомендаційних систем, що рекомендують їжу є розуміння харчової поведінки користувачів та переконання їх змінити харчову поведінку позитивно. Однак це є великим викликом для подібних рекомендаційних систем, оскільки їжа – це поведінка протягом усього життя, на яку впливає безліч факторів, особливо психологічних. Отже, системи, що рекомендують їжу, повинні інтегрувати теорію психології здоров'я, щоб стимулювати користувачів дотримуватися поведінки здорового харчування. Перший підхід можна використовувати, застосовуючи одну просту зміну в певний час, поки поведінка користувача не стане звичною [28]. Інший підхід може бути застосований до систем, що рекомендують їжу, порівняння з ідеальною поживною речовиною. Користувачі можуть знайти структуру ідеального харчування відповідно до віку та рівня фізичної активності на надійних ресурсах (наприклад, USDA), а потім порівняти, яку їжу вони їли, та рекомендовану [28]. Більше про методи рекомендацій, що засновані на знаннях буде розглянуто у наступному розділі.

# 3 ОПИС МЕТОДІВ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ, ВИКОРИСТАНИХ У РОБОТІ

## 3.1 Метод, заснований на прецедентах

Основою цього метода побудови систем є case-based reasoning (CBR) або міркування на основі прецедентів. Міркування на основі прецедентів – це методологія вирішення проблем, яка намагається вирішити проблеми шляхом повторного використання конкретного минулого досвіду, що зберігається у прецедентах-прикладах [29]. Справа моделює минулий досвід, зберігаючи як опис проблеми, так і рішення, застосоване в цьому контексті. Усі справи зберігаються в базі справ. Коли перед системою постає нова проблема, яку потрібно вирішити, вона шукає найбільше подібні випадки (кейси) у базі прецедентів та повторно використовує адаптовану версію отриманого рішення для вирішення нової проблеми.

CBR – це циклічний та інтегрований процес вирішення проблем (рисунок 3.1), який підтримує навчання на попередньому досвіді [30] і має чотири основні етапи: отримання, повторне використання, адаптація та збереження. Фаза адаптації розділена на два підкроки: перегляд та перегляд. На етапі перегляду система адаптує рішення відповідно до конкретних обмежень нової проблеми. Тоді як на етапі огляду побудоване рішення оцінюється шляхом застосування його до нової проблеми, розуміння того, де воно не вдається, та внесення необхідних виправлень.

Наприклад, задля завдання діагностики система отримує симптоми для пацієнта (нова проблема) і намагається поставити діагноз на основі прикладів минулого пацієнта (зберігається в базі справи). Іноді отримане рішення може бути прямо використане повторно в новій задачі, але в більшості випадків отримане рішення не застосовується безпосередньо і повинно бути адаптоване до конкретних вимог нової проблеми. Після цієї адаптації система створює

новий випадок і може зберегти його в основі справи (навчання).

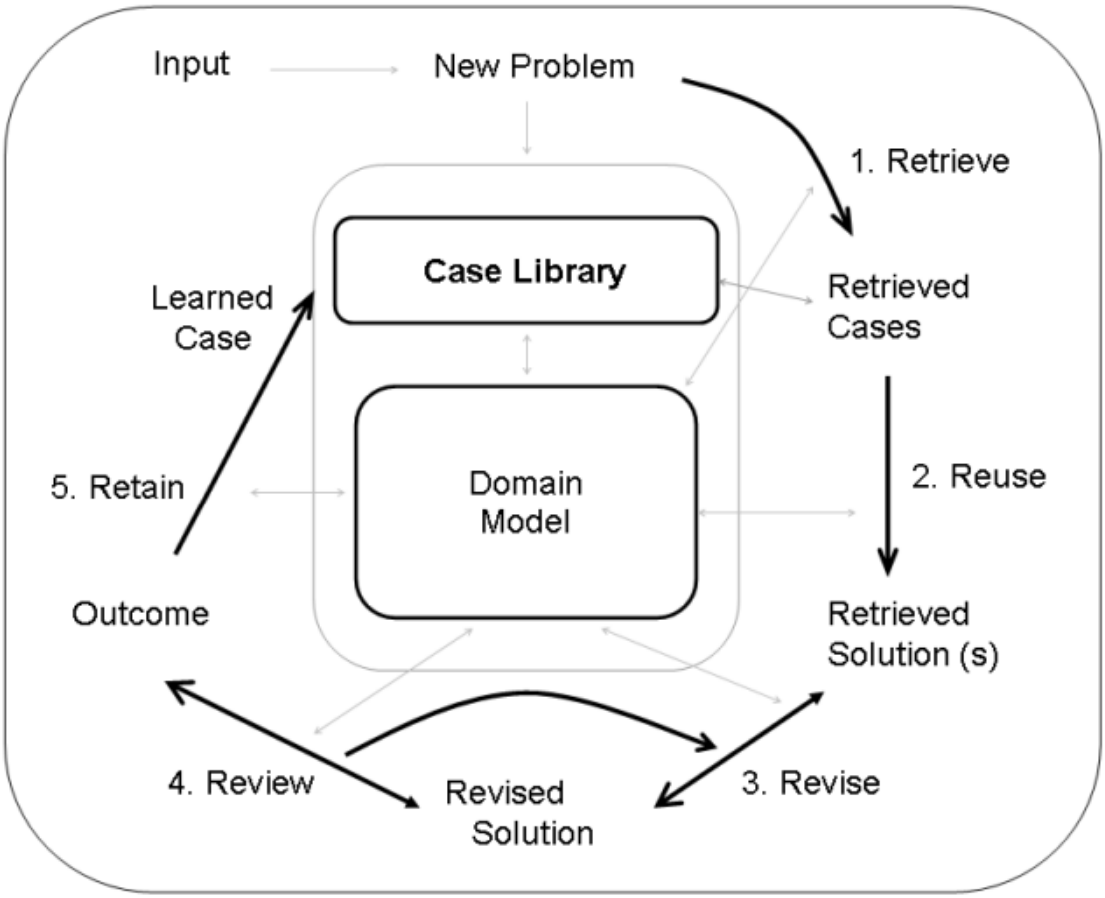


Рисунок 3.1 – Цикл вирішення проблеми за допомогою CBR

Основною проблемою CBR є модель прецедента. Необхідно вирішити, які атрибути повинні складати випадок і яка мова подання найкраще підходить для представлення конкретних знань, задіяних у процесі вирішення проблеми. Отже, завдання представлення прецедента складається з (1) вибору відповідних атрибутів, (2) визначення індексів та (3) структурування знань у конкретному випадку реалізації. Індексація пов’язана зі створенням додаткових структур даних, які можуть зберігатися в пам’яті, щоб пришвидшити процес пошуку, зосередившись на найбільш відповідних вимірах. Індекси визначають атрибути справи, які слід використовувати для

вимірювання подібності справ. Більше того, індекси можуть пришвидшити процес пошуку, забезпечуючи швидкий доступ до тих справ, які необхідно порівняти з проблемою вхідних справ. Наприклад, у системі медичної діагностики, якщо система повинна проводити діагностику інфекції, тоді такі атрибути, як професія, стать або вік, ймовірно, менш важливі, ніж атрибути, що описують симптоми.

Далі розглянемо використання методу саме у рекомендаційних системах.

У процесі найпростішого рекомендації користувач повинен шукати товар, який йому потрібен, і тому система просить надати деякі вимоги до товару, які він вважає найбільш важливим.

У відповідь система ініціює пошук у базі прецедентів, щоб визначити товари, які слід рекомендувати, тобто ті, що відповідають цим вимогам. У цьому процесі ми можемо визначити деякі основні елементи, такі як вхідні дані (де користувач надає свої вимоги), пошук продуктів (де система здійснює пошук продуктів відповідно до вимог користувача) та вихідні дані, де дається певна рекомендація користувачеві.

Як було описано раніше, ці етапи дуже схожі на етапи звичайної CBR-системи. Процес починається з нової проблеми, отримує подібні випадки з бази справ, показує отримане рішення користувачеві або пристосовує його для кращого вирішення нової проблеми та завершує процес, зберігаючи новий випадок.

Першим етапом багатьох методів рекомендацій є введення даних, коли система взаємодіє з користувачем, щоб визначити її уподобання. Існують різні стратегії взаємодії з користувачем. Найпопулярнішою стратегією є діалогова, коли система пропонує користувачеві вказівки, задаючи питання та представляючи альтернативні варіанти продуктів, щоб допомогти користувачеві прийняти рішення. Кілька рекомендаційних систем, на основі прецедентів вимагають від користувачів вимоги, щоб мати уявлення про те,

що шукає користувач. Наприклад, у компромісному пошуку (Compromise-Driven Retrieval) [31] користувач задає такі атрибути персонального комп'ютера, які він шукає (наприклад, тип, ціна, процесор або швидкість). Expertclerk [32] просить безпосередньо користувача відповісти на деякі запитання і, отже, надати особливості справи як відповідь на ці запитання.

Зазвичай, коли користувач шукає товар, можуть виникнути три ситуації:

– користувач точно знає, що він хоче;

– користувач має бажання, але не знає назви товару;

– користувач не знає, що саме він шукає.

У кожній із цих ситуацій, щоб рекомендувати відповідні альтернативні продукти, система вимагає певних знань. У рекомендаційних системах, заснованих на знаннях знання в основному зберігаються в основі справи, і, аналізуючи існуючі рекомендаційні системи, засновані на знаннях, було помічено, що знання, що містяться у справі, можуть стосуватися багатьох характеристик проблемної області. Насправді, в системах, які розглядались у цій роботі, прецедент зберігає інформацію про: товари, що рекомендуються (або мають бути рекомендовані), користувача, якому була надана рекомендація, та контекстну інформацію про сеанс рекомендацій, коли рекомендація була надана. Насправді системи використовують ці основні інгредієнти, щоб визначити власну конкретну модель випадку як їх поєднання. Отже, наприклад, в одній моделі ми можемо знайти опис рекомендованих продуктів та користувача, який отримав рекомендацію, або користувача разом із оцінкою рекомендованих продуктів.

Наступним кроком рекомендаційного циклу є фаза пошуку. Це, як правило, основна фаза, і більшість систем, що рекомендують CBR, можна охарактеризувати як складні пошукові системи. Наприклад, в Order-Based Retrieval система використовує спеціальні оператори для отримання решітки справ, або в Compromise-Driven Retrieval система не тільки отримує подібні прецеденти із бази прецедентів, але також групує їх, складаючи разом ті

випадки, які пропонують однаковий компроміс для користувача і представляють користувачеві лише репрезентативну справу для кожної групи.

Після отримання на етапі повторного використання розглядається рішення справи, і система оцінює, чи можна його повторно використовувати в поточній проблемі, або яку частину справи можна використовувати повторно. У найпростіших CBR-RS система повторно використовує отримані кейси / продукти, показуючи їх користувачеві. У більш досконалих рішеннях, таких як Interest Confidence Value [33] або DieToRecs, отримані випадки не рекомендуються, але використовуються для ранжирування продуктів-кандидатів, ідентифікованих з іншими підходами, наприклад, у DieToRecs, за допомогою інтерактивного компонента управління запитами.

На наступному етапі перегляд повторно використаного випадку адаптується для кращого вирішення нової проблеми. Етап перевірки в рекомендаційних системах, заснованих на знаннях реалізований можливістю користувачеві налаштувати отриманий набір продуктів. Наприклад, у DieToRecs користувач може додати до поточного випадку інші продукти або за допомогою функцій CBR, або за допомогою інших системних функцій (наприклад, перегляд каталогу продукції).

Крок ітерації дуже часто реалізується в розмовних системах. Наприклад, в Entree [34] система дозволяє користувачеві налаштувати початковий запит та шукати товари, що мають незначні відмінності від тих, що вже показані, щодо деяких характеристик товару (наприклад, дешевші товари). У Comparison Based Retrieval [35] система просить користувача надати зворотній зв'язок, позитивний чи негативний, про отриманий товар та автоматично оновлює запит користувача, використовуючи цю інформацію.

Останнім кроком циклу рекомендацій щодо CBR є фаза збереження (або вивчення), коли новий випадок зберігається у базі прецедентів. Наприклад, у DieToRecs всі сеанси рекомендацій користувача / системи зберігаються як нові прецеденти в основі справи.

Приклади популярних рекомендаційних систем на основі прецедентів:

– Entree (EN): система рекомендацій, яка використовує налаштування запитів, щоб рекомендувати ресторани користувачеві;

– DieToRecs: система рекомендацій щодо подорожей, яка пропонує як окремі туристичні послуги (наприклад, готель чи захід), так і повні плани подорожей, що включають більше однієї елементарної послуги;

– First Case : прототип системи, що використовує техніку компромісного керування для отримання та групування справ відповідно до альтернативних компромісів, знайдених системою;

– Expertclerk: інструмент для розробки систем рекомендацій на основі діалогу для веб-сайтів електронної комерції.

Для того, щоб система рекомендацій, що базується на конкретних випадках, працювала ефективно, є два найважливіші аспекти системи, які повинні бути ефективно розроблені:

– метрики подібності: ефективна конструкція метрик подібності дуже важлива в системах, що базуються на прецедентах, для отримання відповідних результатів. Важливість різних атрибутів повинна бути належним чином включена в функцію подібності для ефективної роботи системи;

– методи критики: інтерактивне дослідження простору предметів підтримується із використанням методів критики. Для підтримки різних цілей розвідки доступні різноманітні методи критики.

## 3.2 Метрики подібності

Правильний дизайн метрик подібності є важливим для отримання найбільш відповідних результатів у відповідь на конкретний запит. Найбільш ранні системи впорядковували атрибути за зниженням рівня важливості і спочатку сортували за найважливішим критерієм, потім за наступним по важливості, тощо. Наприклад, у системі рекомендацій ресторану Entree

перший сорт може базуватися на типі кухні, другий на ціні тощо. Хоча цей підхід ефективний, його використання може бути неефективним для кожної предметної області. Загалом бажано розробити функцію схожості закритої форми, параметри якої можуть або встановлюватися експертами доменів, або можуть бути змінені в процесі навчання.

Розглянемо приклад, в якому товар описується d атрибутами. Ми хотіли б визначити значення подібності між двома частковими векторами атрибутів, визначеними в підмножині S розмірності d атрибутів (тобто | S | = s ≤ d). Нехай X = (x1...xd) і T = (t1...td) представляють два d-мірних вектори, які можуть бути частково задані. T представляє собою ціль, та передбачається, що принаймні підмножину атрибутів S ⊆ {1,d} вказано в обох векторах. Ми використовуємо часткові вектори атрибутів, оскільки подібні запити часто визначаються лише на невеликому підмножині атрибутів, визначених користувачем. Наприклад, у прикладі пошуку нерухомості користувач може вказати лише невеликий набір функцій запиту, таких як кількість спалень або ванних кімнат. Тоді функція подібності f (T, X) між двома наборами векторів визначається наступним чином:

$$f(T,X) = \frac{\sum_1^j w_j * Sim(t_j,x_j)}{\sum_1^j w_j}, де\ j \in S, \quad (3.1)$$

тут $Sim(t_j, x_j)$ представляє подібність між значеннями $x_j$ та $y_j$. Вага $w_j$ представляє вагу j-го атрибута, і він регулює відносну важливість цього атрибута.

Спочатку визначення функції подібності $Sim(t_j, x_j)$. Ці атрибути можуть бути кількісними або категоричними, що додатково додає неоднорідності та складності такої системи. Крім того, атрибути можуть бути симетричними або асиметричними з точки зору вищих чи нижчих значень [36]. Точний рівень асиметрії може бути різним для різних атрибутів. Наприклад, для такого

атрибута, як роздільна здатність камери, користувач може виявити більш бажаним роздільну здатність, але перевага може бути настільки великою, як у випадку ціни. Інші атрибути можуть бути повністю симетричними, і в цьому випадку користувач хоче, щоб значення атрибута було точно за цільовим значенням $t_j$.

Приклад у випадку симетричної метрики такий:

$$Sim(t_j, x_j) = 1 - \frac{|t_j - x_j|}{max_j - min_j}, \tag{3.2}$$

тут $max_j$ та $min_j$ представляють максимально чи мінімально можливі значення атрибутів.

У випадку симетричної метрики подібність повністю визначається різницею між двома атрибутами. У випадку асиметричного атрибута можна додати додаткову асиметричну винагороду, яка починає діяти залежно від того, менша чи більша цільова атрибутика. Для випадку атрибутів, у яких більші значення кращі, прикладом можливої функції подібності є такий:

$$Sim(t_j, x_j) = 1 - \frac{|t_j - x_j|}{max_j - min_j} + \alpha_j * I(x_j > t_j) * \frac{|t_j - x_j|}{max_j - min_j}, \tag{3.3}$$

тут останній доданок є так званою асиметричною винагородою, $\alpha_j \geq 0$ – це визначений користувачем параметр, а $I(x_j > t_j)$ – це функція індикатора, яка приймає значення 1, якщо $x_j > t_j$, і 0 в іншому випадку. Варто відзначити, що винагорода починається лише тоді, коли значення атрибута $x_j$ (наприклад, роздільна здатність камери) перевищує цільове значення $t_j$. У випадках, коли

менші значення кращі (наприклад, ціна), функція винагороди є подібною, за винятком того, що менші значення винагороджуються функцією індикатора:

$$Sim(t_j, x_j) = 1 - \frac{|t_j - x_j|}{max_j - min_j} + \alpha_j * I(x_j < t_j) * \frac{|t_j - x_j|}{max_j - min_j}. \quad (3.4)$$

Значення αі вибираються дуже специфічно для домену. Для значень $\alpha_j > 1$ «подібність» насправді збільшується із більшою відстанню до цілі. У таких випадках корисно думати про $Sim(t_j, x_j)$ як про функцію корисності, а не як про функцію подібності. Наприклад, у випадку ціни завжди можна віддати перевагу нижчій ціні перед вищою, хоча цільова ціна може визначати точку перегину в тій мірі, з якою віддають перевагу нижчій ціні перед вищою. Коли значення $\alpha_j$ дорівнює рівно 1,0, це означає, що ніхто не дбає про подальші зміни від цільового значення в одному з напрямків. Прикладом може бути випадок з роздільною здатністю камери, коли людина може не дбати про роздільну здатність, що перевищує певну точку. Коли $\alpha_j \in (0, 1)$, це означає, що користувач віддає перевагу значенню в цілі перед усіма іншими значеннями, але вона може мати асиметричні уподобання по обидві сторони цілі. Це свідчить про те, що не існує простих способів попереднього визначення таких метрик подібності; експерт із предметної області повинен виконати багато роботи.

У випадку категоричних даних визначення значень подібності часто є більш складним завданням. Зазвичай для того, щоб визначити значення подібності будуються ієрархії доменів. Два об'єкти, які ближче один до одного в контексті ієрархії доменів можна вважати більш подібними.

Другим питанням у розробці функцій подібності є визначення відносної важливості різних атрибутів. Відносна важливість j-х атрибутів регулюється параметром $w_j$ у рівнянні 3.1. Одна з можливостей полягає в тому, щоб експерт домену вручну кодував значення $w_j$ за допомогою випробування та досвіду.

Інша можливість полягає у вивченні значень $w_j$ із відгуками користувачів. Пари цільових об’єктів можуть бути представлені користувачам, і користувачів можуть попросити оцінити, наскільки подібні цільові об’єкти. Цей зворотний зв'язок може бути використаний разом із моделлю лінійної регресії для визначення значення $w_j$. Користувачеві простіше вказати відносні впорядкування, а не вказувати явні значення подібності для пар об’єктів.

Проблема відсутності різноманітності полягає в тому, що якщо користувачеві не подобається найкращий результат, йому часто не сподобаються інші результати, схожі на нього. Очевидно, що цей сценарій зменшує дійсний вибір, доступний користувачеві серед найкращих рекомендацій.

Розглянемо сценарій, коли бажано отримати найбільш-k результати, що відповідають конкретному випадку. Одна з можливостей – отримати найкращі b * k результати (для b > 1), а потім випадковим чином вибрати k елементів із цього списку. Ця стратегія також називається стратегією обмеженого випадкового відбору. Однак така стратегія, здається, не дуже добре працює на практиці.

Більш ефективним підходом є обмежена жадібна стратегія відбору [37]. У цій стратегії ми починаємо з основних випадків b · k, подібних до цільових, і поступово будуємо різноманітний набір k випадків з цих випадків b · k. Тому ми починаємо з порожнього набору R і поступово будуємо його, додаючи екземпляри з базового набору b · k випадків. Першим кроком є створення якісної метрики, яка поєднує подібність та різноманітність. Припустимо без втрати загальності, що функція подібності f (X, Y) завжди відображається до значення в (0, 1). Тоді розмаїття D (X, Y) можна розглядати як відстань між X та Y:

$$D(X,Y) = 1 - f(X,Y). \qquad (3.5)$$

Тоді середня розбіжність між кандидатом X та набором R обраних випадків визначається як середня розбіжність між X та випадками у R:

$$D^{avg}(X,R) = \frac{\sum_1^j D(X,Y)}{|R|}, де\ Y \in R. \quad (3.6)$$

Тоді для цільової T загальна якість Q(T,X,R) обчислюється наступним чином:

$$Q(T,X,R) = f(T,X) \ * D^{avg}(X,R). \quad (3.7)$$

Прецедент X з найбільшою якістю поступово додається до множини R, поки потужність множини R не дорівнює k. Цей набір надається користувачеві.

## 3.3 Методи критики

Критика мотивована тим, що користувачі часто не в змозі точно вказати свої вимоги у початковому запиті. У деяких складних предметних областях їм навіть може бути важко перевести свої потреби в семантичному вигляді на значення атрибутів у предметній області. Лише після перегляду результатів запиту користувач може зрозуміти, що їй слід було диктувати свій запит дещо інакше. Критика покликана забезпечити користувачам цю здатність фактично. Після того, як результати були представлені користувачам, зворотній зв'язок, як правило, надається за допомогою критики. У багатьох випадках інтерфейси призначені для критики найбільш схожого відповідного елемента, хоча технічно можливо, щоб користувач критикував будь-який з елементів із отриманого списку k елементів. У критиці користувачі вказують запити на зміну одного чи кількох атрибутів елемента, який їм може сподобатися.

Наприклад, у варіанти вибору житла користувачеві може сподобатися певний будинок, але вона може захотіти будинок в іншому населеному пункті або з ще однією спальнею. Тому користувач може вказати зміни в характеристиках одного з предметів, які їй подобаються. Користувач може вказати спрямованість (наприклад, «дешевше») або заміну (наприклад, «інший колір»). У таких випадках усуваються приклади, які не задовольняють задані користувачем зауваження, і отримуються приклади, схожі на вибраний користувачем елемент (але відповідають поточній послідовності застережень). Коли в послідовних циклах рекомендацій зазначено кілька критичних зауважень, перевага віддається більш пізнім критичним вимогам. У конкретний момент користувач може вказати або одну функцію, або комбінацію функцій для модифікації. Таким чином критики бувають трьох різних типів, а саме прості критики, складні критики та динамічні критики.

У простій критиці користувач вказує одну зміну однієї з функцій рекомендованого елемента. Часто в багатьох системах, таких як системи FindMe, використовується інтерфейс, імітуючий розмову, де користувачі вказують, збільшувати чи зменшувати конкретне значення атрибута, а не явно модифікувати одне із цільових значень атрибутів. Це називається спрямованою або направленою критикою. У таких випадках у списку кандидатів просто обрізають ті об’єкти, для яких атрибут, що критикується, знаходиться на неправильній стороні заявлених користувачем переваг. Перевага такого підходу полягає в тому, що користувач може заявити про свої переваги та орієнтуватися у товарному просторі без необхідності точно вказувати або змінювати значення атрибутів. Такий підхід особливо важливий у предметних областях, де користувачі можуть не знати точного значення атрибута, який слід використовувати (наприклад, кінських сил двигуна). Ще однією перевагою спрямованої критики є те, що вона має простий розмовний стиль, який може бути більш інтуїтивним та привабливим для користувача. У випадках, коли користувач взагалі не знаходить поточного набору отриманих

результатів корисним, вона також має можливість повернутися до точки входу. Це являє собою безрезультатний цикл процесу критики.

Основна проблема простого підходу до критики – це важка навігація. Якщо рекомендований продукт містить багато функцій, які потрібно змінити, це призведе до довшого ланцюга подальших запитів. Крім того, при зміні однієї з функцій системі, що рекомендує, може автоматично знадобитися змінити принаймні деякі інші значення функцій залежно від наявності товару. У більшості випадків неможливо утримувати інші значення ознак за точно постійними значеннями в даному циклі. Як результат, коли користувач змінив кілька функцій до бажаних значень, він може зрозуміти, що інші значення функції більш не є прийнятними. Чим більша кількість циклів рекомендацій, тим менше контролю буде мати користувач над змінами в інших значеннях функцій, які були задовільними в попередніх ітераціях. Ця проблема часто виникає внаслідок нерозуміння користувачем компромісів у предметній області. Наприклад, користувач може не зрозуміти компромісу між кінськими силами та паливною ефективністю і спробувати перейти до автомобіля з високою кінською потужністю, а також з високою економічністю пального в 50 км на літр. Основна проблема багатьох інтерфейсів полягає в тому, що наступний набір рекомендованих елементів базується на останніх критикованих елементах, і немає можливості повернутися до попередніх елементів. Як наслідок, довгий цикл простих запитів іноді може призвести до неправильного результату.

Складена критика були розроблені для зменшення тривалості циклів рекомендацій [38]. У цьому випадку користувач може вказати кілька модифікацій функцій за один цикл. Наприклад, якщо користувач при виборі житла вибирає «просторіше», це означає, що, можливо, доведеться збільшити як кількість спалень, так і кількість ванних кімнат. Що стосується другого типу інтерфейсу, експерт домену повинен витратити значні зусилля на розробку відповідного інтерфейсу та інтерпретацію вибору користувача з точки зору

змін, внесених до багатьох функцій продукту. Це кодування є статичним і виконується наперед. Головною перевагою складеної критики є те, що користувач може змінити кілька функцій у цільовій рекомендації, щоб подати новий запит або обрізати результати пошуку з попереднього запиту. Як результат, такий підхід дозволяє здійснювати значні стрибки через простір продукту, і користувач часто має кращий контроль над процесом критики. Це корисно для зменшення кількості рекомендаційних циклів та підвищення ефективності процесу розвідки. Однак незрозуміло, чи завжди складні зауваження допомагають користувачеві засвоїти товарний простір краще, ніж прості зауваження; короткі цикли критики також зменшують ймовірність того, що користувач вивчить різні компроміси та кореляцію між особливостями товарного простору. З іншого боку, користувач іноді може дізнатися багато нового про товарний простір, проходячи повільний і важкий процес простої критики.

Незважаючи на те, що складена критика дозволяє більші стрибки через навігаційний простір, вона має недолік у тому, що варіанти критики, представлені користувачеві, є статичними в тому сенсі, що вони не залежать від отриманих результатів. Наприклад, якщо користувач переглядає автомобілі, і вона вже переглядає найдорожчий автомобіль з найбільшою можливою кінською силою, опція збільшення кінських сил і ціни все одно буде відображатися в інтерфейсі для критики. Очевидно, що зазначення цих варіантів призведе до безрезультатного пошуку. Це пояснюється тим, що користувачі часто не до кінця усвідомлюють властиві компроміси у складному товарному просторі. При динамічній критиці мета полягає у використанні аналізу даних на отриманих результатах для визначення найбільш плідних шляхів дослідження та представлення їх користувачеві. Таким чином, динамічна критика за визначенням є складною критикою, оскільки вона майже завжди представляє комбінації змін, представлених користувачеві. Основна відмінність полягає в тому, що на основі отриманих на даний момент

результатів представлено лише підмножину найбільш відповідних можливостей. Отже, динамічна критика розроблена, щоб забезпечити кращі вказівки користувачеві під час процесу пошуку. Важливим аспектом динамічної критики є можливість виявляти часті комбінації змін характеристик товару. Поняття підтримки адаптоване на основі частого створення шаблонів [39]. для того, щоб визначити закономірності часто зустрічаються особливостей продукту в отриманих результатах. Підтримка шаблону визначається як частка отриманих результатів, які задовольняють цьому шаблону. Суперечливі варіанти мають менший шанс бути включеними, оскільки вони можуть бути усунені на основі мінімального критерію підтримки. Насправді, як тільки всі шаблони, що відповідають мінімальному порогу підтримки, були визначені, багато рекомендаційних систем впорядковують критики користувачеві у порядку зростання підтримки. Логіка такого підходу полягає в тому, що низькі критичні показники підтримки часто є менш очевидними закономірностями, які можна використовувати для виключення більшої кількості елементів зі списку кандидатів. Реальним прикладом динамічного підходу до критики, який використовує часте видобуток шаблонів та правил асоціацій, є система Qwikishop. Важливим зауваженням щодо систем динамічної критики є те, що вони збільшують когнітивне навантаження на користувача при перегляді на основі циклу, але зменшують когнітивне навантаження протягом усього сеансу через їх здатність швидше отримувати вірні рекомендації. Це є однією з причин того, що ефективний дизайн пояснювальних процесів у циклі критики є більш важливим у системах із динамічною критикою.

Основною небезпекою в системах, що базуються на критиці, є ймовірність того, що користувачі безцільно блукатимуть у просторі знань без успішного знаходження того, що вони шукають. Додавання пояснень до інтерфейсу значно зменшує цю ймовірність.

## 3.4 Метод, заснований на обмеженнях

Системи рекомендацій на основі обмежень дозволяють користувачам задавати жорсткі вимоги або обмеження щодо атрибутів товару. Крім того, використовується набір правил для узгодження вимог користувача з атрибутами товару. Однак користувачі не завжди можуть вказувати свої запити з точки зору тих самих атрибутів, що описують товари. Отже, необхідний додатковий набір правил, який пов’язує вимоги користувача з атрибутами товару. У контексті вибору житла можна навести наступні приклади атрибутів, визначених користувачем: сімейний стан (категоричний), розмір сім'ї (чисельний), пригород чи місто (булевий), мінімальна кількість спалень (чисельний), максимальна кількість спалень (чисельний), максимальна ціна (чисельний). Ці атрибути можуть представляти або власні атрибути користувача (наприклад, демографічні показники), або вони можуть визначати вимоги замовника до товару. Такі вимоги зазвичай визначаються інтерактивно під час діалогу між замовником та рекомендаційною системою.

Хоча відображення деяких атрибутів вимог замовника, таких як максимальна ціна, на атрибути товару є очевидними, відображення інших, таких як категоричні атрибути, не є настільки очевидними. Подібним чином, у фінансовій заявці замовник може вказати вимогу до товару, таку як «консервативні інвестиції», яку потрібно зіставити з конкретними атрибутами товару що безпосередньо описує товари. Очевидно, що якимось чином слід мати можливість відобразити ці атрибути чи вимоги замовника в атрибутах товару, щоб фільтрувати товари для рекомендацій. Це досягається використанням баз знань. Бази знань містять додаткові правила, що відображають атрибути / вимоги замовника до атрибутів товару у форматі «Якщо атрибут приймає значення, то інший атрибут приймає одне або кілька значень», наприклад:

$$\text{Місто} - \text{чи} - \text{село} = \text{Місто} => \text{Місцезнаходження} = \\ < \text{Набір доступних місць} >. \quad (3.8)$$

Такі правила називаються умовами фільтрування, оскільки вони відображають вимоги користувача до атрибутів елемента та використовують це відображення для фільтрування отриманих результатів. Такі типи правил можуть бути або похідними від предметної продукту, або, що рідше, вони можуть бути отримані шляхом історичного аналізу таких наборів даних. У цьому конкретному випадку очевидно, що це правило можна отримати безпосередньо, використовуючи загальнодоступну географічну інформацію. Інший приклад – предметна область автомобіля, де одні певні атрибути и можуть бути дійсними лише з певними іншими атрибутами. Наприклад, двигун з високим крутним моментом може бути доступний лише у спортивній моделі. Такі умови також називаються умовами сумісності, оскільки вони можуть бути використані для швидкого виявлення невідповідності вимог, заданих користувачем, домену продукту. У багатьох випадках такі обмеження сумісності можуть бути інтегровані в інтерфейс користувача. В інших випадках, коли виявлення невідповідностей неможливе в інтерфейсі користувача, такі невідповідності можна виявити під час обробки запиту, у випадку повернення порожнього набору результатів. Деякі інші обмеження сумісності можуть пов'язувати атрибути клієнта між собою. Такі обмеження корисні, коли клієнти вказують про себе особисту інформацію (наприклад, демографічну інформацію) під час інтерактивної сесії. Наприклад, демографічні атрибути можуть бути пов’язані з вимогами до продукту клієнта на основі обмежень для конкретного домену або історичного досвіду. Прикладом такого може бути «Сімейний стан = неодружений => Мінімальна кількість спалень < 3», тощо.

Імовірно, на основі досвіду, що стосується конкретної предметної області, або шляхом аналізу даних історичних наборів даних, було зроблено

висновок, що одинокі особи не вважають за краще купувати дуже великі будинки. Так само маленький будинок може не підійти для дуже великої родини. Це обмеження моделюється за наступним правилом: «Розмір сім'ї < 5 => Мінімальна кількість спалень >= 3».

Таким чином, існує три основних типи вхідних даних до системи рекомендацій на основі обмежень. Перший клас вхідних даних представлений атрибутами, що описують властиві користувачеві властивості (наприклад, демографічні показники, профілі ризику) та конкретні вимоги до продукту (наприклад, Міні-спальні). Деякі з цих атрибутів легко пов’язати з атрибутами товару, тоді як інші можуть бути пов’язані з атрибутами товару лише завдяки використанню баз знань. У більшості випадків властивості та вимоги замовника вказуються інтерактивно під час сеансу, і вони не зберігаються протягом декількох сеансів. Отже, якщо інший користувач задає той самий набір вимог у сеансі, він отримає той самий результат. Це відрізняється від інших типів систем рекомендацій, у яких персоналізація є постійною, оскільки базується на історичних даних.

Другий клас вхідних даних представлений базами знань, які відображають атрибути чи вимоги замовника до різних атрибутів товару. Її створення може бути здійснено як прямо, так і опосередковано.

Прямі правила пов'язують вимоги замовника з жорсткими вимогами щодо атрибутів товару, наприклад залежність ціни від кількості кімнат. Такі правила також називаються умовами фільтрування.

Опосередковані правила пов'язують атрибути чи вимоги замовника із типово очікуваними вимогами до товару. Отже, такі правила також можна розглядати як непрямий спосіб пов’язування атрибутів клієнта з атрибутами товару. Наприклад зв’язок розміру сім'ї із кількістю спалень. Умови з обох сторін правила представляють атрибути замовника, хоча умови з правого боку, як правило, є вимогами замовника, які можна легко зіставити з атрибутами товару. Ці обмеження представляють обмеження сумісності. У випадку, якщо

обмеження сумісності або умови фільтра несумісні із вимогами, визначеними замовником, рекомендований список елементів буде порожнім. Вищезазначені бази знань походять із загальнодоступної інформації, експертів предметних областей, минулого досвіду або аналізу даних історичних наборів даних. Тому значна кількість зусиль залучається до побудови баз знань.

Нарешті, каталог товарів містить перелік усіх товарів разом із відповідними атрибутами товару. Тому проблема зводиться до визначення всіх випадків у наявному переліку товарів, які відповідають вимогам замовника та правилам у базі знань. Найпростіший підхід ранжування предметів відповідно до вимог користувача – дозволити користувачеві вказати один числовий атрибут, на основі якого класифікувати елементи. Наприклад, у програмі для придбання житла система може надати користувачеві можливість ранжувати товари на основі (будь-якої з) ціни будинку, кількості спалень або відстані від певного поштового індексу. Такий підхід фактично використовується у багатьох комерційних інтерфейсах. У багатьох випадках певний запит може повернути порожній набір результатів. В інших випадках набір повернутих результатів може бути недостатньо великим, щоб відповідати вимогам користувача. У таких випадках користувач має два варіанти. Якщо вважається, що очевидного способу зміни обмежень не існує, користувач може вирішити почати рекомендацію спочатку. Як варіант, він може вирішити змінити або послабити обмеження для наступної інтерактивної ітерації.

У деяких випадках кількість повернених результатів може бути дуже великою, і користувачеві може знадобитися запропонувати можливі обмеження, які слід додати до запиту. У таких випадках можна використовувати різноманітні методи, щоб запропонувати користувачеві обмеження разом із можливими значеннями за замовчуванням. Атрибути таких обмежень часто обираються, засновуючись на історії сеансів. Історія сеансів може бути визначена як для всіх користувачів, так і для конкретного

користувача. Останнє забезпечує більш персоналізовані результати, але часто може бути недоступним для рідко купляємих предметів (наприклад, автомобілів чи будинків). Примітно, що системи, засновані на знаннях, як правило, розроблені для того, щоб не використовувати таку постійну та історичну інформацію саме тому, що вони призначені для роботи в умовах холодного запуску; проте така інформація часто може бути дуже корисною для покращення взаємодії з користувачем. Ідея використання історії сеансів полягає у виборі популярних обмежень. Наприклад, якщо користувач вказав обмеження для набору атрибутів елемента, то ідентифікуються інші сеанси, що містять один або кілька з цих атрибутів. Наприклад, якщо користувач вказав обмеження на кількість спалень та ціну, ідентифікуються попередні сеанси, що містять обмеження на спальню та ціну. Зокрема, визначено найближчі сесії найближчих сусідів за кількістю загальних атрибутів. Якщо визначено, що найбільш популярним обмеженням серед цих найбільш-k сесій є кількість ванних кімнат, тоді цей атрибут пропонується інтерфейсом як кандидат для додавання додаткових обмежень.

У багатьох випадках доступне тимчасове впорядкування, за якого користувачі раніше вказували обмеження. У таких випадках також можна використовувати порядок, у якому замовник вказав обмеження, розглядаючи обмеження як замовлений набір, а не як невпорядкований набір [40]. Простий спосіб досягнення цієї мети полягає у визначенні найбільш частого атрибута, який слідує поточному заданому набору обмежених атрибутів у попередніх сесіях. Для визначення таких частих атрибутів можна використовувати послідовний аналіз шаблонів (sequential pattern mining). Обмеження можна запропонувати на основі їх вибірковості в базі даних або на основі середньої інформації щодо користувача за минулі сеанси.

# 4 РОЗРОБКА ПРОТОТИПУ СИСТЕМИ

## 4.1 Огляд використаного середовища та набору даних

Для проектування використовувалося середовище Jupyter Notebook, система написана на мові програмування Python. Jupyter Notebook – це потужний інструмент для розробки та подання проектів Data Science в інтерактивному вигляді. Він об'єднує код і висновок в одному документі, що містить текст, математичні рівняння і візуалізації. Такий покроковий підхід забезпечує швидкий, послідовний процес розробки, оскільки висновок для кожного блоку показується відразу ж [41].

У проекті використовувалися бібліотеки numpy, pandas, ast.

NumPy – розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. Numpy це відкрите програмне забезпечення і має багато розробників. Оскільки Python інтерпретована мова, математичні алгоритми, часто працюють в ньому набагато повільніше ніж у компільованих мовах, таких як C або навіть Java. NumPy намагається вирішити цю проблему для великої кількості обчислювальних алгоритмів забезпечуючи підтримку багатовимірних масивів і безліч функцій і операторів для роботи з ними. Таким чином будь-який алгоритм, який може бути виражений в основному як послідовність операцій над масивами і матрицями, працює так само швидко, як еквівалентний код, написаний на C [42].

Pandas – програмна бібліотека, написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для маніпулювання чисельними таблицями та часовими рядами. Вона надає такий функціонал як об'єкт DataFrame із вбудованим індексуванням для маніпулювання даними, інструменти для

зчитування та записування даних між структурами даних у пам'яті та різними форматами файлів. Вирівнювання даних та вбудована підтримка пропущених даних. Переформатування для отримання зведених наборів даних. Отримання зрізів за мітками, індексування з розширеними можливостями та отримання піднаборів з великих наборів даних. Вставлення та вилучення стовпчиків у структурах даних. Рушій групування, що дозволяє робити з наборами даних операції розділення-зміни-об'єднання (split-apply-combine). Злиття та з'єднання наборів даних. Ієрархічне індексування осей для роботи з даними високої вимірності в структурі даних нижчої вимірності. Функціональність для часових рядів: породження діапазонів дат та перетворення частоти, статистики рухливого вікна, лінійні регресії рухливого вікна, зсування дат та запізнювання [43].

Тестування системи на повному наборі даних проводилося у середовищі Google Colab. Google Colab – це безкоштовний хмарний сервіс на основі Jupyter Notebook. Google Colab надає все необхідне для машинного навчання прямо в браузері, дає безкоштовний доступ до неймовірно швидких процессорів.

Використаний набір даних складається із 231637 рецептів, узятих із сайту Food.com (раніше GeniusKitchen). Дані представлені у форматі csv (рисунок 4.1 та рисунок 4.2) та містить наступні атрибути:

– name – назва рецепту, строковий;

– id – ідентифікатор рецепту, чисельний;

– minutes – приблизна кількість хвилин на приготування рецепту, чисельний;

– contributor_id – ідентифікатор користувача, що запропонував рецепт, чисельний;

– submitted – дата, коли рецепт було додано, дата;

– tags – теги Food.com, строкове представлення списку строк;

– nutrition – поживна цінність рецепту( калорії (кількість), загальний

жир – (Percent Daily Value – PDV, Відсоток щоденної вартості), цукор (PDV), натрій (PDV), білок (PDV), насичені жири), строкове представлення списку чисельних атрибутів.

– n_steps – кількість кроків у рецепті, чисельний;

– steps – опис кроків приготування рецепту, строкове представлення списку строк;

– description – опис рецепту від користувача, строкове представлення списку строк;

– ingredients – інгредієнти, що входять до рецепту, строкове представлення списку строк.

```
df1 = pd.read_csv(io.StringIO(uploaded['RAW_recipes.csv'].decode('utf-8')))
print(df1.dtypes)
df1.head()
```

```
name              object
id                 int64
minutes            int64
contributor_id     int64
submitted         object
tags              object
nutrition         object
n_steps            int64
steps             object
description       object
ingredients       object
n_ingredients      int64
dtype: object
```

Рисунок 4.1 – Опис колонок

| name | id | minutes | contributor_id | submitted | tags | nutrition | n_steps | steps | description | ingredients | n_ingredients |
|---|---|---|---|---|---|---|---|---|---|---|---|
| arriba baked winter squash mexican style | 137739 | 55 | 47892 | 2005-09-16 | ['60-minutes-or-less', 'time-to-make', 'course... | [51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0] | 11 | ['make a choice and proceed with recipe', 'dep... | autumn is my favorite time of year to cook! th... | ['winter squash', 'mexican seasoning', 'mixed ... | 7 |
| a bit different breakfast pizza | 31490 | 30 | 26278 | 2002-06-17 | ['30-minutes-or-less', 'time-to-make', 'course... | [173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0] | 9 | ['preheat oven to 425 degrees f', 'press dough... | this recipe calls for the crust to be prebaked... | ['prepared pizza crust', 'sausage patty', 'egg... | 6 |
| all in the kitchen chili | 112140 | 130 | 196586 | 2005-02-25 | ['time-to-make', 'course', 'preparation', 'mai... | [269.8, 22.0, 32.0, 48.0, 39.0, 27.0, 5.0] | 6 | ['brown ground beef in large pot', 'add choppe... | this modified version of 'mom's' chili was a h... | ['ground beef', 'yellow onions', 'diced tomato... | 13 |
| alouette potatoes | 59389 | 45 | 68585 | 2003-04-14 | ['60-minutes-or-less', 'time-to-make', 'course... | [368.1, 17.0, 10.0, 2.0, 14.0, 8.0, 20.0] | 11 | ['place potatoes in a large pot of lightly sal... | this is a super easy, great tasting, make ahea... | ['spreadable cheese with garlic and herbs', 'n... | 11 |
| amish tomato ketchup for canning | 44061 | 190 | 41706 | 2002-10-25 | ['weeknight', 'time-to-make', 'course', 'main-... | [352.9, 1.0, 337.0, 23.0, 3.0, 0.0, 28.0] | 5 | ['mix all ingredients& boil for 2 1 / 2 hours ... | my dh's amish mother raised him on this recipe... | ['tomato juice', 'apple cider vinegar', 'sugar... | 8 |

Рисунок 4.2 – Представлення датасету

Перед реалізацією рекомендацій необхідно провести передобробку даних, позбавившись зайвої інформації та виділивши нову, виходячи із аналізу набору даних.

## 4.2 Передобробка даних

Першим кроком стане видалення із датасету зайвої інформації (рисунок 4.3). Індекс рецепту, користувача що додав рецепт та дата додання нам не потрібні, тому від відповідних колонок можна позбавитись за допомогою методу Dataframe.drop(). Далі колонку nutrition було перетворено на 7 відповідних колонок, які містимуть ті ж дані з колонки, але індексовані та більш зрозумілі. Вони містять калорії, значення жирів, білку, цукру, натрію та насичених жирів (рисунок 4.4). Після цього видаляється і колонка nutrition, так як всі дані з неї було перетворено.

У нових колонках будуть міститися дані, які до цього зберігалися у вигляді масиву у колонці тегів. Але такий формат даних доволі складно обробляти, тому за можливістю необхідно перетворити ці дані у булевий чи категоричний формат. Це набагато пришвидшить роботу системи із даними при фільтрації та рекомендації.

Повний програмний код наведено в додатку А.

```
df1[['calories','total fat (PDV)','sugar (PDV)','sodium (PDV)','protein (PDV)','saturated fat (PDV)','carbohydrates (PDV)']]
df1['calories'] =  df1['calories'].apply(lambda x: x.replace('[',''))
df1['carbohydrates (PDV)'] =  df1['carbohydrates (PDV)'].apply(lambda x: x.replace(']',''))
df1[['calories','total fat (PDV)','sugar (PDV)','sodium (PDV)','protein (PDV)','saturated fat (PDV)','carbohydrates (PDV)']]
df1.drop(['id', 'contributor_id','submitted','nutrition'], axis=1,inplace = True)
df1.head()
```

Рисунок 4.3 – Видалення зайвих колонок та перетворення колонки nutrition

| calories | total fat (PDV) | sugar (PDV) | sodium (PDV) | protein (PDV) | saturated fat (PDV) | carbohydrates (PDV) |
|---|---|---|---|---|---|---|
| 51.5 | 0.0 | 13.0 | 0.0 | 2.0 | 0.0 | 4.0 |
| 173.4 | 18.0 | 0.0 | 17.0 | 22.0 | 35.0 | 1.0 |

Рисунок 4.4 – Створені колонки

Подальші перетворення здійснюються завдяки аналізу колонки tags. Вона має теги, що надані самим сайтом Food.com та мстять багато необхідних даних для подальшого виділення атрибутів для фільтрації рекомендацій. Але ситуація ускладнюється через те, що колонка також містить інформацію про інгредієнти, тобто частково дублює колонку ingredients. Щоб вирішити цю проблему усі теги було фільтровано. Для цього було створено отримано список унікальних інгредієнтів та унікальних тегів. У результаті цього кількість унікальних тегів була зменшена із 17 тисяч до 344, що представлено на рисунку 4.5.

```
ingredients = [];
for x in df.ingredients:
  for el in ast.literal_eval(x):
    if el not in ingredients:
      ingredients.append(el)
```

```
ing1 = [];
for el in ingredients:
  el_arr = el.split(' ');
  for y in el_arr:
    ing1.append(y)
```

```
unique_ing = list(set(ing1))
```

Рисунок 4.5 – Фільтрація тегів рецепту

Далі на основі тегів було створено ще 10 колонок. Це Lactose, Nuts, Easy, Dietary, Diabetic. Kids, Dish, Quisine, Groups, Vegan. Частина колонок відображала належність певного тегу рецепту та є булевою, інша ж відображала конкретний тег із певної групи, тобто категоричне значення. Приклад створення обох типів колонок та результат представлено на рисунках 4.6 та 4.7.

| Lactose | Easy | Nuts | Dietary | Diabetic | Kids | Dish | Quisine | Groups | Vegan |
|---|---|---|---|---|---|---|---|---|---|
| False | True | True | True | False | False | brunch | NaN | False | False |

Рисунок 4.6 – Додані колонки

```
df['Kids'] = np.nan
df['Kids'] = df['Kids'].astype('bool')
kids_tags = ['kid-friendly', 'toddler-friendly',  'infant-baby-friendly', 'snacks-kid-friendly']
for i in df['tags'].index:
  kids_count = 0;
  for j in range(0, len(kids_tags)):
    if kids_tags[j] in df['tags'][i]:
      kids_count = kids_count + 1;
      df['Kids'][i]=True
  if kids_count == 0:
    df['Kids'][i]=False
```

```
df['Dish'] = np.nan
dish_tags = []
arr = ['main-dish','side-dish','salad','soup','dessert','brunch', 'lunch', 'quick-breads','garnish']
for tag in final_tags:
  for x in arr:
    if x in tag:
      dish_tags.append(tag)
dish_dtype = pd.api.types.CategoricalDtype(
    categories=dish_tags)
df['Dish'] = df['Dish'].astype(dish_dtype)
for i in df['tags'].index:
  for j in range(0, len(dish_tags)):
    if dish_tags[j] in df['tags'][i]:
      df['Dish'][i]=dish_tags[j]
```

Рисунок 4.7 – Створення колонок двох типів

Отож, результуючий набір даних можна описати такими атрибутами:

– name – назва рецепту, строковий;

– minutes – приблизна кількість хвилин на приготування рецепту, чисельний;

– tags – теги Food.com, строкове представлення списку строк;

– nutrition – поживна цінність рецепту (калорії (кількість), загальний жир (Percent Daily Value – PDV, Відсоток щоденної вартості), цукор (PDV), натрій (PDV), білок (PDV), насичені жири, вуглеводи), строкове представлення списку чисельних атрибутів;

– n_steps – кількість кроків у рецепті, чисельний;

– steps – опис кроків приготування рецепту, строкове представлення списку строк;

– description – опис рецепту від користувача, строкове представлення списку строк;

– ingredients – інгредієнти, що входять до рецепту, строкове представлення списку строк;

– calories – кількість калорій у рецепті, чисельний;

– total fat (PDV) – загальний жир, чисельний;

– sugar (PDV) – цукор у рецепті, чисельний;

– sodium (PDV) – натрій у рецепті, чисельний;

– protein (PDV) – білок у рецепті, чисельний;

– saturated fat (PDV) – насичені жири у рецепті, чисельний;

– carbohydrates (PDV) – вуглеводи у рецепті, чисельний;

– Lactose – наявність лактози у рецепті, булевий;

– Easy – чи є рецепт легким, булевий;

– Nuts – наявність горіхів в рецепті, булевий;

– Dietary – чи є рецепт дієтичним, булевий;

– Diabetic – чи є рецепт діабетичним, булевий;

– Kids – чи можна страву вживати дітям, булевий;

– Dish – тип страви, до якої відноситься рецепт, категоричний (перша страва, гарнір, салат, десерт, суп, друга страва, тощо);

– Quisine – кухня, до якої відноситься рецепт, категоричний (57 видів кухні, як конкретних країн так і загальних, наприклад азіатська);

– Groups – розмірність страви, чи підходить для групи людей більше двох, булевий;

– Vegan – чи є рецепт вегетаріанським, булевий.

## 4.3 Реалізація методів рекомендації

Метод рекомендацій, заснований на обмеженнях, що використано в роботі, включає в себе обмеження двох типів – жорсткі та м’які рекомендації. На практиці це означає, що жорсткими обмеженнями система не може поступитися у разі недоліку рекомендацій, а м’які впливатимуть лише на приорітет усіх рекомендацій, що доступні після застосування жорстких обмежень.

Жорсткі обмеження представляють собою обмеження, що стосуються користувача системи. Вони дійсні увесь час роботи із системою і включають в себе:

– вік користувача, булевий;

– список інгредієнтів, що не споживається користувачем через проблеми здоров'я чи інші причини (наприклад, релігійні);

– хвороби / алергії користувача, категоричний (строковий список);

– допустима кількість калорій, чисельний;

– наявність дієти, булевий;

– продвинута дієта, список чисельних значень для жирів, натрію, цукру, насичених жирів, натрію, вуглеводів відповідно;

– кулінарні навички користувача, категоричний (початковий, середній, високий).

М’які ж обмеження відносяться до самих рецептів. Вони можуть змінюватися із кожним новим певного користувача. Вони включають у себе:

– складність рецепту, категоричний (легка, середня, важка);

– допустима кількість калорій, чисельна;

– дієта, список чисельних значень для жирів, натрію, цукру, насичених жирів, натрію, вуглеводів відповідно;

– час на приготування у хвилинах, чисельна;

– кількість людей на яких готується рецепт більше двох, булевий;

– кухня, до якої відноситься рецепт, категорична;

– тип страви, до якої відноситься рецепт, категорична;

– інгредієнти, що мають міститися у рецепті, строковий список;

– інші особливості страви (що не мають власної категорії, наприклад святкова страва), строковий список.

На основі цих обмежень і було створено базу знань, що перетворює вхідні параметри системи, задані користувачем на ті, що використовуються у предметній області для рекомендації рецептів.

Список правил у базі знань включає в себе такі 32 правила:

– «Якщо вік користувача менше 6, повертати ті рецепти, в яких атрибут Kids є дійсним.»;

– «Якщо в користувача не споживає певні інгредієнти, повертати ті рецепти, що не містять жодного інгредієнта зі списку»;

– «Якщо в користувача діабет, повертати ті рецепти, в яких атрибут Diabetic є дійсним»;

– «Якщо в користувача лактозна непереносимість, повертати ті рецепти, в яких атрибут Lactose є недійсним»;

– «Якщо в користувача алергія на арахіс, повертати ті рецепти, в яких

атрибут Nuts є недійсним»;

– «Якщо користувач є вегетаріанцем, повертати ті рецепти, в яких атрибут Vegan є дійсним»;

– «Якщо в користувача є ліміт на калорії, повертати ті рецепти, в яких кількість калорій менше або дорівнює цьому ліміту»;

– «Якщо користувач на дієті, повертати ці рецепти, в яких атрибут Dietary є дійсним»;

– «Якщо користувач вказав необхідний ліміт загальних жирів у дієті, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо користувач вказав необхідний ліміт білків у дієті, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо користувач вказав необхідний ліміт цукрів у дієті, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо користувач вказав необхідний ліміт насичених жирів у дієті, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо користувач вказав необхідний ліміт натрію у дієті, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо користувач вказав необхідний ліміт вуглеводів у дієті, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо навички користувача є початковими, повертати ті рецепти, в яких атрибут Easy є дійсним»;

– «Якщо складність рецепта є легкою, повертати ті рецепти, в яких кількість кроків для приготування є менше чи дорівнює 8»;

– «Якщо складність рецепта є середньою, повертати ті рецепти, в яких кількість кроків для приготування більше 8 та менше чи дорівнює 15 (медіана по всім рецептам)»

– «Якщо складність рецепта є легкою, повертати ті рецепти, в яких кількість кроків для приготування є більше 15»;

– «Якщо вказано калорійність страви, повертати ті рецепти, в яких кількість калорій менше або дорівнює цьому ліміту»;

– «Якщо вказано необхідний ліміт загальних жирів у страві, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо вказано необхідний ліміт білків у страві, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо вказано необхідний ліміт цукру у страві, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо вказано необхідний ліміт натрію у страві, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо вказано необхідний ліміт насичених жирів у страві, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо вказано необхідний ліміт вуглеводів у страві, повертати ті рецепти, в яких їх кількість менше або дорівнює ліміту»;

– «Якщо вказано час на приготування страви, повертати ті рецепти, в яких час менше або дорівнює вказаному»;

– «Якщо кількість людей, на яких розрахована страва більше двох, повертати ті рецепти, в яких параметр Groups дійсний»;

– «Якщо вказано кухню страви, повертати ті рецепти, в яких параметр Quisine відповідає вказаному»;

– «Якщо вказано тип страви, повертати ті рецепти, в яких параметр Dish відповідає вказаному»;

– «Якщо вказано інгредієнти, що мають міститися у страві, повертати ті рецепти, в яких містяться всі задані інгредієнти»;

– «Якщо вказано інгредієнти, що не мають міститися у страві, повертати ті рецепти, в яких не містяться всі задані інгредієнти»;

– «Якщо вказано додаткові параметри страви, повертати ті рецепти, у яких у тегах містяться усі задані параметри».

Засновуючись на цьому було створено функцію recommend, що приймає три аргумента – user (містить жорсткі обмеження щодо користувача), recipe (містить м'які обмеження щодо рецепта) і target (цільовий рецепт, використовується при рекомендаціях на основі прецедентів, що буде розглянуто далі).

Спочатку усі доступні рецепти фільтруються за тими параметрами, що задані в аргументі user завдяки базі знань, що перетворює вхідні параметри на ті, за якими відбувається фільтрація. Якщо після цього кількість рецептів для рекомендацій буде малою, система відобразить повідомлення про те, що рекомендацій замало, і рекомендується змінити обмеження користувача.

Потім ті рецепти, що залишилися після жорстких обмежень проходять фільтрацію за параметрами аргумента recipe завдяки базі знань. Якщо після якогось із кроків фільтрації кількість рецептів для рекомендацій буде малою, система відобразить повідомлення про те, що рекомендацій замало, і рекомендується змінити той параметр, який призвів до цього. Код частини алгоритму рекомендацій та бази знань наведено на рисунку 4.8.

```
def recommend(user, recipe, target):
  global warning;
  if target == False:
    global result;
    print('Initial', len(result));
    if user['age'] <= 6:
      result = result[result['Kids']]
      print('age!', len(result));
    if len(user['allergies']) != 0:
      allerg = pd.DataFrame({'ingredients': user['allergies']});
      allerg_indices = [];
      for z in range(0,len(result)):
        b = any(allerg.isin(ast.literal_eval(result['ingredients'][z]))['ingredients'])
        if b:
          allerg_indices.append(z)
      result = result[result.index.isin(allerg_indices)];
      print('allergies!', len(result))
    if user['calories']:
      result = result[result['calories']<=user['calories']]
      print('calories!', len(result))
```

Рисунок 4.8 – Частина алгоритму рекомендації / бази знань

Після цього було створено функцію подібності, що реалізує рекомендації аналогічних рецептів. Для цього вона приймає у якості параметра цільовий рецепт, і після цього підраховує подібність за атрибутами рецепту відносно до цільового.

Частину коду функції подібності зображено на рисунку 4.9.

```
def countSimilarity(df, target_id):
  global result;
  result['Similarity'] = np.nan;
  targ = df.loc[[target_id]];
  for i in df.index:
    steps_s = 1 - (abs(df.n_steps[i] - targ.n_steps) / (max(df.n_steps) - min(df
    minutes_s = 1 - (abs(df.minutes[i] - targ.minutes) / (max(df.minutes) - min(
    ningr_s = 1 - (abs(df.n_ingredients[i] - targ.n_ingredients) / (max(df.n_ing
    calories_s = 1 - (abs(df.calories[i] - targ.calories) / (max(df.calories) -
    tfat_s = 1 - (abs(df['total fat (PDV)'][i] - targ['total fat (PDV)']) / (max
    sugar_s = 1 - (abs(df['sugar (PDV)'][i] - targ['sugar (PDV)']) / (max(df['su
    sodium_s = 1 - (abs(df['sodium (PDV)'][i] - targ['sodium (PDV)']) / (max(df[
    protein_s = 1 - (abs(df['protein (PDV)'][i] - targ['protein (PDV)']) / (max(
    sfat_s = 1 - (abs(df['saturated fat (PDV)'][i] - targ['saturated fat (PDV)']
    carbs_s = 1 - (abs(df['carbohydrates (PDV)'][i] - targ['carbohydrates (PDV)'
    dish_s = 1 if pd.isna(df.Dish[i]) and pd.isna(targ.iloc[0].Dish) or df.Dish[
    quisine_s = 1 if pd.isna(df.Quisine[i]) and pd.isna(targ.iloc[0].Quisine) or
    group_s = 1 if  pd.isna(df.Groups[i]) and pd.isna(targ.iloc[0].Groups) or df
    tags_df = pd.DataFrame({'tags': ast.literal_eval(targ.iloc[0].tags)});
    z = tags_df.isin(ast.literal_eval(df.tags[i]));
    tags_s = len(z[z.tags==True])/len(z);
    ingr_df = pd.DataFrame({'ingredients': ast.literal_eval(targ.iloc[0].ingredi
    z = ingr_df.isin(ast.literal_eval(df.ingredients[i]));
    ingr_s = len(z[z.ingredients==True])/len(z);


    sim = 0.3*steps_s[[target_id]] + 0.2*minutes_s[[target_id]] + 0.2*ningr_s[[t


    result['Similarity'][i] = sim/5.6;
```

Рисунок 4.9 – Функція подібності

Коефіцієнти у функції подібності мають наступний приорітет:

– інгредієнти рецепту;
– теги;
– калорійність;
– харчова цінність;
– тип страви;

– кухня;

– складність страви;

– час приготування.

Якщо один чи кілька атрибутів не вказані у запиті, то вони не приймають участі у підрахуванні коефіцієнта подібності.

## 4.4 Демонстрація роботи системи

Для демонстрації роботи системи було задано наступні параметри:

– вік користувача дорівнює 18 рокам;

– користувач має алергію на яйця та не споживає свинину через релігійні причини;

– користувач не хоче споживати більше 1500 калорій за раз;

– користувач на дієті;

– складність страви середня;

– допустима калорійність страви 1500 калорій;

– час на приготування не більше трьох годин;

– на одну-дві людини;

– страва має містити картоплю.

Ці параметри у вигляді об’єкта із трьома складовими частинами – користувачем, рецептом та цільовим рецептом представлені на рисунку 4.10.

Змінивши деякі параметри можна досягнути ще меншої кількості, і продемонструвати роботу сповіщень системи.

Для полегшення демонстрації на кожному етапі виводиться кількість можливих рекомендацій. У цьому випадку можна побачити, що кількість рецептів змінилася із 23 тисяч до 98.

Останній параметр визначає ідетифікатор цільового рецепта, тобто рецепта, відносно якого вираховується коефіцієнт подібності інших рецептів у наборі даних.

```
query = { "user": {
    'age': 18,
    'allergies': ['pork','eggs'],
    'issues': [],
    'calories': 1500,
    'diet': True,
    'advanced_diet': False,
    'skill': False
}, "recipe":{
    'dish_complexity': 'Medium',
    'calories': 1500,
    'advanced_diet': False,
    'time_to_make': 170,
    'for_groups': False,
    'quisine': False,
    'dish': False,
    'ingredients': ['potato'],
    'tags':[]
}, "target": False}
```

Рисунок 4.10 – Вхідні дані

У наступному прикладі до необхідних інгредієнтів у страві було додано кілька спецій, що призвело до того, що кількість рекомендацій впала з 98 до 13. На рисунку 4.11 можна побачити сповіщення системи про те, що бажано змінити параметр інгредієнтів, бо результуюча кількість рекомендацій замала.

У подальшому прикладі із подібними рецептами також було змінено вхідні дані, було додано теги, хворобу та цільовий рецепт. Це також призвело до зменшення результуючого набору даних.

```
Initial 231637
allergies! 34078
calories! 31303
diet! 20007
complexity! 15755
calories! 15755
time! 15261
dish1! 15261
ingr! 98
final 98
```

```
Initial 231637
allergies! 34078
calories! 31303
diet! 20007
complexity! 15755
calories! 15755
time! 15261
dish1! 15261
Change your recipe restrictions if possible, reccomendations amount is small.
We recommend changing the Ingredients restriction.
ingr! 13
final 13
```

Рисунок 4.11 – Зміна кількості рекомендацій при різних запитах

На рисунку 4.12 продемонстровано частину результату роботи системи.

| | name | minutes | tags | n_steps | steps | description | ingredients | n |
|---|---|---|---|---|---|---|---|---|
| **34542** | canadian bacon and potato omelet | 20 | ['30-minutes-or-less', 'time-to-make', 'course... | 12 | ['whisk together eggs , egg whites , canadian ... | this omelet is based on a recipe i saw in redb... | ['eggs', 'egg whites', 'canadian bacon', 'cila... | |
| **37559** | catfish cakes | 50 | ['60-minutes-or-less', 'time-to-make', 'course... | 10 | ['broil catfish fillets in baking pan 5-8 minu... | a lady that i work with turned me on to these.... | ['catfish fillets', 'eggs', 'potato', 'onion',... | |
| **58540** | corned beef hash and eggs | 17 | ['30-minutes-or-less', 'time-to-make', 'course... | 7 | ['cook onion and pepper in butter until tender... | great weekend breakfast for 2. | ['onion', 'green pepper', 'butter', 'cooked co... | |
| **67497** | crustless quiche 2 w bacon and feta | 55 | ['weeknight', '60-minutes-or-less', 'time-to-m... | 6 | ['whip eggs and milk', 'i use kitchenaid', 'ad... | this is another version of my other recipe for... | ['eggs', '1% low-fat milk', 'self rising flour... | |
| **78989** | egg and tomato salad sandwiches | 25 | ['30-minutes-or-less', 'time-to-make', | 4 | ['peel the eggs and cut in half and remove | love egg salad and this one is from the | ['eggs', 'low-fat mayonnaise', 'english | |

Рисунок 4.12 – Частина результату роботи системи

Далі було протестовано роботу функції рекомендації подібних рецептів, для цього у запит було додано target – id рецепту із підібраного набору. Роботу системи представлено на рисунку 4.13, де можна побачити що в результаті було створено атрибут «Similarity», що відображає подібність кожного рецепту до цільового рецепту у проміжку від 0 до 1, де 1 – сам цільовий рецепт. Цей атрибут вираховується за допомогою функції подібності, вагові коефіцієнти ж були визначені за допомогою тестування та аналізу предметної області. Завдяки цьому було створено приорітет атрибутів, і відповідно задано коефіцієнти від 0.2 до 0.9 кожному атрибуту.

| Dish | Quisine | Groups | Vegan | Similarity |
|---|---|---|---|---|
| side-dishes | north-american | False | True | 0.743371 |
| desserts | NaN | False | False | 0.657158 |
| quick-breads | north-american | False | True | 0.737296 |

Рисунок 4.13 – Демонстрація роботи функції подібності

Далі, відсортувавши набір даних за зменшенням ми отримаємо результуючий набір подібних рецептів. На рисунку 4.14 можна побачити графік, що відображає подібність усіх рецептів для прикладу, що розглядається.

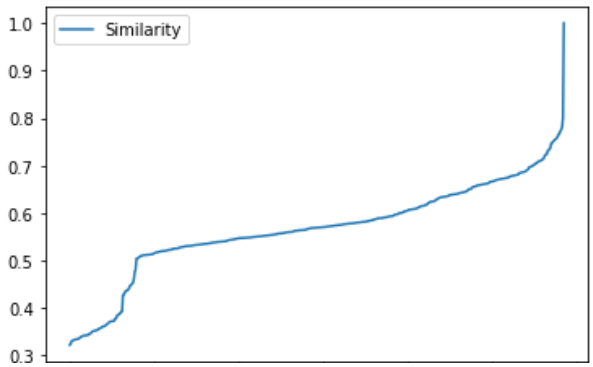


Рисунок 4.13 – Подібність рецептів результуючого набору даних

Із нього можна зробити висновок, що подібність у цьому випадку коливається від 0.3 до 0.8, із великим плато рецептів із подібністю від 0.5 до 0.6.

# ВИСНОВКИ

В даній кваліфікаційній роботі було показано, як використання такої технології як рекомендаційної системи, заснованої на знаннях може покращити підбір та зберігання кулінарних рецептів.

Було проаналізовано рекомендаційні системи в загальному плані, та особливості побудови саме систем, заснованих на знаннях. Так, як подібні системи розробляться для областей, де пошукові елементи значно налаштовуються та у ній можна виділити певні залежності, предметна область кулінарних рецептів є підходящою для цього.

Було проведено детальний аналіз предметної області та існуючих у ній кулінарних рекомендаційних систем. Прикладів побудувати подібну систему засновуючись лише на знаннях було мало, частіше використовувалися інші підходи. Через порівняння різних додатків були виявлені переваги та недоліки кожної із них. Після чого було встановлено, що потрібно реалізувати в нашій системі для рішення цих недоліків, зберігши переваги кожного із систем.

Було обрано набір даних кулінарних рецептів та зроблено його передобробку, щоб позбавитися нерелевантних даних та виділити найбільш важливі для побудови системи. Завдяки аналізу набору даних із нього було отримано залежності та створено базу знань. Використовуючи підхід, заснований на обмеженнях було побудовано та протестовано рекомендаційну систему. Далі її було перетворено на гібридну, додавши до неї міркування на основі прецедентів, реалізоване у вигляді подібних рецептів. Таким чином система має два типа обмежень – для користувача та для рецепту та можливість шукати рецепти, подібні до іншого завдяки функції подібності. Крім того система має сповіщення про обмеження, які призводять до значного зменшення рекомендацій, тобто таких, які частіше за все є хибними.

Розроблена система довела, що можливо побудувати рекомендаційну систему у предметній області, засновану лише на знаннях. Але хоча її і можна

покращити певним чином, наприклад виділивши більше категорій у предметній області, визначити залежності за допомогою ембеддингів чи створити онтологію інгредієнтів для поліпшення рекомендацій, є недоліки котрі неможливо усунути, не використовуючи інші методи побудови рекомендаційних систем. Мова йде про персоналізацію під конкретного користувача, наприклад створення списків рецептів чи просто запам`ятовування улюблених інгредієнтів, або й щось більш комплексне, як відстеження комплексної дієти впродовж тривалого часу, а не в межах однієї страви. Тому цю систему раціонально буде використовувати у гібридній моделі із CB чи CF моделлю.

Рекомендаційну систему написано на мові програмування Python із використанням середовища Jupyter Notebook та Google Colab.

Розроблена система відповідає сформульованим вимогам завдання, таким чином, завдання на кваліфікаційну роботу виконано в повному обсязі.

Результати проведених досліджень також були опубліковані в рамках 25-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» [44].

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Charu C. Aggarwal. Recommender Systems. Switzerland: Springer International Publishing, 2016. 168 с.

2. Bouraga S., Jureta I., Faulkner S., Hersse C. Knowledge-Based Recommendation Systems: A Survey. *International Journal of Intelligent Information Technologies*, 2014. 2 с.

3. Набори даних movielens. URL: http://grouplens.org/datasets/movielens/ (дата звернення: 17.04.2021).

4. Сайт Amazon. URL: http://www.amazon.com (дата звернення: 17.04.2021).

5. Сайт Netflix. URL: http://www.netflix.com (дата звернення: 17.04.2021).

6. Freyne J., Berkovsky S. Recommending food: reasoning on recipes and ingredients: Proceedings of the International Conference on User Modeling, Adaptation, and Personalization UMAP, 2010. С. 381–386.

7. Jannach D., Zanker M., Felfernig A., Friedrich G. Recommender Systems. An Introduction, 2010.

8. Zhou T., Kuscsik J, Liu J., Medo M., Wakeling J.R., Zhang Y. Solving the apparent diversity-accuracy dilemma of recommender systems: Proceedings of the National Academy of Sciences, 2010. С. 4511–4515.

9. Sheth S., Bell J., Arora N., Kaiser G. Towards Diversity in Recommendations using Social Networks. URL: https://academiccommons.columbia.edu/doi/10.7916/D81J9JPV (дата звернення: 17.04.2021).

10. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013.

11. Skeels C., Patel Y. Caviar's Word2Vec tagging for menu item recommendations., 2018. URL: https://medium.com/square-corner-blog/caviars-word2vec-tagging-for-menu-item-recommendations-13f63d7f09d8 (дата звернення: 17.04.2021).

12. Knowler W., Barrett-Connor E., Fowler S., Hamman R., Lachin J., Walker D., Nathan D. Reduction in the incidence of type 2 diabetes with lifestyle intervention or metformin. *New England Journal of Medicine*, 2002. С. 393–403.

13. Elsweiler D., Harvey M., Ludwig B., Said A. Bringing the healthy into food recommenders. / ed. V. Ge, F.Ricci. DMRS, CEUR-WS.org, CEUR workshop proceedings, 2015. Т. 1533. С. 33–36.

14. Mika S. Challenges for nutrition recommender systems. Context Aware Intelligent Assistance: Proceedings of the CEUR-WS.org, Workshop, 2011. С. 25–33.

15. El-Dosuky M., Rashad M., Hamza T., El-Bassiouny A. Food recommendation usingontology and heuristics. AMLTA, Springer communications in computer and information science: 2012. Т. 322. С. 423–429.

16. Svensson M., Laaksolahti J., Ho'ok K., Waern A. A recipe based on-line food store: Proceedings of the 5th international conference on intelligent user interfaces, New York, NY, USA: IUI, 2000. С. 260–263.

17. Elahi M., Ge M., Ricci F., Fernandez-Tobias I., Berkovsky S., Massimo D. Interaction design in a mobile food recommender system: Proceeding of the CEUR workshop, 2015. Т.1438. С. 49–52.

18. Kuo F., Li C., Shan M., Lee S. Intelligent menu planning: Recommending set of recipes by ingredients: Proceedings of the ACM multimedia 2012 workshop on multimedia for cooking and eating activities, ACM, New York, NY, USA: CEA, 2012. С. 1–6.

19. Aberg J. Dealing With malnutrition: A meal planning system for elderly, 2006.

20. Schwartz B. The Paradox of Choice: Why More Is Less. USA: Ecco, 2003.

21. Husain W., Wei L., Cheng S., Zakaria N. Application of data mining techniques in a personalized diet recommender system for cancer patients. IEEE Colloquium on Humanities, Science and Engineering Research (CHUSER 2011), 2011. C. 239–244.

22. Freyne J., Berkovsky S., Smith G. Recipe Recommendation: *Accuracy and Reasoning*: Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization, UMAP, 2011. C. 99–110.

23. Oh Y., Choi A., Woo W. u-BabSang: a context-aware food recommendation system. *Journal of Supercomputing*, 2010. C.61–81.

24. Yang L., Hsieh C., Yang G., Pollak J.P., Dell N., Belongie S., Cole C., Estrin D. Yum-me. A personalized nutrient-based meal recommender system. ACM Trans. Inf. Syst, 2017. 36 c.

25. Ueta T., Iwakami W., Ito T. A recipe recommendation system based on automatic nutrition information extraction: Proceedings of the fifth international conference on knowledge science, engineering and management, Springer-Verlag, Berlin, Heidelberg: KSEM, 2011. C. 79–90.

26. Van Pinxteren Y., Geleijnse G., Kamsteeg P. Deriving a recipe similarity measure for recommending healthful meals: Proceedings of the 16th international conference on intelligent user interfaces. New York: IUI, 2011. C. 105–114.

27. Hoxmeier J., Manager C. System response time and user satisfaction. *An experimental study of browser-based applications*: Proceedings of the association of information systems Americas conference, 2000. C. 10–13.

28. Snooks M. Health Psychology: Biological, Psychological, and Sociocultural Perspectives. USA, 2009.

29. Kolodner J. Case-Based Reasoning. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

30. Aamodt A., Plaza E. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 1994. C. 39–59.

31. Similarity and compromise / McSherry D., Aamodt A., Bridge D., Ashley K., editors, ICCBR 2003, the 5 International Conference on Case-Based Reasoning. Trondheim, Norway: 2003. C. 291–305.

32. Shimazu H. Expertclerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, 2002. C. 223–244.

33. Montaner M., L'opez B., De la Rosa J. *Improving case representation and case base maintenance in recommender systems*: Proceedings of the 6th European Conferenceon Case Based Reasoning. Aberdeen, Scotland: Springer Verlag, 2002. C. 234–248.

34. Burke R. Knowledge-based recommender systems. / ed. J. E. Daily, A. Kent, and H. Lancour. *Encyclopedia of Library and Information Science*. USA: Marcel Dekker, 2000. T. 69.

35. McGinty L., Smyth B. On the role of diversity in conversational recommender systems. / ed. A. Aamodt, D. Bridge, and K. Ashley. ICCBR 2003, the 5th International Conference on Case-Based Reasoning. Trondheim, Norway: 2003. C. 276–290.

36. Smyth B. Case-based recommendation. The Adaptive Web. USA: Springer, 2007. C. 342–376.

37. Smyth B., McClave P. Similarity vs. diversity. Case-Based Reasoning Researchand Development, 2001. C. 347–361.

38. McCarthy K., Reilly J., McGinty L., Smyth B. On the dynamic generation of compound critiques in conversational recommender systems. *Adaptive Hypermedia and Adaptive Web-Based Systems*, 2004. C. 176–184.

39. Aggarwal C., Han J. Frequent pattern mining. New York, NY: Springer, 2014.

40. Mahmood T., Ricci F. Learning and adaptivity in interactive recommender systems. International Conference on Electronic Commerce, 2007. С. 75–84.

41. Визначення Jupyter Notebook. URL: https://en.wikipedia.org/wiki/Project_Jupyter (дата звернення: 17.04.2021).

42. Визначення Pandas. URL: https://en.wikipedia.org/wiki/Pandas_%28software%29 (дата звернення: 17.04.2021).

43. Визначення NumPy. URL: https://en.wikipedia.org/wiki/NumPy (дата звернення: 17.04.2021).

44. Беседин Ф.О. Исследование основных подходов к построению рекомендательных систем, основанных на знаниях: тез. докл. 25-го Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті». Харків: ХНУРЕ, 2021. Том 6, С. 25-26.