Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ НАВЧАЛЬНОЇ СИСТЕМИ НА ОСНОВІ ОНТОЛОГІЙ

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
к.т.н., доц. Жежерун О. П.

_____

“___” _____ 2020 р.

Виконала студентка 4-го р.н.
Жиліна Є. К.

_____

“___” _____ 2020 р.

Київ – 2020

Тема: «Розробка рекомендаційної навчальної системи на основі онтологій»

Календарний план виконання роботи:

	Назва етапу курсової роботи	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	30.10.2019	
2.	Огляд літератури за темою роботи	30.01.2020	
3.	Створення практичної частини роботи	20.03.2020	
4.	Написання текстової частини роботи	05.04.2020	
5.	Надання роботи для попередньої перевірки	10.04.2020	
6.	Коригування роботи за результатами перевірки	15.04.2020	
7.	Подання роботи на кафедру для перевірки на плагіат	19.04.2020	
8.	Захист курсової роботи	19.04.2020	

Студент _____

Керівник _____

“___” _____ 2020 р.

ЗМІСТ

Анотація

В теоретичній частині цієї роботи розглядаються основні факти про рекомендаційні системи, парадигми їх проектування з прикладами конкретних способів реалізації, наприклад, матрична факторизація як метод колаборативної фільтрації. Окремий підрозділ приділено використанню онтологій у рекомендаційних системах, їх перевагах та випадках влучного використання. В практичній частині роботи описуються методи реалізації, функціонал та особливості користувацького інтерфейсу рекомендаційної системи, що використовує онтологію для порівняння академічних дисциплін за наданими характеристиками.

Вступ

Рекомендаційні системи за останні роки стали популярним засобом покращення ефективності роботи інтернет ресурсів, що призвело до одночасного підвищення зручності користування ресурсом та прибутку ресурсів, які заробляють на активності користувачів. Сферу навчання рекомендаційні системи також не оминули і зараз їх активно використовують на сайтах онлайн курсів і вебінарів, академічних ресурсах та сайтах з будь-якою іншою навчальною інформацією. Не залежно від сфери використання та особливостей реалізації метою рекомендаційної системи є надання користувачеві найбільш релевантної для нього інформації, якою можуть виступати різноманітні продукти від фільмів і книжок до будинків або фінансових послуг.

На сьогоднішній день існує широке різноманіття способів імплементації рекомендаційних систем. Деякі з них є більш універсальними і можуть бути використані в будь-якій сфері, інші — спираються на особливості даної сфери і конкретної категорії продуктів. Є такі, що використовують дані про користувачів, збирають дані про взаємодію користувачів з продуктами системи або про конкретні уподобання користувачів. В залежності від потреб ресурсу, складності структури його продуктів, аспектів, які потрібно врахувати для генерації коректної рекомендації рекомендаційна система може або не використовувати моделі взагалі, або використовувати моделі різних рівнів складності. Саме про парадигми проектування рекомендаційних систем йтиметься у першому розділі.

Другий розділ роботи присвячений розробці навчальної рекомендаційної системи з використанням онтології. Спираючись на зібрані теоретичні відомості про рекомендаційні системи буде обрано конкретне застосування і методи, за допомогою яких буде відбуватися розробка PC та графічного інтерфейсу.

1 Рекомендаційні системи

1.1 Загальний опис

Рекомендаційні системи вважаються одним з найбільш потужних засобів сучасного цифрового простору. Більшість популярних новітніх інтернет платформ, мета якої надати користувачу інформацію, що його цікавить, використовують рекомендаційні системи для більшого задоволення клієнта і як наслідок його повернення на платформу. Такі ресурси і веб сайти як Amazon, Facebook, Google, Instagram, Netflix, Youtube пропонують користувачеві такі об’єкти (результати пошуку, сторінки людей чи організацій, дописи, різноманітний медіа контент від світлин до фільмів, продукти, товари та інше), що був спеціально обраний, спираючись на попередньо зібрану персональну інформацію про користувача. Таким чином зручність і задоволення користуванням підвищується і, як правило, зі збільшенням кількості даних про активність та уподобання користувачів збільшується точність пропозицій.

Для e-commerce платформ ефективність наявної рекомендаційної системи є особливо важливою, оскільки товар, що відповідає персональним потребам конкретної людини, має більшу імовірність бути придбаним, аніж товар, що просто відповідає заданому пошуковому запиту. Amazon відзначає, що в околі 35% відсотків продаж відбуваються завдяки їх рекомендаційній системі. Переважно рекомендаційні системи застосовуються для того, щоб показати клієнту товар, якого він не шукав, але який з великою імовірність його зацікавить. Це, звичайно, має як позитивні наслідки, так і негативні. Так Youtube у розділі “Запропоноване” відображає такі відео, що є схожими на те, яке користувач переглядає в даний момент, а Facebook пропонує додати у друзі людину, яку користувач скоріше за все знає. Google відображає на екрані користувача саме таку рекламу, що людина не втримається клікнути на неї;

механізми Facebook, щоб підвищити популярність рекламованої сторінки, спершу будуть показувати її тим людям, які мають звичку коментувати дописи і розміщувати їх на своїй сторінці; а Amazon запропонує клієнту такий товар, що він не стримається і вмить його придбає, навіть якщо він йому насправді не потрібен. Високоякісна рекомендаційна система може перетворити досвід користувача із прикрого на приємний і закласти основу для довгострокової довіри до даного ресурсу.

Існує декілька парадигм проектування рекомендаційних систем і вибір однієї з них напряму залежить від того, які аспекти роботи даного ресурсу необхідно врахувати і яким чином. Найпопулярніші з них це:

- колаборативна фільтрація;
- фільтрація за змістом;
- фільтрація на основі знань.

До кожної з цих парадигм належать різні методи роботи рекомендаційних систем. У наступних розділах наведені вище парадигми та методи, що до них належать, будуть розглянуті детально.

1.2 Колаборативна фільтрація

Методи, що належать до колаборативної фільтрації, ґрунтуються виключно на попередніх взаємодіях, зафіксованих між користувачами та предметами, з метою створення нових рекомендацій. Ці взаємодії зберігаються у так званій матриці взаємодій між елементами та користувачами. За основу взята ідея, що одних тільки цих даних про взаємодію достатньо, щоб визначити схожих користувачів або схожі предмети і згенерувати прогноз, спираючись на розраховану близькість. Група методів колаборативної фільтрації поділяється на дві підгрупи:

- методи, що базуються на даних (memory based), визначають найбільш схожі продукти за принципом k найближчих сусідів, для чого необхідні лише дані з матриці взаємодій;
- методи, що базуються на моделях (model based), визначають найбільш схожі продукти за допомогою накладання даних з матриці взаємодій на обрану модель, яка описує певну логіку того, як саме взаємодіють користувачі і продукти [1].

Головною перевагою колаборативного підходу є відсутність потреби володіти даними про користувачів або продукти, тільки їх взаємодія необхідна для розрахунку рекомендацій. Саме тому цей підхід є універсальним, адже властивості самих об’єктів не відіграють ніякої ролі під час розрахунків. Крім того, чим більшою кількістю інформації про взаємодію об’єктів володіє система, тим більш точнішими стають рекомендації, тож з часом якість системи підвищується.

При цьому головним недоліком колаборативного підходу постає той факт, що при невеликій кількості даних про взаємодію в системі якість рекомендацій стрімко погіршується. На початку роботи рекомендаційної системи виникає так звана проблема «холодного старту», тобто ситуація коли знань про попередні взаємодії недостатньо для коректного розрахунку рекомендацій. Цю проблему зазвичай вирішують одним з таких способів:

- рекомендують випадкові продукти новим користувачам (або нові продукти випадковим користувачам);
- рекомендують популярні продукти новим користувачам (або нові продукти користувачам з найбільшою активністю);
- рекомендують тестовий набір різних продуктів новим користувачам (або нові продукти випадковому набору користувачів);

- використовують обраний неколаборативний метод на перших етапах роботи системи.

1.2.1 Методи, що базуються на даних

1.2.1.1 Підхід “Користувач - користувач”

Суть цього підходу заключається в тому, що для того, щоб згенерувати рекомендацію для користувача, система спочатку визначає набір користувачів, чиї профілі є максимально схожими на профіль даного користувача. Профілем користувача тут виступає вектор взаємодій користувача з продуктами. Після вибору k найближчих «сусідів» система знаходить найпопулярніші продукти серед тих, що були згадані в профілях користувачів-сусідів.

Очевидно, даний підхід передбачає три етапи:

а) виділити окремі профілі усіх користувачів в системі;

б) визначити схожість між даним профілем та усіма іншими та виділити k профілів з найбільшою схожістю на даний;

в) відібрати n найпопулярніших продуктів.

На етапі визначення схожості профілів варто застосовувати коректний механізм для визначення відсотку «однаковості» профілів. Розглянемо проблему на прикладі. В системі зареєстровані п’ять продуктів і три користувача, кожен з яких має щонайменше одну взаємодію хоча б з одним продуктом. Користувач №1 взаємодіяв з продуктами №1, №2, №3. Користувач №2 взаємодіяв з продуктом №1. Користувач №2 взаємодіяв з продуктами №1, №2, №4, №5. Метою системи в даний момент є запропонувати продукти для користувача №1. При перевірці профілів на схожість виникає наступна ситуація:

- серед продуктів користувача №1 є усі продукти користувача №2 (здається, це 100% співпадіння),

- серед продуктів користувача №1 є два з чотирьох продуктів користувача №3 (тобто 50% співпадіння).

При цьому такі твердження про співпадіння помилкові, адже очевидно, що профілі №1 і №2 не є на 100% відсотків схожими, а отже коректною буде така логіка:

- серед продуктів користувача №2 є лише один з трьох продукт користувача №1 (співпадіння 33,3%),
- серед продуктів користувача №3 є два з трьох продуктів користувача №1 (співпадіння 66,6%).

Варто зазначити, що взаємодія між користувачем та продуктом може включати в себе різноманітні характеристики. Наприклад, якщо рекомендаційна система пропонує для користувачів фільми для перегляду, то характеристики взаємодії можуть бути такі:

- тип взаємодії (читання опису фільму, перегляд фільму);
- час взаємодії (у секундах).

Якщо в системі передбачено можливість оцінювання користувачем деякого фільму за шкалою, то оцінку фільму також буде включено до взаємодії.

1.2.1.2 Підхід “Продукт - продукт”

Цей підхід заключається в тому, що для того, щоб запропонувати користувачеві новий продукт, необхідно знайти такі продукти, які є максимально схожими на ті продукти, які вже наявні в векторі взаємодій цього користувача і ці взаємодії є позитивними.

Позитивною взаємодією на прикладі рекомендаційної системи з фільмами можна назвати таку взаємодію, коли користувач саме переглянув фільм, а не лише прочитав його опис. Також можна врахувати час перегляду фільму, наприклад, якщо користувач переглянув менше 30% усієї картини, і це можна інтерпретувати як той факт, що користувач не був достатньо зацікавленим, то такий перегляд не вважається “позитивним”. Усе залежить від реалізації самої рекомендаційної системи.

Два продукти вважаються схожими, якщо їх вектори взаємодій є схожими, тобто якщо більшість з користувачів, які якось провзаємодіяли з першим продуктом, провзаємодіяли з другим продуктом максимально схоже.

Даний підхід передбачає три етапи для того, щоб згенерувати рекомендацію для користувача:

а) виділити окремі профілі (вектори взаємодій) усіх продуктів в системі;
б) визначити продукт, який користувач вподобав найбільше, та отримати профіль цього продукту;
в) визначити схожість між профілем даного продукту та усіма іншими та виділити k профілів з найбільшою схожістю на даний.

Очевидно, що більш релевантними будуть такі продукти, які одночасно схожі не тільки на один «позитивний» продукт, а на декілька (це підвищує ймовірність того, що цей продукт дійсно сподобається користувачу і також отримає в подальшому позитивну взаємодію). Для цього під час етапу б) замість «найулюбленішого» продукту користувача можна обрати n найулюбленіших і порівнювати решту продуктів з кожним з них.

1.2.2 Методи, що базуються на моделях

1.2.2.1 Матрична факторизація

Основне припущення матричної факторизації полягає в тому, що існує такий неявний простір характеристик з досить низькою розмірністю, за допомогою якого можна представити як користувачів, так і продукти, і при цьому скалярний добуток вектору користувача на вектор продукту буде відповідати значенню взаємодії між користувачем і продуктом з матриці взаємодій.

Наприклад, якщо у матриці взаємодій зберігаються взаємодії між користувачами та фільмами, то для моделювання цих взаємодій можна припустити, що існують такі характеристики, які одночасно:

- описують властивості фільму (характеристика має високу оцінку, якщо фільм відповідає цій характеристиці, і низьку, якщо навпаки);
- описують уподобання користувачів (характеристика має високу оцінку, якщо користувачу подобається така характеристика, і низьку, якщо навпаки).

На рисунку 1.1 зображено приклад матриці взаємодій між користувачами і фільмами та дві матриці, що є результатом факторизації матриці взаємодій, що представляють користувачів та фільми. У матриці зліва зображені співвідношення між характеристиками та користувачами, а у матриці зверху — між характеристиками та фільмами. Припустимо, що характеристики в даному контексті відповідають жанрам фільмів. Тоді, інформація про користувача це те чи подобається (або наскільки подобається) йому даний жанр, а для фільмів — чи відповідає (або наскільки відповідає) фільм даному жанру.

	Ф1	Ф2	Ф3	Ф4	Ф5
Х1	3	1	1	3	1
Х2	1	2	4	1	3

	Х1	Х2
К1	1	0
К2	0	1
К3	1	0
К4	1	1

	Ф1	Ф2	Ф3	Ф4	Ф5
К1	3	1	1	3	1
К2	1	2	4	1	3
К3	3	1	1	3	1
К4	4	3	5	4	4

Рисунок 1.1 — Матриця взаємодій та її факторизація. Позначення: К - користувач, Ф - фільм, Х - характеристика.

Однак, ці характеристики є неявними, а отже не надаються системі прямо. Система повинна підібрати такі допоміжні характеристики самотужки і згенерувати власне представлення користувачів і предметів виходячи з них. Через те, що ці характеристики генеруються системою, кожна з цих характеристик відіграє відповідне математичне значення для самої системи, але їх складно інтуїтивно проінтерпретувати, а отже для людини вони не мають сенсу і є складними (якщо не неможливими) для розуміння.

Проте, структури, що породжуються системою, нерідко є надзвичайно близькими до інтуїтивної декомпозиції, яку могла запровадити і людина. Дійсно, наслідком такої факторизації є те, що близькі один до одного користувачі (з точки зору їх уподобань), а також близькі продукти (з точки зору їх властивостей) в результаті мають схожі представлення в отриманому неявному просторі. [2]

Класичний ітеративний підхід для виконання матричної факторизації базується на алгоритмі градієнтного спуску, що дозволяє обробляти дуже великі матриці без необхідності завантажувати всі дані одночасно.

Припустимо, що M — це оригінальна матриця взаємодій розміром n на m, в якій тільки невелика кількість продуктів була оцінена користувачами, а більшість клітинок матриці залишилась незаповненою. В результаті факторизації необхідно отримати такі матриці X та Y, що

$$M \approx XY,$$

де X — це матриця з даними про користувачів розміром n на l, де кожен рядок відповідає одному користувачу
і Y — це матриця з даними про продукти розміром m на l, де кожен рядок відповідає одному продукту:

$$user_i \equiv X_i \qquad \forall i \in \{1, \dots, n\}$$
$$item_j \equiv Y_j \qquad \forall j \in \{1, \dots, m\}$$

В даному контексті l — це розмірність неявного простору характеристик, в якому будуть представлені (охарактеризовані) користувачі та продукти. Отже, необхідно знайти такі матриці X і Y, що їх добуток максимально наближений до існуючої матриці взаємодій. Позначимо E набір пар (i, j) таких, що значення $M(i, j)$ є визначеним, тобто ті значення, де користувачі встановили мали взаємодії з продуктами. Метою є знайти такі X та Y, при яких мінімізується значення так званої похибки реконструкції

$$(X, Y) = \underset{X,Y}{\operatorname{argmin}} \sum_{(i,j) \in E} [(X_i) \cdot (Y_j) - M_{ij}]^2$$

Матриці X і Y можна отримати шляхом застосування методу градієнтного спуску, для якого варто виділити деякі особливості, які полягають в наступних твердженнях:

- градієнт не обов’язково розраховувати для всіх пар в E на кожному кроці, можна обрати деяку підмножину пар і оптимізувати цільову функцію частинами;
- значення X та Y не обов’язково оновлювати одночасно, можна виконувати градієнтний спуск почергово на X і Y, фіксуючи одну матрицю і оптимізуючи іншу на одній ітерації.

Коли факторизацію отримано, обсяг даних стає значно меншим, що постає додатковою перевагою, а обсяг інформації щодо взаємодій користувачів та продуктів залишається тим самим, при цьому ми отримуємо «нові» дані про характеристики користувачів та продуктів. Для того, щоб згенерувати нову рекомендацію (фактично передбачити якою була б взаємодія користувача з тим чи іншим продуктом) необхідно лише помножити вектор користувача на вектор продукту.

Факторизацію також можна застосувати для методів «Користувач - користувач» і «Продукт - продукт». В цих випадках замість того, щоб шукати «найближчого сусіда» в усій матриці взаємодій, достатньо порівняти тільки невеликі вектори, що влучно характеризують користувача або предмет.

Звичайно ж, існують методи покращення матричної факторизації та адаптації цього методу під конкретні вимоги. Наприклад, якщо взаємодія між користувачем та продуктом оцінюється булевим значенням, то виникає необхідність при реконструкції матриці взаємодій отримувати тільки 0 або 1. Для цього до отриманого під час оптимізації моделі значення застосовують деяку функцію.

1.3 Фільтрація за змістом

На відміну від колаборативних методів, які для створення рекомендацій використовують тільки матрицю взаємодії між користувачами і продуктами, методи, що належать до парадигми фільтрації за змістом, спираються також на дані про користувачів та продукти, які не стосуються взаємодії. Зазвичай, для користувача це дані з анкети користувача, які заповнюються при реєстрації, дані про географічне положення тощо, для продукту — характеристики продукту, які вказуються при створенні нового продукту.

В методах, що належать до парадигми фільтрації за змістом, задача знайти коректну рекомендацію фактично постає або задачею класифікації (визначити належить продукт до уподобань користувача чи ні) або задачею регресії (визначити можливу оцінку, яку користувач дасть даному продукту). В обох випадках необхідно зазначити модель, яка буде випливати з характеристик користувача, продукта або них обох, саме ці характеристики і виступають «змістом» в даному контексті. [2]

Якщо класифікація або регресія відбувається на основі характеристик користувачів, то такий підхід називається орієнтованим на продукт, адже моделювання, оптимізації та обрахунки будуть відбуватися для певного обраного продукту. Тоді використання такої моделі допоможе знайти відповідь на питання «Які ймовірності того, що кожен користувач з усіх можливих буде мати позитивну взаємодію з даним продуктом?» Користувачі, очевидно, змінюються, але продукт залишається тим самим.

Якщо брати за основу характеристики продуктів, то такий підхід, відповідно, буде вважатися орієнтованим на користувача, бо моделювання, оптимізації та обрахунки будуть відбуватися для певного користувача. Тоді відбувається

тренування конкретної моделі для конкретного користувача і ця модель дає відповідь на питання «Які ймовірності, що даний користувач буде мати позитивну взаємодію кожним з запропонованих продуктів?»

Використання продукто-орієнтовного підходу зазвичай надає більш ґрунтовні результати, бо з кожним продуктом мали взаємодію досить велика кількість користувачів, а отже було проаналізовано достатньо багато даних. Проте, результати можуть стати менш персоналізованими, адже навіть якщо деякі користувачі мають схожі характеристики, їх уподобання можуть суттєво відрізнятися, але ці деталі модель опускає.

Використання користувацько-орієнтовного підходу, в свою чергу, означає, що результат буде більш персоналізованим, адже користувач є центром моделі, але менш ґрунтовним, бо до уваги беруться тільки предмети, з яким користувач мав взаємодію, що їх зазвичай не достатньо багато.

В дійсності, дізнатися істинну інформацію про користувача, як правило, набагато складніше, ніж інформацію про продукт. Багато людей під час реєстрації не повністю заповнюють форму або заповнюють її фальшивими даними заради власної безпеки, але найчастіше тих даних, що надають користувачі під час реєстрації, просто не вистачає для побудови ґрунтовної моделі. Навпаки ж, під час реєстрації нового продукту на конкретному ресурсі, людина, яка заповнює дані про продукт, зацікавлена в тому, щоб його опис був максимально повним заради того, щоб в подальшому його було запропоновано відповідним користувачам і ці рекомендації були якомога більш коректними.

В залежності від складності відносин між користувачами та продуктами, які необхідно відобразити, складність моделі буде варіюватися від простих моделей до глибоких нейронних мереж. Якщо в конкретній системі не можна застосувати

продукто-орієнтовний або користувацько-орієнтовний підходи, то можливо також поєднувати вектори, що відповідають характеристикам користувача та продукту і разом закладати їх у архітектуру нейронної мережі.

1.4 Фільтрація на основі знань

Головна ідея парадигми фільтрації на основі знань полягає в тому, що система визначає рекомендації для користувача не на основі його взаємодій і не на основі даних про нього, а на основі тих пошукових запитів, які він здійснює. Система може навіть запитати користувача уточнити його запит за допомогою деяких правил (фільтрів) або обрати такий продукт, що найбільше задовольняє його потребу, з запропонованих. Фактично, таким чином користувач власноруч надає системі дані про те, що саме його цікавить. [3]

Спочатку фільтрація на основі знань здається звичайним пошуком продукту за характеристиками, але вона, на відміну від банального фільтрування, може персоналізувати результати пошуку для кожного окремого користувача і фільтрувати продукти не строго за запитом, а враховуючи деякий показник, який показує наскільки даний продукт підпадає під шукану категорію. При цьому даний підхід включає активну участь користувача не тільки на момент першого пошуку, а й під час корекції пошукового запиту, наприклад, коли користувач задає уточнюючі фільтри.

Очевидно, що користувацький інтерфейс передбачає зручність використання, а отже назви параметрів фільтрування можуть бути адаптовані і не відповідати справжнім назвам характеристики продукту. Наприклад, при пошуку фільмів користувач вказує правило «нові», але у жодного фільму немає такої характеристики. Система має володіти певними доменними знаннями, щоб

проводити паралель між терміном «новий» і фільмами, дата виходу яких була не давніше, ніж, припустимо, два роки тому.

Для того, щоб коректно оцінювати чи підпадає деякий продукт під заданий пошуковий запит, необхідно для кожної характеристики продукту мати:

- допоміжну функцію схожості, за допомогою якої визначається різниця між двома значеннями цієї характеристики;
- значення важливості цієї характеристики для даного продукту.

Рекомендаційні системи, що використовують методи фільтрації на основі знань, поширені серед ресурсів, на яких продукти мають складну структуру і багато характеристик, але де взаємодій у формі виставлення користувачами оцінок недостатньо для використання матриці взаємодій для майбутніх припущень. Загалом до такої категорії можна віднести сайти з продажом або довгостроковою орендою квартир, будинків, машин та інших дороговартісних товарів, а також послугами у сфері фінансів, туристичними послугами тощо.

1.5 Застосування онтологій у рекомендаційних системах

У контексті рекомендаційних систем структуризація та систематизація продуктів та рекомендація продуктів, що відповідають потребам користувача (отриманих напряму через пошуковий запит чи фільтрацію або через дані про уподобання користувача), стає більш складною задачею, коли категорій та різновидів продуктів на ресурсі стає все більше. Для цього критично необхідно мати змогу точно описати характеристики продуктів та користувачів. [1]

Представлення характеристик користувача або продукту з використанням онтологій є більш точним та багатшим, допомагає уникати неоднозначності, бо чітко визначає домен певної характеристики, та полегшує впровадження

підкатегорій або будь-якого іншого поділу властивостей на більш конкретизовані.

Використання онтологій є особливо вигідним в задачах, напряму пов’язаних з персоналізацією, наприклад:

- звичайний персоналізований пошук, що базується на вподобаннях користувача;
- засвоєння семантичних уподобань користувача впродовж деякого проміжку часу;
- динамічна контекстуалізація уподобань користувача;
- динамічна адаптація колаборативної фільтрації. [4]

Якщо ресурс передбачає взаємодію користувачів з різними групами продуктів, що не мають спільних характеристик, має сенс запровадити спільну базову модель, яку згодом можна буде розширити і конкретизувати для деякої групи продуктів. Для конкретизації використовуються онтології такої складності, якої потребує характеристика довільного продукту конкретної категорії. Ці онтології також використовуються для вираження уподобань користувача у певній категорії товарів. Таке застосування онтологій полегшує змогу:

- сортувати і фільтрувати результати пошуку відповідно до уподобань та інтересів користувачів;
- адаптувати взаємодію користувача з ресурсом, наприклад, змінивши інтерфейс відповідно до його особистості;
- налаштувати результати пошуку відповідно до контексту користувача;
- рекомендувати користувачеві комерційні пропозиції відповідно до його уподобань. [5]

1.5.1 Функції виміру семантичної схожості

Найпростішим типом онтології є дерево з кореневою нодою φ, яка представляє деяку загальну сутність. Структура дерева розгалужується нижче до більш конкретних сутностей, при цьому відношення між деякою даною нодою та батьківською — це відношення типу *is-a* (одностороннє). Формально, онтологія O — це кортеж (E, φ, p), де E — це набір сутностей, φ — коренева сутність $(\varphi \in E)$, p — батьківська функція $p: E \to E$ така, що будь-яка з сутностей онтології має нециклічний шлях до φ. Функція виміру семантичної схожості на O це функція, яка попарно визначає близькість двох довільних сутностей з онтології.

Якщо t і u це сутності онтології O, то більшість функцій виміру семантичної схожості використовують такі поняття:

- d: максимальна кількість кроків від будь-якої сутності до φ;
- m: максимальна кількість кроків між будь-якою парою сутностей;
- $depth_t$: кількість кроків від сутності t до φ;
- $path_{t,u}$: кількість кроків від сутності t до u;
- $LCS_{t,u}$: найбільш конкретний (розташований якомога далі від φ) спільний предок сутностей t і u;
- IC_t: інформаційний контент (інформаційне наповнення) сутності t. [6]

Інформаційний контент — це міра інформативності даної сутності в ієрархії, отримана зі статистики, зібраної з колекції записів. У випадку рекомендаційної системи, яка використовує онтологію, колекцією записів буде інформація про деякі об’єкти (користувачі та продукти), що їх характеристики виражені за допомогою сутностей онтології. Тоді інформаційний контент сутності t можна виразити як $IC_t = -log\ P(t)$, де $P(t)$ — це ймовірність появи t у колекції. Чим

менша ймовірність появи сутності, тим більшою вважається її інформативність, і навпаки.

Нижче наведено одні з найпоширеніших функцій виміру семантичної схожості.

$$Sim_{t,u}^{PATH} = \frac{1}{path_{t,u} + 1}$$

$$Sim_{t,u}^{LCH} = Max\left[-\log\left(\frac{path_{t,u}}{2d}\right)\right]$$

$$Sim_{t,u}^{WUP} = \frac{2 \cdot depth_{LCS_{t,u}}}{depth_t + depth_u}$$

$$Sim_{t,u}^{RES} = \frac{IC_{LCS_{t,u}}}{10}$$

$$Sim_{t,u}^{LIN} = \frac{2 \cdot IC_{LCS_{t,u}}}{IC_t + IC_u}$$

$Sim_{t,u}^{LCH}$ — функція Лікока і Ходорова, $Sim_{t,u}^{WUP}$ — функція Ву і Палмера, $Sim_{t,u}^{RES}$ — функція Резніка, $Sim_{t,u}^{LIN}$ — функція Ліна.

2 Розробка рекомендаційної системи з використанням онтології

Метою практичної частини курсової роботи була розробка рекомендаційної систему на навчальну тему, використовуючи деяку онтологію. Найпопулярнішими напрямками рекомендаційних системам на дану тематику є такі:

- вибір академічних дисциплін;
- ресурси для навчання;
- онлайн навчання;
- навчальні тренінги. [7]

Для виконання практичної частини було обрано напрямок «вибір академічних дисциплін», а отже продуктом в створеній рекомендаційній системі виступає деяка академічна дисципліна, і метою РС є запропонувати користувачу такі дисципліни, які його цікавлять. В якості даних було використано інформацію про деякі з курсів, розміщених в системі запису НаУКМА. За підходом реалізована рекомендаційна система належить до категорії фільтрації за змістом.

Результатом роботи став ресурс, що надає користувачам (наприклад, студентам або абітурієнтам) можливість переглядати дисципліни за пошуковими параметрами. Даний ресурс виконує такі функції:

- надає коротку інформацію про кожен з доступних в системі курсів;
- дозволяє фільтрувати курси за:
    - університетом;
    - факультетом;
    - спеціальністю;
    - темою;
- дозволяє шукати курси, що схожі за тематикою на деякий обраний курс;
- генерує випадковий курс з усіх доступних в системі.

Серед усіх засобів, що були використані для розробки застосунку хочеться виділити такі:

а) мова програмування R і середовище RStudio;

б) пакет RShiny для створення графічних інтерфейсів на мові R;

в) пакети Ontology Index і Ontology Similarity для роботи з онтологіями;

г) середовище Protégé;

д) СКБД Postgres (реляційну модель створеної бази даних можна переглянути в додатку А).

2.1 Використання онтології в РС

У даній реалізації рекомендаційної системи головною характеристикою дисципліни постає список академічних тем, яких стосується даний курс. Саме за допомогою таких тем користувачу надано можливість шукати схожі один на одного курси. Даний прийом, коли користувач обирає продукт, який його зацікавив, та отримує схожі продукти, було згадано у розділі 1.4. Академічні теми зберігаються у вигляді онтології.

Розроблена Educational Theme Ontology (ETO) налічує більше 70 тем, логічно розташованих у деревовидній структурі. Кореневою нодою виступає загальна тема *educational thematic entity*, на наступному рівні знаходяться три теми: *humanities*, *natural and formal sciences* та *social sciences*. Загальну структуру ETO зображено на рисунку 2.1.

Використання таких тем для характеризування дисциплін в даній рекомендаційній системі має такі переваги:

- кожен курс має принаймні одну конкретну тему, що надає курсу вузьку спеціалізацію;

- характеристика самого курсу не залежить від університету, факультету чи спеціальності курсу;
- таким чином стає реальним шукати максимально схожі або фактично еквівалентні курси між університетами;
- без використання тем курси, що викладаються для однакових спеціальностей на одному році навчання можна було б вважати схожими, але насправді їх наповнення може бути абсолютно різним:
    - наприклад, на першому курсі студенти усіх спеціальностей НаУКМА слухають такі курси як «Вступ до Могилянських студій» та «Українська мова», насправді, ці курси не мають спільних тем.

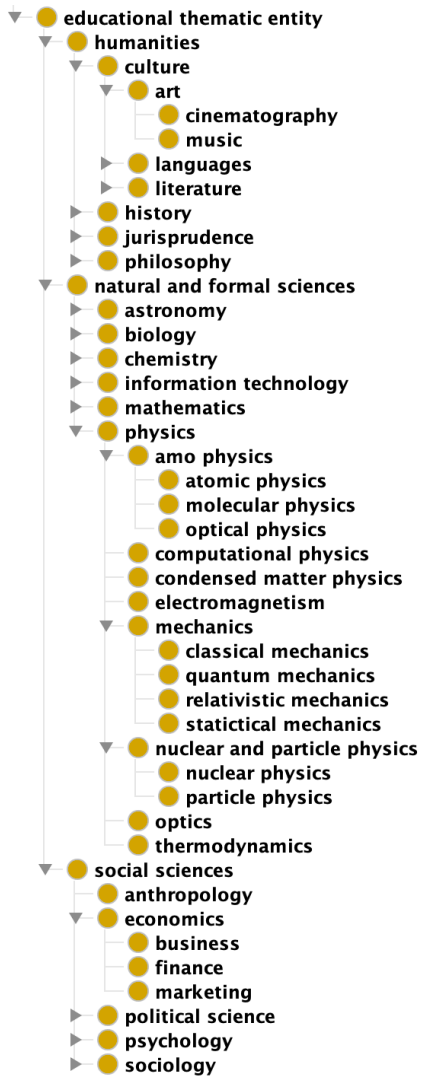


Рисунок 2.1 — Схема ЕТО

Для роботи з онтологією та визначення схожості дисциплін на основі схожості наборів тем було використано пакети мови R «OntologyIndex» та «OntologySimilarity».

Щоб зчитати онтологію, представлену у форматі OBO, в програму використовуємо метод `get_ontology` та автоматично генеруємо інформаційний контент для усіх сутностей онтології (для даної онтології цього достатньо) за допомогою методу `descendants_IC` (рисунок 2.2).

```
6
7  # Load ontology from file and generate IC
8  education_ont <- get_ontology("../educational-themes-obo.owl")
9  information_content <- descendants_IC(education_ont)
10
```

Рисунок 2.2 — Підготовка до роботи з онтологією

Для того, щоб розрахувати схожості деякого заданого курсу з усіма іншими, які містяться в базі даних, використовуємо функцію

```
get_profile_sims(profile, term_sets, ontology,
                 information_content),
```

де `profile` — це вектор ідентифікаторів усіх термів, що характеризують заданий курс, `term_sets` — масив векторів ідентифікаторів усіх термів, що характеризують кожен курс в системі, `ontology` — задана онтологія, `information_content` — інформаційний контент усіх термів онтології. [8] Одним з опціональних аргументів функції є `term_sim_method`, тобто функція виміру семантичної схожості, доступні функції — Ліна і Резника (за замовченням використовується функція Ліна), описані у розділі 1.5.1. Функцію, що розраховує значення схожості заданого по ID курсу з усіма курсами в системі, зображено на рисунку 2.2. Повний код з усіма методами, що стосуються операцій з онтологією, викладено у додатку Б.

```
27
28  # Function to get similarity values for {course_id} profile
29  get_course_similarity <- function(course_id) {
30    for (name in names(course_themes_lists)) {
31      if (name == course_id) {
32        course_of_interest <- course_themes_lists[[name]]
33
34        similarity = get_profile_sims(
35          profile = course_of_interest,
36          term_sets = course_themes_lists,
37          education_ont,
38          information_content)
39        return(similarity)
40      }
41    }
42  }
```

Рисунок 2.*3* — Розрахунок схожості курсів; позначення: `course_theme_lists` - список з темами для кожного курсу

2.2 Огляд функціоналу і користувацького інтерфейсу

Для реалізації графічного інтерфейсу користувача було використано пакет мови R «RShiny», що дозволяє з легкістю динамічно змінювати на сторінці вивід, якщо користувач змінює параметри запиту, за допомогою реактивних змінних та так званого спостерігання на змінами. Пакет надає інструментарій, за допомогою якого можна створювати текстові поля, поля для вибору з деякого набору значень, кнопки, відображення таблиць, графіків, діаграм, карт та інших видів графічних елементів, які зручно використовувати для відображення статистичних даних. [9]

В результаті розробки було створено простий і зрозумілий інтерфейс, який зображено на рисунку 2.3. Зліва розташовані поля для фільтрування курсів за темою, університетом, факультетом та спеціальністю і кнопка для генерації довільного курсу, а справа — основна частина з базовою інформацією про кожен курс та кнопкою для знаходження схожих курсів. У даній імплементації програми були використані тільки дисципліни НаУКМА, отже значення університету за замовченням визначено відповідним чином.

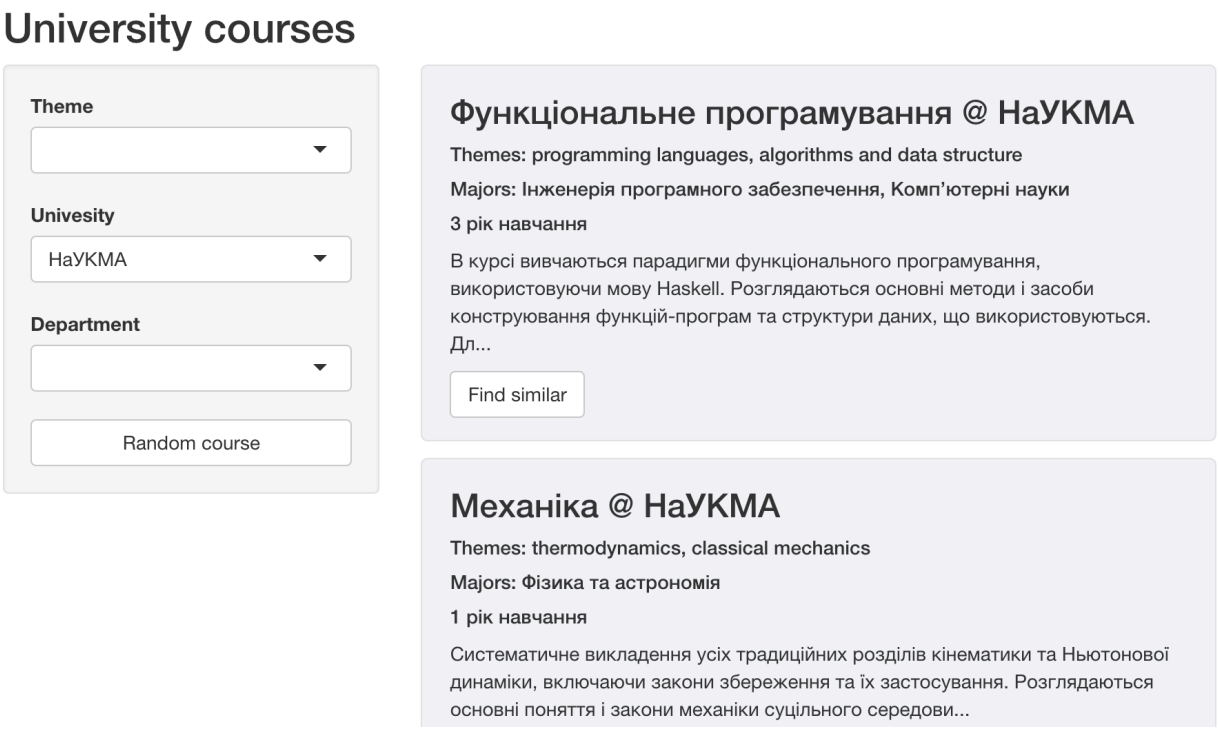


Рисунок 2.*4* — Початковий вигляд програми при першому запуску

Якщо користувач бажає відфільтрувати курси за факультетом, він може це зробити після вибору університету, бо інакше вибір факультету буде недоступним. Аналогічно, перед вибором спеціальності необхідно обрати відповідний факультет. Приклад фільтрації за спеціальністю «Фізика і астрономія» можна побачити на рисунку 2.4.

Фільтрація за темою відбувається разом з фільтрацією за іншими полями. Це означає, що якщо користувач обере бажану спеціальність (або факультет чи університет) разом із темою, то курси буде відфільтровано за обома значеннями одночасно і буде показано курси за обраною темою, що викладаються для обраної спеціальності. При виборі теми у списку запропонованих тем доступні не лише конкретні теми (листки дерева), що безпосередньо можуть бути присвоєні дисципліні як характеристика, а взагалі усі теми онтології (окрім кореневої). Тоді при виборі теми, яка містить в собі поділ на більш конкретні,

фільтрація буде відбуватися за усіма конкретними темами, що належать даній темі. Приклад фільтрації за темою зображено на рисунку 2.5.

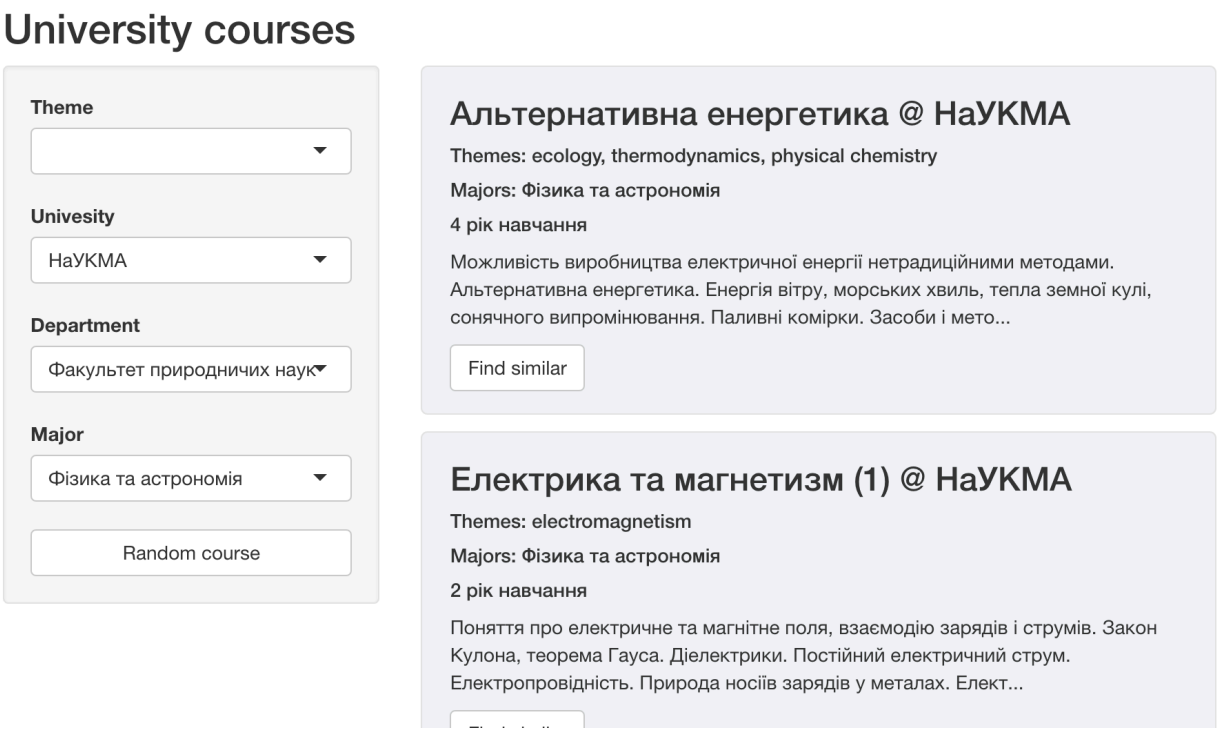


Рисунок 2.5 — Фільтрація за спеціальністю

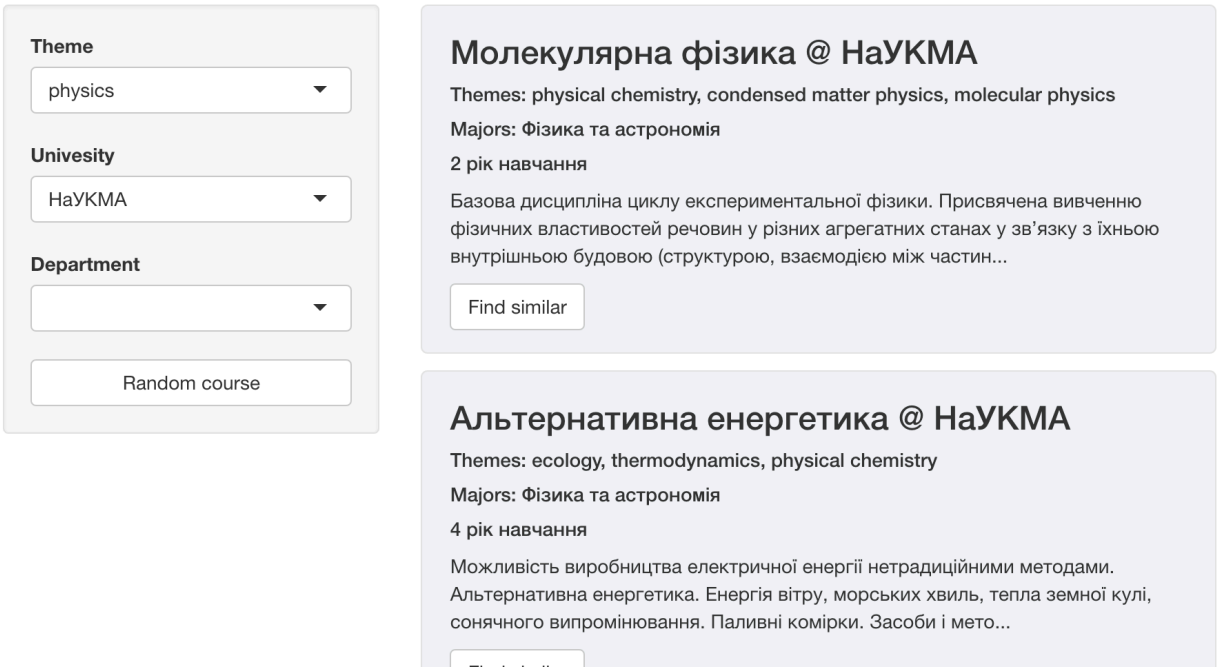


Рисунок 2.6 — Фільтрація за темою

Застосунок реагує на задані користувачем зміни за допомогою переглядачів. Переглядач схожий за властивостями на реактивні змінні — він буде запускатися щоразу, коли змінні, від яких він залежить, будуть змінені. Таким чином для відслідковування встановлення користувачем нових значень в полях для фільтрації використовується метод, зображений на рисунку 2.7. Відслідковуємо зміни в полях вибору значення, якщо значення хоча б одного з них змінилося, встановлюємо режим фільтрації (допоміжна функція) і накладаємо фільтри на базовий data-frame зі значеннями. В свою чергу зміна значень в data-frame запускає зміну відображення даних на екрані.

```
50      # Handle changes in filters
51      observe({
52          theme = input$theme_select
53          uni = input$uni_select
54          dep = input$dep_select
55          maj = input$maj_select
56
57          set_filter_mode()
58          apply_filters(theme, uni, dep, maj)
59      })
```

Рисунок 2.7 — Обробка зміни значень фільтрів

Для того, щоб показати схожі на деяку дисципліну курси, необхідно натиснути на кнопку «Find similar» на бажаному курсі. Приклад, зображений на рисунку 2.8, ілюструє результат пошуку курсів, що схожі на «Функціональне програмування». Обраний курс показано першим, далі розташовано схожі курси, перший з них має показник близькості з обраним 0.75918387. Результати відсортовано за спаданням схожості.

Для фіксування кліків на кнопки «Find similar» при кожній новій генерацій курсів до усіх щойно створеної кнопок додаємо переглядач застосовуючи метод, зображений на рисунку 2.9.

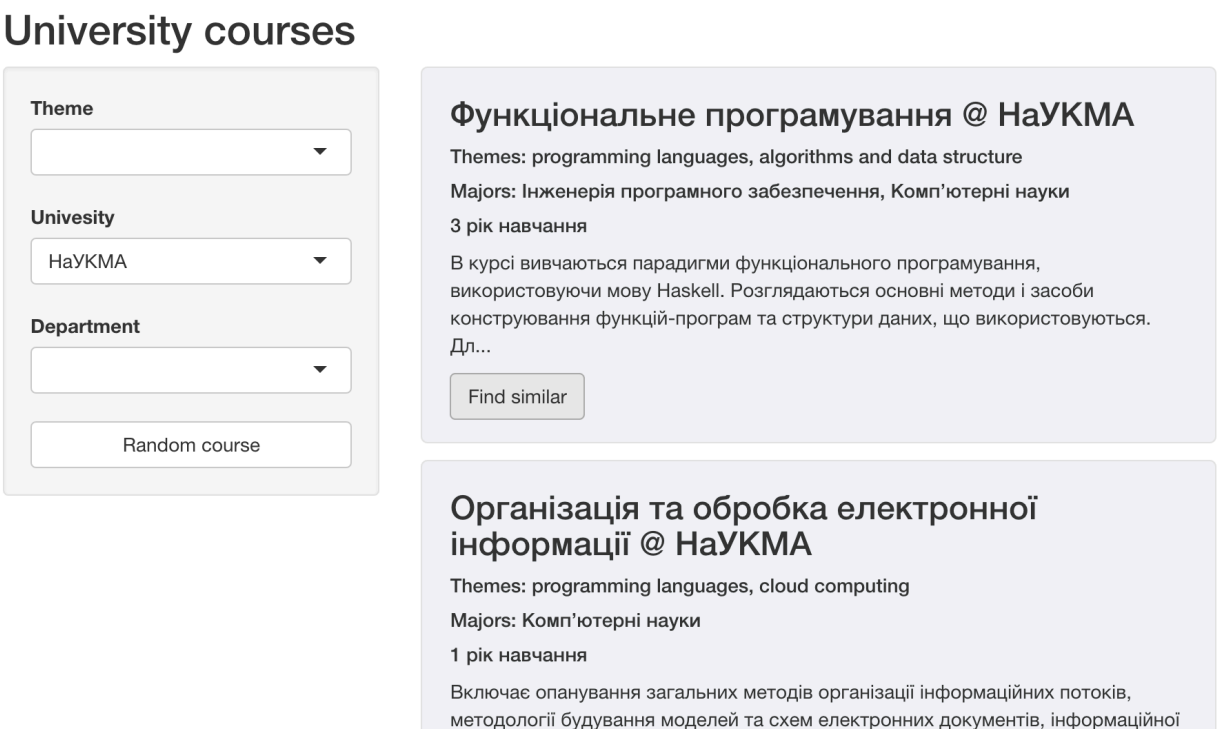


Рисунок 2.8 — Пошук схожих дисциплін

```
67      # Register and handle Show similar button clicks
68      observe({
69          input_btn <- paste0("fs_btn_", ids())
70          lapply(input_btn, function(x) {
71              observeEvent(
72                  input[[x]],
73                  {
74                      finish_filter_mode()
75                      course_id = as.numeric(sub("fs_btn_", "", x))
76                      set_similar(course_id)
77                  }
78              )
79          })
80      })
```

Рисунок 2.9 — Фіксування та обробка кліків на кнопки «Show similar»

Для того, щоб отримати інформацію про будь-який випадковий курс, достатньо натиснути кнопку «Random course». Приклад такої операції зображено на рисунку 2.10, а функцію для реєстрації кліків на кнопку наведено на рисунку 2.11.

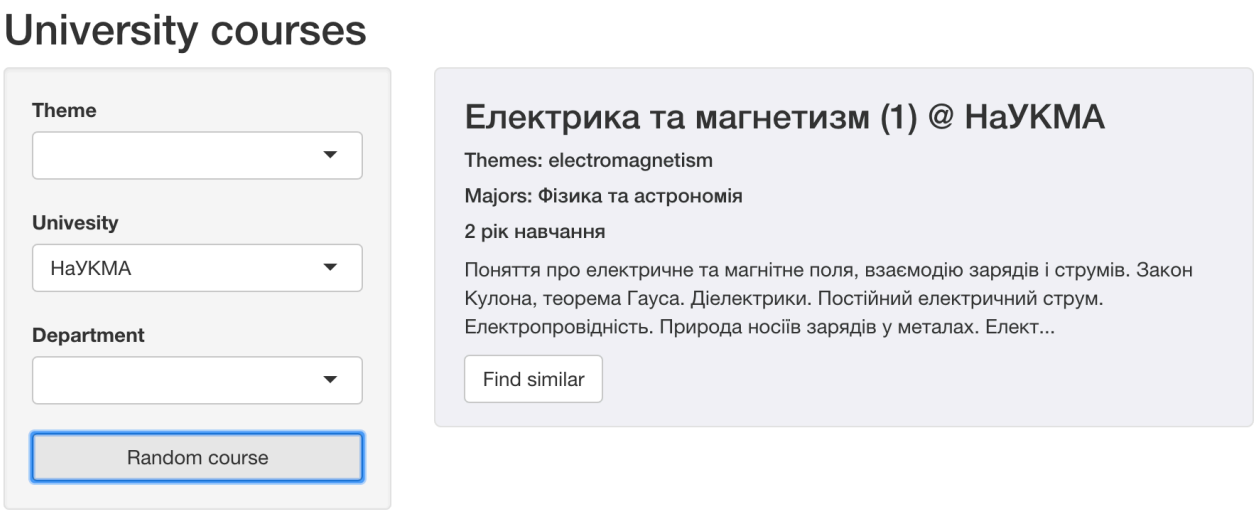


Рисунок 2.10 — Генерація випадкового курсу

```
60
61      # Handle Random button click
62      observeEvent(input$random_btn, {
63          finish_filter_mode()
64          set_random()
65      })
66
```

Рисунок 2.11 — Обробка кліків на кнопку «Random course»

Можливі покращення роботи ресурсу:

- використання деякого виду персоналізації результатів на основі даних про користувача, використовуючи дані сесії;
- розширення бази дисциплін та набору навчальних тем в онтології;
- впровадження в якості характеристики курсу «головних» та «другорядних» тем, що підвищить точність характеризації курсів та покращить точність рекомендацій.

Висновки

В результаті написання курсової роботи було досліджено актуальність та області застосування рекомендаційних систем, зокрема навчальних, розглянуто три основні парадигми проектування рекомендаційних систем, такі як:

- колаборативна фільтрація (є універсальними для будь-яких РС, адже спираються на взаємодію користувачів і продуктів, а не їх характеристики);
- фільтрація за змістом (допомагає уникнути проблеми «холодного старту», адже використовує для побудови моделей дані про користувачів або продуктів і не потерпає від нестачі даних про взаємодію);
- фільтрація на основі знань (використовується для реалізації більш персоналізованих та досконалих ресурсів, де користувач активно задає пошукові запити власноруч).

Було досліджено ключові випадки та способи використання онтологій в рекомендаційних системах, зокрема найпоширеніші функції виміру семантичної схожості, що використовуються для пошуку схожих об’єктів.

В результаті виконання практичної частини курсової роботи, було розроблено ресурс, де відображено інформацію про дисципліни, які відповідають пошуковим запитам користувача, які він задав за допомогою полів для фільтрації або за допомогою зазначення курсу, який його цікавить. Розрахунок схожості двох курсів відбувається на основі тем, які характеризують кожен з курсів та зберігаються у вигляді онтології, за допомогою бібліотеки «OntologySimilarity».

Література

[1] H. Mohana, «A Study on Ontology Based Collaborative Filtering Recommendation Algorithms in E-Commerce Applications», 2017.

[2] B. Rocca, «Introduction to recommender systems», 2019. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

[3] J. Wu, «Knowledge-Based Recommender Systems: An Overview», 2019. Available: https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba.

[4] K. Rabahallah, «MOOCs Recommender System using Ontology and Memory-based Collaborative Filtering», Algiers, 2018.

[5] A. C. M. Costa, «COReS: Context-aware, Ontology-based Recommender system for Service recommendation», Trento-Povo.

[6] M. O. Paul Sheridan, «An Ontology-Based Recommender System with an Application to the Star Trek Television Franchise», 2019.

[7] A. C. Rivera, «Recommendation Systems in Education: A Systematic Mapping Study», Guayaquil, 2018.

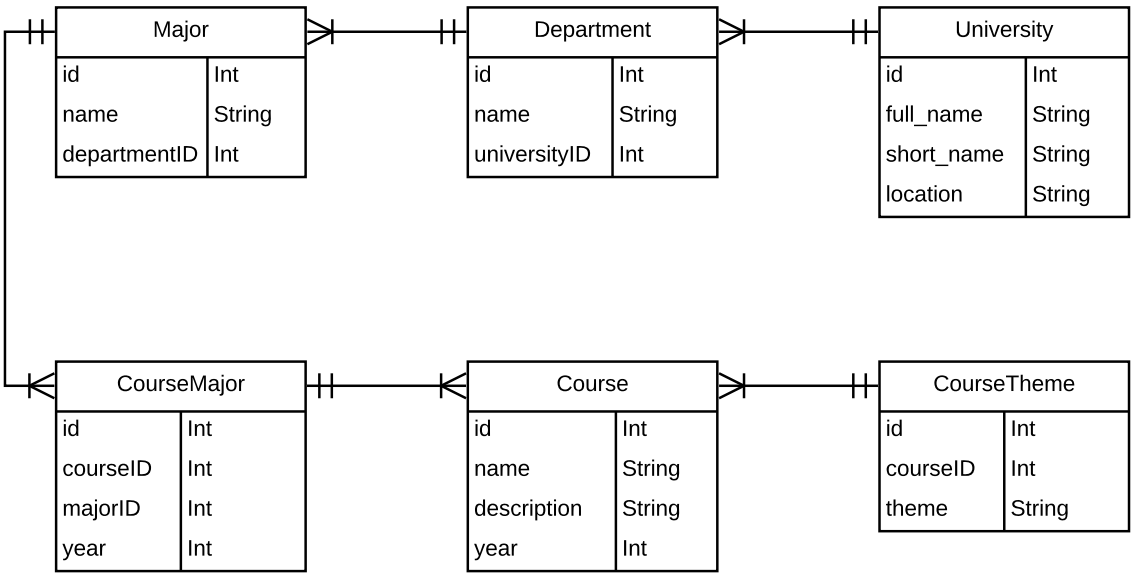
[8] D. Greene, «ontologySimilarity: Functions for Calculating Ontological Similarities», 2019. Available: https://cran.r-project.org/web/packages/ontologySimilarity/index.html.

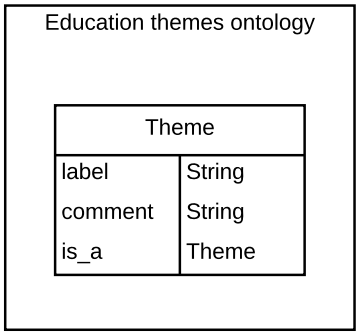
[9] W. Chang, «shiny: Web Application Framework for R», 2020. Available: https://cran.r-project.org/web/packages/shiny/index.html.

[10] S. S. Jeremy Debattista, «Ontology-based Rules for Recommender Systems», Galway.

Додаток А.

Реляційна модель використаної бази даних та умовне зображення даних онтології





Додаток Б.

R файл з операціями з онтологіями

```
# Load libs
library(ontologyIndex)
library(ontologySimilarity)
library(dplyr)

# Load ontology from file and generate IC
education_ont <- get_ontology("../educational-themes-obo.owl")
information_content <- descendants_IC(education_ont)

# Get all themes
themes <- get_descendants(
    education_ont,
    roots="educational-thematic-entity",
    exclude_roots = TRUE
  )

# Function to beautify and sort theme names
get_beautiful_themes <- function() {
  theme_names <- gsub("-", " ", themes)
  sorted <- sort(theme_names)
  return(sorted)
}

# Function to get similarity values for {course_id} profile
get_course_similarity <- function(course_id) {
  for (name in names(course_themes_lists)) {
    if (name == course_id) {
      course_of_interest <- course_themes_lists[[name]]

      similarity = get_profile_sims(
        profile = course_of_interest,
        term_sets = course_themes_lists,
        education_ont,
```

```
        information_content)
      print(similarity)
      return(similarity)
    }
  }
}

# Function to get top similar courses (with all info) for course
get_top_sim_course_ids <- function(course_id) {
  similarity = get_course_similarity(course_id)
  ordered_vec = similarity[order(similarity, decreasing = TRUE)]
  top_courses = names(ordered_vec[ordered_vec > 0.5])
  return(top_courses)
}

# Function to get course by theme (considering descendants)
get_by_theme <- function(theme_name, table) {
  descendants_names = get_descendants(
           education_ont,
           roots = theme_name
      )
  theme_courses = df_course_theme[df_course_theme$theme %in%
descendants_names, ]
  course_ids = unique(theme_courses$course_id)
  return(table[table$course_id %in% course_ids, ])
}
```