КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Кваліфікаційна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз»

за спеціальністю 124 Системний аналіз

на тему:

РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ В СФЕРІ ПРАЦЕВЛАШТУВАННЯ НА ОСНОВІ ОГОЛОШЕНЬ ПРО ВАКАНСІЇ

Виконав студент 4-го курсу
Павло САХНЮК _____

Науковий керівник:
кандидат фізико-математичних наук
Юлія ШЕВЧУК _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Роботу розглянуто й допущено до захисту на засіданні кафедри системного аналізу та теорії прийняття рішень

«07» 06 2022 р., протокол № 10

Завідувач кафедри
Олександр НАКОНЕЧНИЙ

Київ -2022

Зміст

Вступ

З початком ери Web 2.0 Інтернет почав рости та розвиватися з величезною швидкістю. З’явилося багато можливостей, таких як обмін знаннями, інформацією, думками з іншими користувачами. Це сприяло розвитку таких соціальних мереж, як Facebook. Сьогодні автори можуть ділитися своїми творами з мільйонами читачів по всьому світу. Музиканти-аматори можуть стати відомими швидше, ніж будь-коли, просто завантаживши свої треки. Діловий світ знайшов більше клієнтів і прибутку в Інтернеті. В Інтернеті з'явилося безліч інтернет-магазинів, аукціонів або блошиних ринків.

Сьогодні кожен користувач всесвітньої павутини може придбати практично будь-який товар, перебуваючи в будь-якій країні світу. На відміну від справжніх магазинів, в Інтернеті немає обмежень на місця. Насправді місце тут майже нескінченне. Проте люди зіткнулися з новою проблемою у WWW. Обсяг інформації та предметів став надзвичайно великим, що призвело до інформаційного перевантаження. Знайти те, що насправді шукає користувач, стало великою проблемою. Пошукові системи частково вирішили цю проблему, однак персоналізації інформації не було. Тож розробники знайшли рішення в рекомендаційних системах. Системи рекомендацій – це інструменти для фільтрації та сортування елементів та інформації. Вони використовують думки спільноти користувачів, щоб допомогти людям у цій спільноті ефективніше визначити вміст, що цікавить, із потенційно величезного набору варіантів.

Поява на ринку мобільних пристроїв з новими технологіями, такими як стандарти GPS і 3G, поставило нові виклики. Рекомендаційні системи залучилися до розвитку туристичної, безпекової та інших сфер. Сучасні системи рекомендацій підвищують точність своїх рекомендацій за допомогою контекстно-залежних, семантичних та інших підходів. Сьогодні рекомендації більш конкретні та персоналізовані. Проблеми поєднання різних технологій та

рекомендацій підходів для кращих результатів будуть існувати завжди і будуть причиною нових досліджень.

В роботі розглядається розробка рекомендаційної системи, яка була б застосована в сфері працевлаштування. В її основі лежить метод аналізу контенту. В якості контенту використовувався опис вакансій..

Кваліфікаційна робота складається зі вступу, трьох розділів, висновку, списку використаної літератури. Список використаної літератури включає в себе 23 джерела. Робота виконана на 43 сторінках друкованого тексту.

У першому розділі розглянуто поняття рекомендаційних систем, проаналізовано основні їх цілі. Розібрані категорії рекомендаційних систем та принципи їх роботи.

У другому розділі розібрані методи обробки інформації. Розібраний детально кожний метод та показано приклад.

У третьому розділі повністю розписується розробка конкретної рекомендаційної системи. Вказані методи, які використовувались та як вони були використані при програмній реалізації. Також, присутній опис веб-порталу, який розроблявся спеціально для цієї рекомендаційної системи.

1. Рекомендаційні системи

Більшість користувачів Інтернету так чи інакше стикалися з системою рекомендацій. Уявимо, наприклад, що ви нещодавно пройшли курси з програмування, а потім ви відвідуєте сайт з пошуком роботи. Після введення ключових слів (наприклад Developer) знайдуться вакансії, які мають в назві або в описі ці слова. При переході на конкретну вакансію, в одній частині веб-сторінки, яка, можливо, називається «Схожі вакансії», відображається список додаткових вакансій, які ніби вас цікавлять. Якщо ви звичайний користувач того самого сервісу з пошуку роботи, такий персоналізований список рекомендацій з’явиться автоматично, щойно ви зайдете на сайт. Програмна система, яка визначає, які вакансії слід показати конкретному відвідувачеві, є системою рекомендацій.

Як правило, рекомендаційні системи працюють через три фази (етапи). Фази наступні:

Фаза моделювання: цей етап зосереджується на підготовці даних, які будуть використані на наступних двох фазах. Для цього є три випадки, перший — це створення матриці рейтингів, яка містить користувачів як записи, елементи як атрибути, а значення клітинки кожної матриці — це оцінка, зроблена користувачем для певного елемента. По-друге, створення профілю користувача — це здебільшого вектор для кожного користувача, який пояснює його уподобання щодо елемента в цілому або щодо деяких аспектів елемента. По-третє, створення профілю предмета, який містить особливості конкретного предмета.

Фаза передбачення: цей етап має на меті передбачити рейтинг або оцінку небачених/невідомих елементів для конкретного користувача за допомогою функції корисності залежно від вилученої інформації під час фази моделювання.

Фаза рекомендацій: ця фаза є розширенням фази прогнозування, де застосовуються різні підходи для підтримки рішення користувача шляхом фільтрації найбільш підходящих елементів. Він рекомендує/пропонує користувачеві нові елементи (тобто набір топ-N предметів з найвищими прогнозованими оцінками), які, швидше за все, привабливі для нього.[18]

Рекомендаційна система може принести користь користувачам, веб-сайтам електронної комерції та багатьом онлайн-компаніям. Зараз, з величезним поширенням онлайн-покупок, багато користувачів покладаються на рекомендаційні системи для отримання персоналізованих рекомендацій і мінімізації величезних транзакцій для вибору товарів. Зазвичай користувачі використовують цей тип програми, щоб отримати певну послугу, як-от бронювання готелю/квитка/ресторану, придбання продукту, перегляд фільму тощо. Очевидно, що користувачі, які споживали певний продукт генерують кілька типів відгуків, які висловлюють свою думку про цей продукт. Користувачі дають оцінки для всього продукту (тобто шкала оцінок від 1 до 5), або для певної характеристики продукту (наприклад, комфорт, ціна тощо).

Крім того, користувачі можуть писати відгуки, окрім рейтингів. Цілком очевидно, що ці оцінки впливатимуть на думку користувача та спонукатимуть його купувати чи не купувати продукт. На додаток до цінних відгуків, вона також надає користувачам рекомендації на основі деяких методів. Рекомендаційна система може рекомендувати продукти з високими рейтингами, або рекомендувати продукти, які мають схожі характеристики з цією маскою для обличчя, або рекомендувати продукти на основі історії перегляду користувача.[18]

1.1. Цілі рекомендаційних систем

Основна мета рекомендаційних систем - вирішити проблему перевантаження інформацією через експоненційне зростання потоку інформації в Інтернеті, щоб надати користувачам/клієнтам різні джерела про

послуги, такі як продукти, готелі та ресторани. Для початку, сформулюємо дві головні категорії рекомендаційних задач:

1. **Прогнозована версія задачі.**

   Перший підхід полягає в тому, щоб передбачити значення рейтингу для комбінації користувач-елемент. Передбачається, що навчальні дані доступні та вказують на переваги користувача для елементів. Для m користувачів і n елементів це відповідає неповній матриці m×n, де вказані (або спостережувані) значення використовуються для навчання. Відсутні (або неспостережені) значення прогнозуються за допомогою цієї навчальної моделі. Цю задачу також називають проблемою заповнення матриці, оскільки у нас є неповністю задана матриця значень, а решта значень прогнозуються за допомогою алгоритму навчання.

2. **Рейтинговий варіант задачі**.

   На практиці не потрібно передбачати оцінки користувачів для конкретних елементів, щоб давати рекомендації користувачам. Швидше, продавець може порекомендувати найпопулярніші товари для конкретного користувача або визначити найпопулярніших користувачів для націлювання на певний товар. Визначення топ-k елементів є більш поширеним, ніж визначення топ-k користувачів, хоча методи в обох випадках точно аналогічні. [1]

Вирішення другої задачі важливо для постачальника послуг для досягнення наступних цілей:

1. **Збільшення кількості проданих товарів.**

   Це, мабуть, найважливіша функція для комерційної рекомендаційної системи, тобто можливість продавати додатковий набір товарів у порівнянні з тими, які зазвичай продаються без будь-яких рекомендацій. Ця ціль досягається тому, що рекомендовані елементи, ймовірно, відповідають потребам і бажанням користувача. Ймовірно,

користувач усвідомить це після того, як спробував кілька рекомендацій. Некомерційні програми мають подібні цілі, навіть якщо для користувача немає жодних витрат, пов’язаних з вибором елемента. Наприклад, тематична мережа прагне збільшити кількість вакансій, які переглядаються на своєму сайті. Загалом, можна сказати, що з точки зору постачальника послуг, головною метою впровадження рекомендаційної системи є підвищення коефіцієнта конверсії, тобто кількості користувачів, які приймають рекомендацію та споживають товар, порівняно з кількістю простих відвідувачів, які просто переглядають інформацію.

2. **Підвищити задоволеність користувачів.**

Добре розроблена рекомендаційна система також може покращити досвід роботи користувача з сайтом або програмою. Користувач знайдуть рекомендації цікавими, доречними, а при правильно розробленій взаємодії людини та комп’ютера їй також сподобається користуватися системою. Поєднання ефективних, тобто точних, рекомендацій та зручного інтерфейсу підвищить суб’єктивну оцінку системи користувачем. Це, у свою чергу, збільшить використання системи та збільшить ймовірність того, що рекомендації будуть прийняті

3. **Підвищити надійність користувача.**

Користувач повинен бути лояльним до веб-сайту, який при відвідуванні впізнає старого клієнта і ставиться до нього як до цінного відвідувача. Це нормальна особливість рекомендаційної системи, оскільки багато рекомендаційних систем обчислюють рекомендації, використовуючи інформацію, отриману від користувача під час попередніх взаємодій, наприклад, її оцінки елементів. Отже, чим довше користувач взаємодіє із сайтом, тим більш досконалою стає її модель користувача, тобто системне представлення уподобань користувача, і

тим більше можна ефективно налаштувати вихід рекомендацій, щоб він відповідав уподобанням користувача.

4. **Краще зрозуміти, чого хоче користувач.**

   Іншою важливою ціллю рекомендаційної системи, яку можна використовувати в багатьох інших програмах, є опис уподобань користувача, зібраних явно або передбачених системою. Потім постачальник послуг може вирішити повторно використати ці знання для низки інших цілей, таких як покращення управління запасами або виробництвом товару. [1]

Існує велика різноманітність видів продукції, рекомендованих такими системами. Деякі системи рекомендацій, наприклад Facebook, не рекомендують продукти безпосередньо. Натомість вони можуть порекомендувати соціальні зв'язки, які мають непряму користь для сайту, підвищуючи його зручність використання та прибуток від реклами.

1.2. Класифікація рекомендаційних систем

Основні моделі для рекомендаційних систем працюють з 2 типами даних, які включають взаємодію користувача з елементом, наприклад рейтинги або купівельну поведінку, а також інформацію про атрибути користувачів і елементів, наприклад текстові профілі або відповідні ключові слова. [14] Залежно від типу даних рекомендаційні системи можна поділити на 4 головні класи:

1. Рекомендаційна система на основі контенту (Content-Based Recommender System): підхід, що ґрунтується на вмісті, виробляє відповідні рекомендації для користувача на основі його нещодавньої поведінки відповідно до того, що користувачеві сподобалося, купив або дивився
2. Рекомендаційна система на основі знань (Knowledge-Based Recommender System): дає рекомендації на основі не історії оцінок користувача, а конкретних запитів користувача.

3. Рекомендаційна система з колаборативною фільтрацією (Collaborative filtering Recommender System): генерує рекомендацію для користувача на основі подібності серед користувачів, які в минулому мали схожі з ним уподобання/інтереси.
4. Гібридна рекомендаційна система (Hybrid Recommender System): поєднує в собі всі зазначені вище системи разом.

Повний опис всіх класів розберемо в наступному розділі.

1.2.1. Рекомендаційна система на основі контенту

У методах на основі вмісту описи елементів, позначені оцінками, використовуються як навчальні дані для створення специфічної для користувача задачі класифікації або регресійного моделювання. Для кожного користувача навчальні документи відповідають описам товарів, які вона купила або оцінила. Змінна класу (або залежна) відповідає вказаним рейтингам або купівельній поведінці. Ці навчальні документи використовуються для створення моделі класифікації або регресії, яка є специфічною для поточного користувача (або активного користувача). Ця специфічна для користувача модель використовується, щоб передбачити, чи сподобається відповідній особі товар, для якого її рейтинг або купівельна поведінка невідомі.

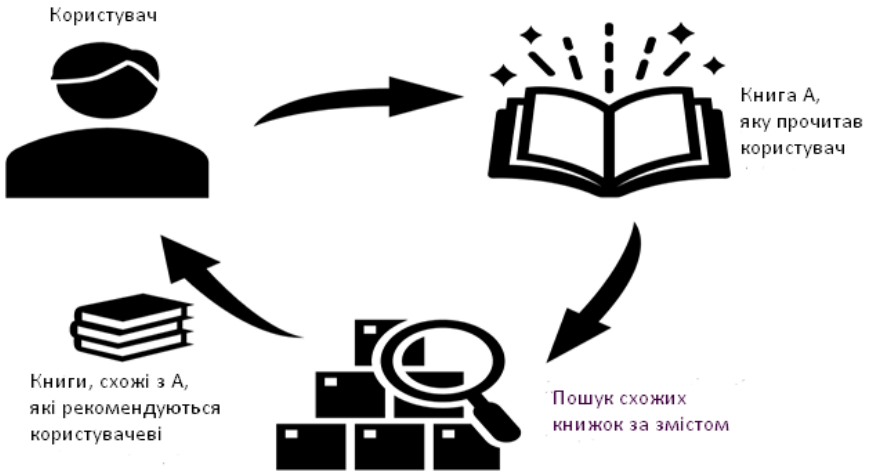

Рис. 1 - Проста ілюстрація рекомендаційної системи на основі контенту

На самому базовому рівні системи, що базуються на вмісті, залежать від двох джерел даних:

1. Першим джерелом даних є опис різних елементів з точки зору змістовних атрибутів. Прикладом такого представлення може бути текстовий опис товару від виробника.
2. Другим джерелом даних є профіль користувача, який формується на основі відгуків користувачів про різні елементи. Відгуки користувачів можуть бути явними або неявними. Відвертий відгук може відповідати оцінкам, тоді як неявний відгук може відповідати діям користувача. Рейтинги збираються подібним чином до систем співпраці.[15]

У системах рекомендацій на основі контенту описові атрибути елементів використовуються для створення рекомендацій. Термін «контент» або «вміст» відноситься до цих описів. У методах на основі вмісту оцінки та купівельна поведінка користувачів поєднуються з інформацією про вміст, доступною в елементах. Наприклад, розглянемо ситуацію, коли на онлайн сервісу з пошуку роботи, користувач розглядає певну вакансію на посаду менеджера в певній організації, але ми не можемо дізнатися, хто з інших користувачів відкликався на цю вакансію, щоб порадити вакансії, на які також відкликались ці користувачі. Однак, опис вакансії містить ключові слова, схожі на описи вакансій на цю ж посаду в інших компаніях. Тому ці вакансії можна порекомендувати користувачеві.

Розглянемо детальніше. Нехай користувач розглядає дану вакансію, у якої такі ключові слова та значення TF-IDF (опис цієї моделі буде в наступних розділах):

	manage	develop	...	assist
Vacancy 1	0,0026	0	...	0,08

Рис. 2 - Приклад вакансії

Рекомендаційна система повинна обрати найбільш схожу вакансію за даною із списку всіх вакансій

	manage	develop	...	assist
Vacancy 2	0	0,033	...	0,0006
Vacancy 3	0,02	0,03	...	0,05
Vacancy 4	0,01	0,06	...	0

Рис. 3 - Приклад інших вакансій

Існують різні алгоритми вимірювання подібності між елементами в базі даних і в профілі користувача. Одним з таких підходів є косинусна подібність (cosine similarity). Представляючи елементи у вигляді векторів на координатному просторі, він вимірює кути між векторами та видає їх значення косинуса. Вектори w_cі w_s двох елементів з атрибутами порівнюються у функції подібності косинуса наступним чином:

$$u(c,s) = cos(\vec{w_c}, \vec{w_s}) = \frac{\vec{w_c} \cdot \vec{w_s}}{||\vec{w_c}|| \times ||\vec{w_s}||} =$$

$$= \frac{\sum_{i=1}^{K} \vec{w_{ic}} \vec{w_{is}}}{\sqrt{\sum_{i=1}^{K} \vec{w_{ic}^2}} \sqrt{\sum_{i=1}^{K} \vec{w_{is}^2}}}$$

Чим більше подібні два елементи, тим менший кут між їх векторами:

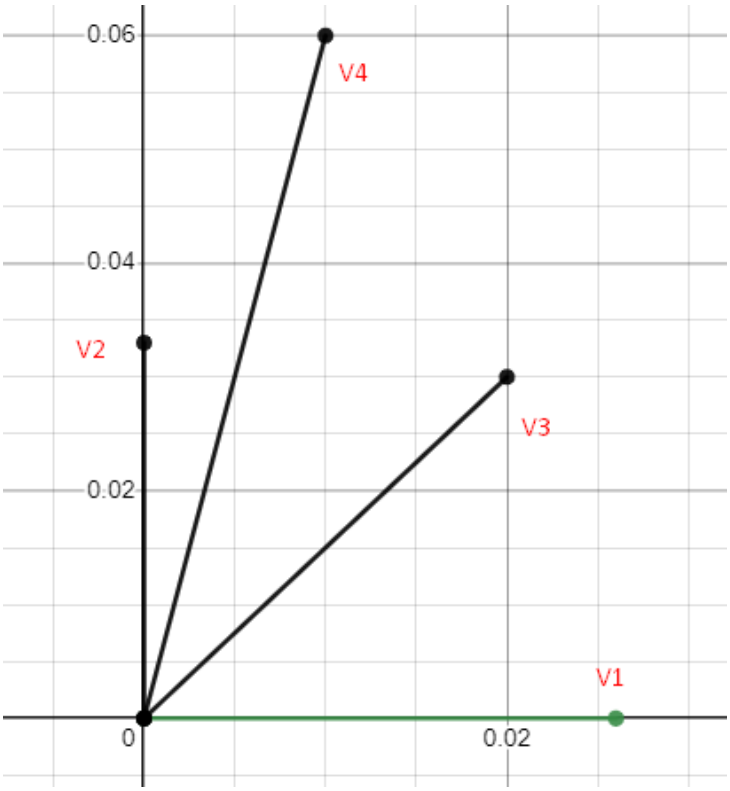

Рис. 4 - Вакансії у вигляді векторів

Пошук схожості вимагає детальної інформації про елементи. Краще описані предмети призводять до більш точних рекомендацій.[2]

Переваги системи на основі контенту:

1. Модель не потребує будь-яких даних про інших користувачів, оскільки рекомендації є специфічними для цього користувача. Це полегшує масштабування для великої кількості користувачів.
2. Модель може охопити конкретні інтереси користувача та рекомендувати елементи, які цікавлять небагатьох інших користувачів.

Недоліки:

1. Модель може давати лише рекомендації на основі наявних інтересів користувача. Іншими словами, модель має обмежені можливості для розширення наявних інтересів користувачів.

2. Великі розміри елементів вважаються основною проблемою, оскільки, коли рекомендація зроблена, вміст кожного елемента має бути перевірено, щоб виявити елементи, які, швидше за все, пов’язані з інтересами користувача.

1.2.2. Рекомендаційна система на основі знань

Рекомендаційні системи на основі знань особливо корисні в контексті товарів, які купуються не дуже часто. Приклади включають такі предмети, як нерухомість, автомобілі, туристичні запити, фінансові послуги або дорогі предмети розкоші. У таких випадках для процесу рекомендації може бути недоступним достатній рейтинг. Оскільки предмети купують рідко і з різними типами детальних опцій, важко отримати достатню кількість оцінок для конкретного екземпляра (тобто комбінації варіантів) даного предмета. Ця проблема також зустрічається в контексті проблеми холодного запуску, коли для процесу рекомендації не доступні достатні оцінки. Крім того, характер споживчих переваг може змінюватися з часом при роботі з такими предметами. Наприклад, модель автомобіля може суттєво розвиватися протягом кількох років, в результаті чого уподобання можуть показати відповідну еволюцію. В інших випадках може бути важко повністю охопити інтерес користувачів за допомогою історичних даних, таких як рейтинги. Конкретний елемент може мати атрибути, пов’язані з ним, які відповідають його різним властивостям, і користувача можуть цікавити лише елементи з певними властивостями.

Наприклад, автомобілі можуть мати кілька марок, моделей, кольорів, варіантів двигунів та варіантів інтер’єру, а інтереси користувачів можуть регулюватися дуже специфічною комбінацією цих параметрів. Таким чином, у цих випадках домен елемента, як правило, є складним з точки зору його різноманітних властивостей, і важко пов’язати достатні оцінки з великою кількістю наявних комбінацій.

Такі випадки можна вирішувати за допомогою рекомендаційних систем на основі знань, у яких рейтинги не використовуються для рекомендацій. Швидше, процес рекомендацій виконується на основі подібності між вимогами замовника та описами товарів або використання обмежень, що визначають вимоги користувачів. Процес полегшується за допомогою баз знань, які містять дані про правила та функції подібності для використання під час процесу пошуку. Насправді, бази знань настільки важливі для ефективного функціонування цих методів, що підхід отримав свою назву від цього факту. Явна специфікація вимог призводить до більшого контролю користувачів над процесом рекомендацій. Як у спільних системах, так і в системах, що базуються на вмісті, рекомендації вирішуються виключно на основі минулих дій/оцінок користувача, дій/оцінок його однолітків або комбінації обох. Системи, засновані на знаннях, унікальні тим, що вони дозволяють користувачам явно вказувати, що вони хочуть.

Системи рекомендацій, засновані на знаннях, можна класифікувати на основі типу інтерфейсу (і відповідних знань), що використовуються для досягнення вищезгаданих цілей:

1. Системи рекомендацій на основі обмежень: у системах на основі обмежень користувачі зазвичай вказують вимоги або обмеження (наприклад, нижню або верхню межі) для атрибутів елемента. Доменні правила використовуються для відповідності вимог користувачам атрибутам елемента. Ці правила представляють предметні знання, які використовуються системою. Такі правила можуть мати форму специфічних для домену обмежень щодо атрибутів елемента (наприклад, «Автомобілі до 1970 року не мають круїз-контролю»). Крім того, системи на основі обмежень часто створюють правила, які пов’язують атрибути користувача з атрибутами товару (наприклад, «Старі інвестори не інвестують у продукти з надвисоким ризиком»). У таких випадках атрибути користувача також можуть бути вказані в

процесі пошуку. Залежно від кількості та типу результатів, що повертаються, користувач може мати можливість змінити свої початкові вимоги. Наприклад, вони можуть послабити деякі свої обмеження, коли повертається занадто мало результатів, або вони можуть додати більше обмежень. Цей процес пошуку повторюється в інтерактивному режимі, поки користувач не отримає бажані результати.[5]

2. Системи рекомендацій на основі випадків: у системах рекомендацій на основі випадків конкретні випадки вказуються користувачем як цілі або опорні точки. Показники подібності визначаються в атрибутах елемента, щоб отримати схожі елементи для цих випадків. Показники подібності часто ретельно визначаються залежно від домену. Таким чином, показники подібності формують знання предметної області, які використовуються в таких системах. Повернені результати часто використовуються як нові цільові випадки з деякими інтерактивними змінами користувача. Наприклад, коли користувач бачить повернений результат, який майже схожий на те, що він хоче, він може повторно надіслати запит із цією метою, але з деякими атрибутами, зміненими на вподобання користувача. Цей інтерактивний процес використовується, щоб направляти користувача на предмети, що цікавлять.[10]

Примітно, що як системи на основі знань, так і на основі контенту, значно залежать від атрибутів предметів. Через використання атрибутів вмісту системи, засновані на знаннях, успадковують деякі з тих самих недоліків, що й системи на основі вмісту. Наприклад, як і системи, засновані на вмісті, рекомендації в системах, заснованих на знаннях, іноді можуть бути очевидними, оскільки не використовують рейтинги спільнот. Основна відмінність полягає в тому, що системи, засновані на вмісті, навчаються на поведінці минулих користувачів, тоді як системи рекомендацій, засновані на знаннях, рекомендують на основі активної специфікації користувачів їх потреб та інтересів.

Системи рекомендацій, засновані на знаннях, як правило, розроблені для доменів, у яких елементи дуже налаштовані, і важко оцінити інформацію, яка безпосередньо відображає більші уподобання. У таких випадках бажано надати користувачеві більший контроль у процесі рекомендацій за допомогою специфікації вимог та інтерактивності. Системи, засновані на знаннях, значною мірою базуються на вимогах користувачів, і вони включають лише обмежену кількість історичних даних. Тому вони зазвичай ефективні при вирішенні проблем з холодним запуском. Недоліком цього підходу є те, що історична інформація не використовується для «заповнення прогалин». В останні роки також були розроблені методи для включення більшої кількості персоналізації з використанням історичної інформації з сеансів користувачів.

1.2.3. Рекомендаційна система з колаборативною фільтрацією

Підхід до спільної фільтрації є найпопулярнішою технікою, яка використовується в рекомендаційниї системах. Він генерує рекомендацію для користувача на основі подібності серед користувачів, які мали схожі переваги/інтереси з ним у минулому. Цей підхід ґрунтується на такій гіпотезі: люди, які домовилися з користувачем у минулому, також погодяться в майбутньому. Основний опис підходу для пояснення попередньої гіпотези показано на рисунку 4. Колаборативна фільтрація ідентифікує нову асоціацію користувач-елемент, визначаючи відносини між користувачами та взаємозалежності між елементами. Він використовує неявні знання спільноти користувачів про використані предмети, щоб визначити відносини цих елементів з іншими користувачами, які не використовували/бачили ці елементи в спільноті. Це можна представити у вигляді матриці користувачів × елементів, у якій кожна клітинка представляє оцінку користувача певного елемента.

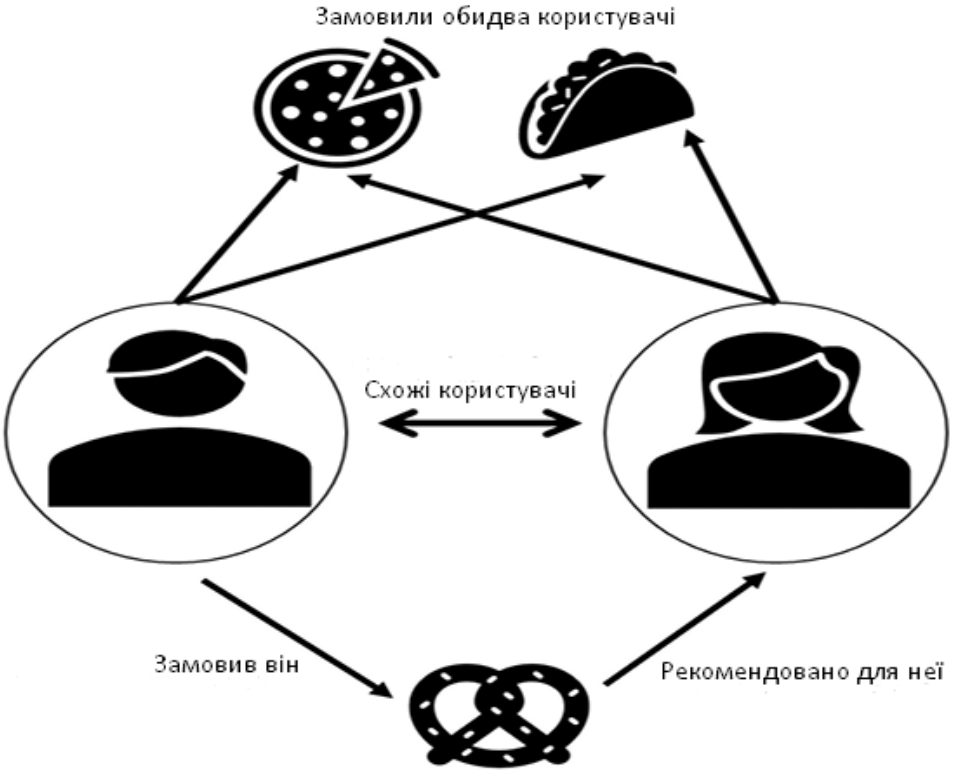

Рис. 5 - Проста ілюстрація методу колаборaційної фільтрації

Існують два типи методів, які зазвичай використовуються у спільній фільтрації, які називаються методами на основі пам’яті (memory-based) та методами на основі моделі (model-based)[1]:

1. Методи на основі пам’яті:

   Методи, засновані на пам’яті, також називають алгоритмами спільної фільтрації на основі сусідства. Це були одні з найперших спільних алгоритмів фільтрації, в яких оцінки комбінацій користувача та елемента передбачаються на основі їх околиць. Ці околиці можна визначити одним із двох способів:

   1.1. Спільна фільтрація на основі користувачів (User-based collaborative filtering):

   Основна ідея полягає в наступному: враховуючи базу даних рейтингів та ідентифікатор поточного (активного) користувача як вхідні дані, ідентифікуються інші

користувачі (іноді їх називають користувачами-рівноправними або найближчими сусідами), які мали подібні переваги до активних користувачів в минулому. Потім для кожного продукту p, який активний користувач ще не бачив, обчислюється прогноз на основі оцінок для p, зроблених схожими користувачами. Основні припущення таких методів полягають у тому, що якщо користувачі мали подібні смаки в минулому, вони будуть мати подібні смаки в майбутньому, і уподобання користувачів залишаються стабільними та незмінними з часом.

1.2. Спільна фільтрація на основі елементів (Item-based collaborative filtering):

Щоб передбачити оцінку цільового елемента B користувачем A, першим кроком є визначення набору S елементів, які найбільш схожі на цільовий елемент B. Використовуються рейтинги в наборі елементів S, які визначаються Функції подібності обчислюються між стовпцями матриці рейтингів, щоб виявити схожі елементи.[8]

Переваги методу на основі пам’яті:

1. Легко реалізувати.
2. Легко додавати нові дані поступово.
3. Зміст рекомендованих елементів розглядати не потрібно.

Недоліки методу на основі пам’яті:

1. Залежить від людських рейтингів.
2. У розріджених даних ефективність рекомендацій знижується.
3. Не можна рекомендувати нові елементи/користувачів (наприклад, проблема холодного запуску).

4. Великі набори даних обмежують масштабованість.

2. Методи на основі моделі:

У методах, заснованих на моделях, у контексті прогнозних моделей використовуються методи машинного навчання та аналізу даних. У випадках, коли модель параметризована, параметри цієї моделі вивчаються в контексті системи оптимізації. Деякі приклади таких методів, заснованих на моделі, включають дерева рішень, моделі, засновані на правилах, байєсівські методи та моделі латентних факторів. Багато з цих методів, наприклад моделі латентного фактора, мають високий рівень охоплення навіть для розріджених матриць рейтингів.

Переваги методу на основі моделі:

1. Проблеми масштабованості та розрідженості краще вирішувати.
2. Покращена продуктивність прогнозування.
3. Для рекомендацій надано інтуїтивне обґрунтування.

Недоліки методу на основі моделі:

1. Створення моделі коштує дорого
2. Зроблено компроміс між масштабованістю та продуктивністю передбачення.
3. Корисна інформація втрачається через методи зменшення розмірності.

Найбільш часто використовувані підходи для визначення схожості користувача/елемента:

1. Коефіцієнт кореляції Пірсона вимірює, наскільки сильний зв’язок між двома користувачами/елементами за допомогою наступного рівняння:

$$Sim(x,y) = \frac{\sum_{i=1}^{n}(r_{x,i} - \overline{r_x})(r_{y,i} - \overline{r_y})}{\sqrt{\sum_{i=1}^{n}(r_{x,i} - \overline{r_x})^2}\ \sqrt{\sum_{i=1}^{n}(r_{y,i} - \overline{r_y})^2}}$$

Де Sim(x,y) у наведеному вище рівнянні позначає подібність між двома користувачами x і y;$r_{x,i}$— це оцінка, яку користувач x надав елементу i; $\underline{r_x}$ – це середня оцінка, яку дає користувач x, тоді як n – загальна кількість місця для елемента користувача.[18]

2. Косинусна подібність відрізняється від міри на основі Пірсона тим, що це модель векторного простору на основі лінійної алгебри, а не статистичний підхід. Він обчислює косинусний кут, спроектований у багатовимірному просторі між двома векторами. Чим ближче значення косинуса до 1, тим менший кут і тим більше подібність між векторами. Ця міра широко використовується в пошуку інформації та аналізі текстів для порівняння двох документів. Подібність двох користувачів x і y можна визначити як:

$$Sim(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x} \cdot \overrightarrow{y}}{|\overrightarrow{x}| * |\overrightarrow{y}|} = \frac{\sum_{i=1}^{n} r_{x,i} * r_{y,i}}{\sqrt{\sum_{i=1}^{n} {r_{x,i}}^2}\ \sqrt{\sum_{i=1}^{n} {r_{y,i}}^2}}$$

Як згадувалося раніше, подібність користувача/елемента визначається на основі показників подібності. Щоб отримати докладнішу інформацію, схожість користувачів у спільній фільтрації на основі користувачів вимірюється шляхом порівняння оцінок загальних елементів, які вони оцінювали. Спільна фільтрація на основі елементів вимірює схожість між елементами, а не користувачами. Він отримує всі елементи, оцінені активним користувачем, із матриці елемента користувача, вирішує, наскільки вилучені елементи подібні до цільового елемента, а потім вибирає k елементів, які найбільш схожі на цільовий елемент. Після обчислення подібності прогноз цільового елемента для користувача розраховується за допомогою різних

алгоритмів передбачення, щоб рекомендувати/не рекомендувати цей елемент користувачеві на основі прогнозованого значення.

Переваги рекомендаційної системи з колаборативною фільтрацією:

1. Здатна рекомендувати більш тонкі елементи і може вловити більше нюансів навколо предметів.
2. Гнучка і підходить для різних доменів.
3. Немає необхідності аналізувати вміст предметів.

1.2.4. Гібридна рекомендаційна система

Цей підхід має на меті пом’якшити слабкі сторони як систему на основі контенту, так і систему на основі колаборативної фільтрації та отримати вигоду з їх сильних сторін шляхом інтеграції двох або більше компонентів рекомендацій або реалізацій алгоритму в єдину систему рекомендацій для підвищення точності рекомендаційної системи та кращої продуктивності. Коли гібридний підхід генерується шляхом гібридизації двох або більше алгоритмів, необхідно враховувати два основні моменти: перший — це рекомендаційні моделі, які оголошують необхідні вхідні дані та визначення, на якому буде ґрунтуватися гібридний рекомендаційний. Другий момент — це визначення стратегії, яка буде використовуватися в рамках гібридного рекомендаційного засобу.[4]

Було ідентифіковано сім різних стратегій гібридизації рекомендаційної системи. Розглянемо кожну з них:

1. Зважений гібрид (Weighted):

   У системі зважених рекомендацій ми можемо визначити кілька моделей, які здатні добре інтерпретувати набір даних. Зважена рекомендаційна система бере результати кожної з моделей і об’єднує результат у статичні зважування, вага яких не змінюється в тестовому наборі. Наприклад, ми можемо поєднати модель на основі контенту та модель спільної фільтрації, і кожна з них має вагу 50% для остаточного

прогнозу. Перевага використання зваженого гібриду полягає в тому, що ми інтегруємо кілька моделей для підтримки набору даних у процесі рекомендацій у лінійний спосіб.

2. Гібрид перемикання (Switching):

   Гібрид перемикання вибирає єдину систему рекомендацій на основі ситуації. Модель використовується для створення конфіденційного набору даних на рівні елемента, ми повинні встановити критерії вибору рекомендацій на основі профілю користувача або інших функцій. Гібридний підхід перемикання вводить додатковий рівень на рекомендаційну модель, який вибирає відповідну модель для використання. Система рекомендацій чутлива до сильних і слабких сторін моделі рекомендацій.

3. Змішаний гібрид (Mixed):

   Змішаний гібридний підхід спочатку використовує профіль користувача та функції для створення різних наборів кандидатів даних. Система рекомендацій відповідно вводить різний набір кандидатів до рекомендаційної моделі та об’єднує передбачення для отримання рекомендації результату. Змішана гібридна рекомендаційна система здатна надавати велику кількість рекомендацій одночасно і пристосовувати частковий набір даних до відповідної моделі, щоб мати кращу продуктивність.

4. Гібрид комбінації функцій (Feature Combination):

   У гібриді комбінації функцій ми додаємо до системи віртуальну модель рекомендацій, яка працює як розробка функцій щодо вихідного набору даних профілю користувача. Наприклад, ми можемо впровадити функції колаборативної рекомендаційної моделі в модель рекомендацій на основі контенту. Гібридна модель здатна розглядати спільні дані з підсистеми, спираючись виключно на одну модель.

5. Гібрид посилення функції (Feature augmentation):

Додаткова рекомендаційна модель використовується для створення рейтингу або класифікації профілю користувача/елемента, який надалі використовується в основній системі рекомендацій для отримання кінцевого прогнозованого результату. Гібрид розширення функцій здатний підвищити продуктивність основної системи без зміни основної рекомендаційної моделі. Наприклад, за допомогою правила асоціації ми можемо покращити набір даних профілю користувача. Завдяки розширеному набору даних ефективність моделі рекомендацій на основі вмісту буде покращена.

6. Каскадний гібрид (Cascade):

   Каскадний гібрид визначає систему рекомендацій із суворою ієрархічною структурою, так що основна рекомендаційна система дає первинний результат, а ми використовуємо вторинну модель для вирішення деяких незначних проблем основного результату, наприклад, порушення рівності в оцінці. На практиці більшість набору даних є розрідженою, вторинна рекомендаційна модель може бути ефективною проти проблеми з рівною оцінкою або проблемою з відсутністю даних.

7. Мета– рівень (Meta-Level):

   Гібрид мета-рівня подібний до гібриду розширення функцій, так що модель, що сприяє, надає доповнений набір даних до основної моделі рекомендацій. На відміну від гібриду розширення функцій, мета-рівень замінює вихідний набір даних на вивчену модель з моделі, що сприяє, як вхідні дані для основної рекомендаційної моделі.

Гібридні рекомендаційні системи використовуються або для використання можливостей кількох джерел даних, або для покращення ефективності існуючих рекомендаційних систем в рамках певної модальності даних. Важливою мотивацією для побудови гібридних рекомендаційних систем є те, що різні типи рекомендаційних систем, такі як спільні, змістовні та засновані на знаннях методи, мають різні сильні та слабкі сторони. Деякі

рекомендаційні системи працюють ефективніше при холодному запуску, тоді як інші працюють ефективніше, коли є достатньо даних. Гібридні рекомендаційні системи намагаються використовувати додаткові переваги цих систем для створення системи з більшою загальною надійністю.

Гібридні системи проектуються як монолітні системи, комплексні системи або змішані системи. Комплексні системи зазвичай розробляються з використанням послідовного або паралельного розташування рекомендацій. У монолітному дизайні або змінюються існуючі рекомендації, або створюються абсолютно нові рекомендації шляхом поєднання функцій з кількох модальностей даних. У змішаних системах рекомендації від кількох двигунів представлені одночасно. У багатьох випадках мета-функції також можуть бути вилучені з певної модальності даних, щоб об'єднати передбачення різних рекомендацій у спосіб, що залежить від входу. Велика сила гібридних і ансамблевих систем обумовлена їх здатністю використовувати додаткові переваги в різних системах[19]

2. Методи обробки інформації

Кількість цифрового вмісту, доступного в Інтернеті, постійно зростає, і це може виявитися складним, щоб якнайкраще використовувати величезну кількість доступних даних. Один із способів вирішення цієї проблеми — класифікувати дані за різними категоріями, таким чином даючи кращий огляд того, які типи даних доступні. Це, у свою чергу, може, наприклад, дозволити користувачам сайту новин краще відфільтрувати статті, які їх цікавлять. Щоб не доводилося виконувати цю процедуру категоризації вручну, замість цього можна використовувати методи машинного навчання для автоматизації процесу.

У більшості статей фактичний текст є як дуже об'ємним, так і неструктурованим. Це означає, що кожне унікальне слово можна розглядати як окремий вимір і що різні статті структуровані по-різному. Це може

ускладнити застосування багатьох класифікаторів до вихідного тексту. Тому спочатку можна виділити найбільш відмінні ознаки тексту, зменшуючи таким чином розмірність. Цей процес називається вилученням функцій (feature extraction). За допомогою деяких методів вилучення ознак ми можемо перетворити текст у матрицю (або вектор) ознак. Деякі з найпопулярніших методів вилучення ознак:

1. Bag-of-Words
2. TF-IDF

Зазвичай, текст для обробки береться з опису товарів, статей, коментарів, тощо. Тому, перед тим, як почати процедуру вилучення функцій, потрібно привести текст до нормального вигляду. Зробити це можна за допомогою наступних операцій:

1. Пониження регістру

   Якщо текст в одному регістрі, машині легко інтерпретувати слова, оскільки нижній і верхній регістр трактуються машиною по-різному. наприклад, такі слова, як «Run» і «run», програма трактує по-різному. Отже, нам потрібно зробити текст в одному регістрі, а найбільш бажаний регістр — це нижній регістр, щоб уникнути таких проблем.
2. Видалення знаків пунктуації

   Один з інших прийомів обробки тексту – видалення розділових знаків. Всього є 32 основні розділові знаки, про які потрібно подбати, а саме !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
3. Видалення стоп-слів

   Стоп-слова – це найпоширеніші слова в тексті, які не містять жодної цінної інформації (артиклі, сполучники, прийменники, тощо).
4. Стеммінг і лемматизація

   Стеммінг – це процес скорочення слова до його кореневої основи, наприклад run, running, runs, runed, що походить від того самого

слова, що й run. В основному стеммінг слова - це видалити префікс або суфікс зі слова, наприклад ing, s, es тощо.

Лематизація подібна до корінного, використовується для того, щоб скласти кореневі слова, але відрізняється в роботі. Насправді лемматизація — це систематичний спосіб звести слова до їхньої леми, зіставляючи їх із мовним словником.

5. Видалення зайвих пробілів

   У більшості випадків текстові дані містять додаткові пробіли або під час виконання вищезазначених методів попередньої обробки між текстом залишається більше одного пробілу, тому нам потрібно контролювати цю проблему.

Отже, коли текст приведений підготовлений, можна приступати до вилучення ознак одним із вказаних вище методів. Розглянемо кожен з них більш детально.

2.1. Модель «Bag of Words»

Один з найпростіших типів моделей вилучення ознак називається Bag of Words. Назва Bag of Words (мішок слів) посилається на той факт, що ця модель не враховує порядок слів. Натомість можна уявити, що кожне слово кладеться в мішок, де порядок слів губиться. Хоча існує кілька різних варіацій цієї моделі, найпоширенішим з них є просто підрахувати кількість зустрічей кожного слова в документі та зберегти результат у векторі (надалі іменований як вектор підрахунку). Таким чином частоти термінів залишаються незмінними, хоча граматика і порядок втрачені. Інший підхід полягає в тому, щоб замість цього мати двійковий вектор, який відстежує, чи існує слово в документі. Однак цей підхід також втрачає множинність слів на додаток до порядку та граматики.[11]

Давайте зробимо модель Bag of Words конкретною на напрацьованому прикладі.

Перший крок. Збирання даних

Нижче наведено уривок перших кількох рядків тексту з книги Чарльза Діккенса «Повість про два міста».

It was the best of times,

it was the worst of times,

it was the age of wisdom,

it was the age of foolishness

Для цього невеликого прикладу розглянемо кожен рядок як окремий «документ», а 4 рядки — як весь наш корпус документів.

Другий крок. Створення словника слів

Унікальними словами тут є:

1. “it”
2. “was”
3. “the”
4. “best”
5. “of”
6. “times”
7. “worst”
8. “age”
9. “wisdom”
10. “foolishness”

Це словниковий запас із 10 слів із корпусу, який містить 24 слова.

Третій крок. Створення векторів документів

Наступним кроком є оцінка слів у кожному документі.

Мета полягає в тому, щоб перетворити кожен документ вільного тексту у вектор, який ми можемо використовувати як вхідні дані або вихідні дані для моделі машинного навчання. Оскільки ми знаємо, що словниковий запас має 10 слів, ми можемо використовувати представлення документа фіксованої довжини 10 з однією позицією у векторі для оцінки кожного слова.

Найпростіший метод оцінки полягає в тому, щоб позначити наявність слів як логічне значення, 0 – відсутність, 1 – наявне.

Використовуючи довільний порядок слів, перерахованих вище, у нашому словнику, ми можемо пройти через кожний документ і перетворити його в двійковий вектор.

Оцінка кожного документа буде виглядати так:

“it was the best of times” = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

Увесь порядок слів номінально відкидається, і ми маємо послідовний спосіб вилучення функцій з будь-якого документа нашого корпусу, готового до використання в моделюванні. Увесь порядок слів номінально відкидається, і ми маємо послідовний спосіб вилучення функцій з будь-якого документа нашого корпусу, готового до використання в моделюванні.

2.2. Модель «TF-IDF»

Одним із методів, який зарекомендував себе як простий та ефективний для вилучення ознак, є Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF – це техніка пошуку інформації, яка може бути використана для визначення відповідності термінів документів щодо запиту . У цьому випадку його можна використовувати для виділення ознак, визначаючи, які терміни в документі найбільше відрізняють цей документ. Цей метод також можна розглядати як модель BagofWords, оскільки він не враховує граматику чи порядок.

TF-IDF складається з двох кроків: спочатку обчислюється частота термінів (TF), а потім обчислюється частота зворотного документа (IDF). Існує кілька варіантів обох цих частин.

Для обрахунку значення TF спочатку обчислюється, скільки разів термін зустрічається в документі, так само, як і для вектора підрахунку. Міркування тут полягає в тому, що слова, які часто зустрічаються в документі, ймовірно, важливіші за слова, які зустрічаються нечасто. Потім результат нормалізується, розділивши його на кількість слів у всьому документі. Ця нормалізація виконується для того, щоб запобігти упередженням щодо більш довгих документів, щоб ми отримали частоту, з якою зустрічається термін, а не лише необроблену кількість термінів. Формула має наступний вигляд:

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

Де $n_{t,d}$– кількість випадків, коли термін t зустрічається в документі d, а $n_{k,d}$– кількість зустрічей кожного терміна в документі d.

Для обчислення частини формули IDF один із варіантів полягає в тому, щоб взяти загальну кількість документів у корпусі та поділити її на кількість документів, де з'являється термін. Потім результат логарифмується. Частина формули IDF діє як форма присвоєння ваги, надаючи більшу вагу важливим термінам і меншу вагу несуттєвим термінам. Формула має наступний вигляд:

$$idf_t = log\frac{|D|}{|D_t|}$$

Де $|D|$ – загальна кількість документів, а $|D_t|$– кількість документів, у яких з'являється термін t.

Помноживши частину TF і частину IDF на певний термін, ми отримаємо міру того, наскільки цей термін відрізняється. У випадку, коли використовується корпус новинних статей, слово «компанія», ймовірно, матиме досить високу оцінку TF-IDF, оскільки воно часто зустрічатиметься в статтях, пов'язаних із бізнесом. Однак це не буде дуже поширеним у кожному

іншому документі, оскільки новини, що стосуються спорту, культури та інших тем, ймовірно, не дуже часто містять цей конкретний термін. На відміну від цього поширене слово, як-от «сьогодні», ймовірно, отримає досить низьку оцінку TF-IDF, оскільки воно, ймовірно, з’явиться в статтях новин, що стосуються кожної теми новин, і тому не дуже розрізняє термін.

Однією зі слабких сторін цього методу та подібних методів Bag of Words загалом є те, що він не враховує контекст та синоніми. Це, наприклад, не бере до уваги, що терміни «священик» і «преподобний» стосуються подібного предмета. Він також не бере до уваги, що термін «Apple» може означати компанію Apple Inc. або фрукт залежно від контексту.[11]

3. Опис розробки рекомендаційної системи в сфері працевлаштування на основі оголошень про вакансії

3.1. Опис вхідних даних

В якості бази даних, був обраний набір даних, який містить поточні оголошення про вакансії, доступні на офіційному сайті вакансії міста Нью-Йорк. Дані взяті за сайту data.world. Формат даних – файл csv.

Атрибути файлу:

1. Job ID
2. Agency
3. Posting Type
4. # Of Positions
5. Business Title
6. Civil Service Title
7. Title Code No
8. Level
9. Job Category
10. Full-Time/Part-Time indicator

11. Salary Range From
12. Salary Range To
13. Salary Frequency
14. Work Location
15. Division/Work Unit
16. Job Description
17. Minimum Qual Requirements
18. Preferred Skills
19. Additional Information
20. To Apply
21. Hours/Shift
22. Work Location 1
23. Recruitment Contact
24. Residency Requirement
25. Posting Date
26. Post Until
27. Posting Updated
28. Process Date

3.2. Метод підбору рекомендацій

В роботі розглядається рекомендаційна система, на основі контенту. До кожної вакансії підбирається подібна їй вакансія за атрибутом «Job Description»(«Опис вакансії»). Для аналізу тексту використовувався TF-IDF метод.

Для початку були проведені зміни до тексту, а саме:

1. Приведення тексту до нижнього регістру: Management Information Systems - management information systems
2. Видалення знаків пунктуації !, \, ", #, $, %, &, (,), *, +, -, ., /, :, ;, <, =, >, ?, @, [, \,].
3. Використання стемінгу (зведення до спільного корення слів)

Далі, отримаємо множину унікальних слів з усіх набор. Для кожного слова вираховується значення TF-IDF в кожному наборі за формулою:

TF-IDF= TF * IDF

TF = k/d, де k – кількість конкретного слова в наборі, а d – кількість слів в наборі

IDF = ln(N/n), де N – кількість наборів слів, а n – кількість наборів в якому упоминається конкретне слово

Для складення рейтингу подібності вакансій, використовується cosine similarity, який формульно має вигляд:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$$

де x, y – вектори, які отримані із значень TF-IDF для кожної вакансії.

Після розрахунків маємо значення схожості даної вакансії з іншими, тому ми можемо скласти рейтинг та обрати M схожих вакансій.

3.3. Опис розробки

Мовою програмування була обрана мова Python, оскільки за допомогою неї зручно зберігати та використовувати дані. Python використовувався для розрахунків та для розробки інтерфейсу користувача.

Для початку, дані з csv файлу були записані в датафрейм, який був створений за допомогою бібліотеки pandas.

Після проведення змін до тексту та створення словника слів, рахується значення IDF для усіх слів. Для збереження цих значень використовується словник (dict), де ключем виступає слово, а значенням – число.

IDF = {

'each': 2.0402208285265546,

'performs': 2.407945608651872,

'retrieve': 4.605170185988092,

'claiming': 3.912023005428146,

…}

Далі, перевіряється кожен набір опису вакансії та для кожного слова обраховується значення TF і відразу ж значення TF-IDF.

Для збереження значень TF-IDF використовувався словник. Ключем в словнику виступає «Job ID». Під кожним ключем, зберігається також словник даних, де під ключем виступає слово з множини слів, а значеннями виступає значення TF-IDF.

Словник TF-IDF має наступний вигляд:

TF-IDF = {

87990: {student:0.0056

recommendation:0.0

audit:0.0708

…}

97899: {student:0.0

recommendation:0.0026

audit:0.0

…}

102221: {student:0.0

recommendation:0.0

audit:0.0

…}

114352: {student:0.0

recommendation:0.0

audit:0.013

…}

350278: {student:0.0

recommendation:0.0

audit:0.285

…}

…}

Наступним кроком була розробка функції cosine similarity. Для розробки була використана бібліотека numpy. За допомогою цієї функції був складений словник вакансій, в якому ключем виступав «Job ID», а значеннями список зі Job ID інших вакансій, розташованих у порядку спадання значення cosine similarity. Приклад:

{87990 : [97899, 345905, 231077, 151937, 120749, ...]

97899 : [120749, 132786, 87990, 124564, 172053, ...]

102221 : [204915, 151405, 170412, 174736, 192766, ...]

114352 : [152679, 187199, 197355, 192766, 181317, ...]

350278 : [230490, 187714, 195805, 192062, 192756, ...]

… }

3.4. Інтерфейс користувача

Для зручної роботи користувача був розроблений інтерактивний веб-портал за допомогою бібліотеки streamlit в Python. На початку програми словник, отриманий на минулому кроці, був записаний в окрему змінну.

Після відкриття веб-порталу, перед користувачем з’являється стартова сторінка, де він може ввести ключове слово для пошуку:

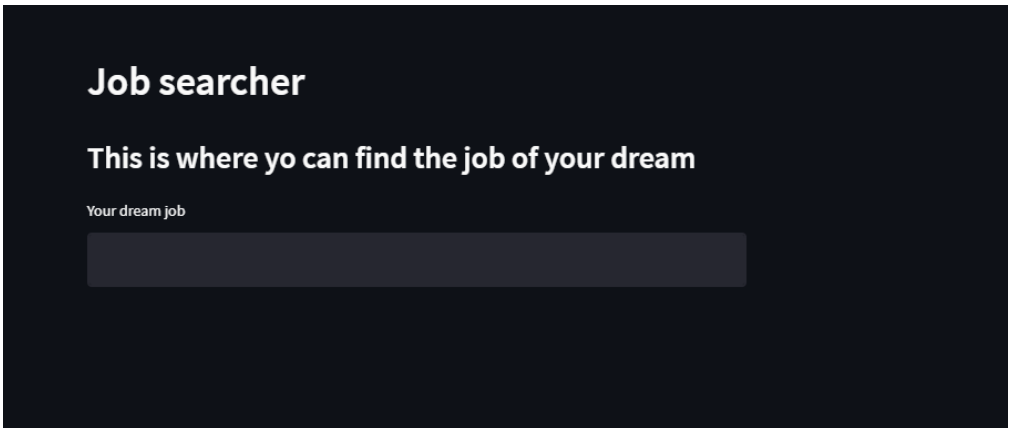

Рис. 6 - Стартове вікно

Після введення ключового слова, з’являється список вакансій, в назві яких присутнє введене слово. Також, є можливість зробити пошук ще раз або повернутися на стартову сторінку:

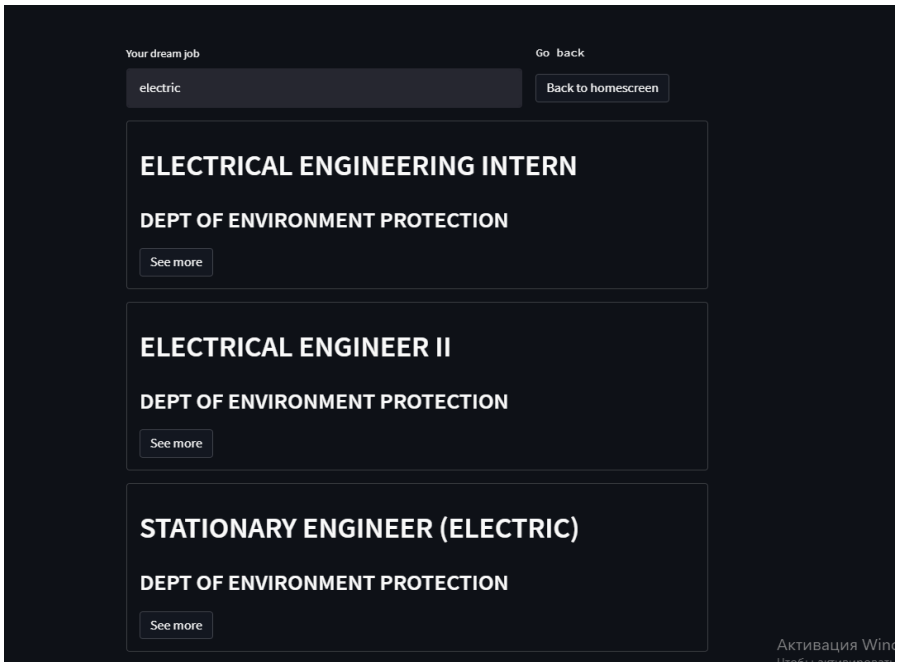

Рис. 7 - Список знайдених вакансій

При натисканні кнопки «See more», відкривається вікно з повним описом вакансії:

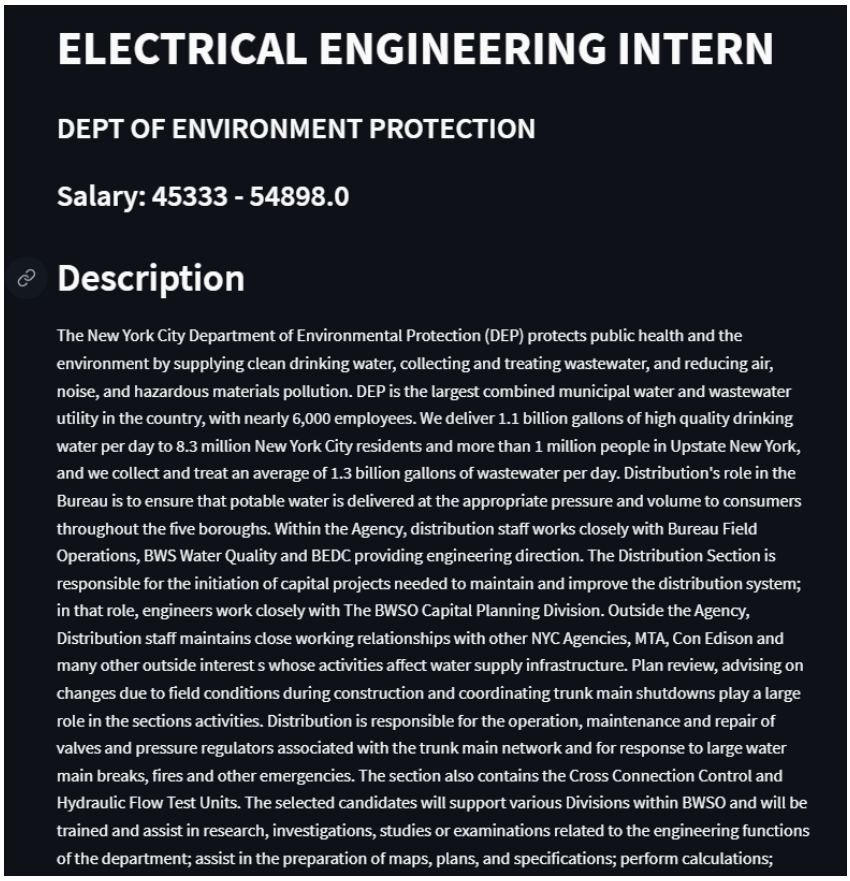

Рис. 8 - Опис вакансії

У вікні вакансії можна знайти наступні атрибути:

1. Назва вакансії
2. Назва компанії
3. Заробітна плата
4. Опис вакансії
5. Вимоги до кандидата
6. Схожі вакансії

Останній пункт зображено на рисунку 9. При натисканні на кнопку «See more», користувач перейде на опис цієї вакансії. В самому низу вікна знаходиться кнопка «Back to list», за допомогою якої користувач може повернутися на початок пошуку.

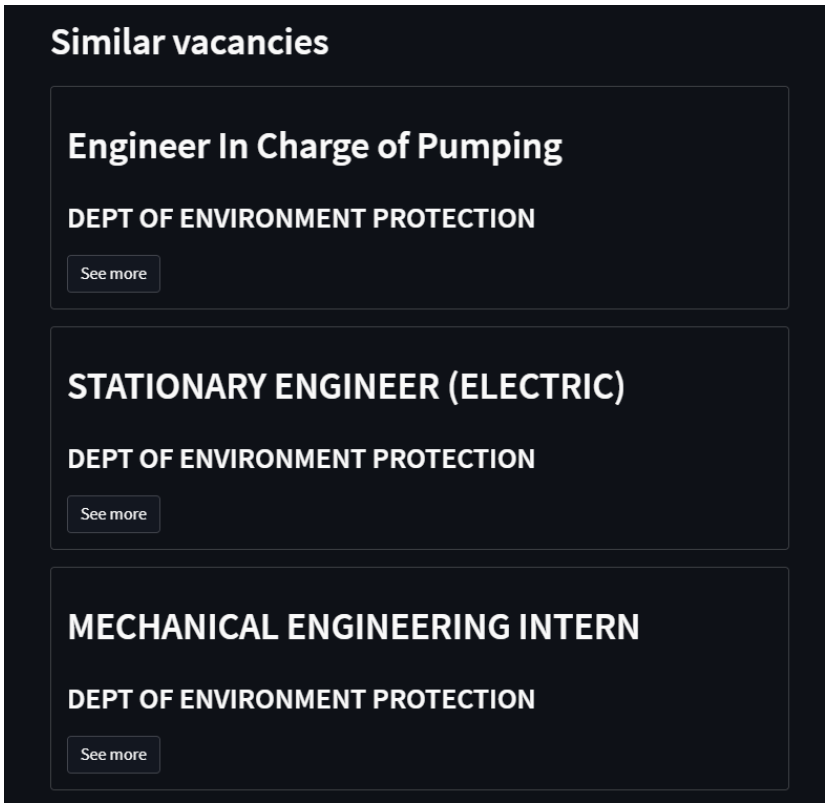

Рис. 9 - Список схожих вакансій

Висновок

В кваліфікаційній роботі було досліджено основні принципи побудови та види рекомендаційних систем. В роботі було запропоновано рекомендаційну систему побудовану на основі методу аналізу контенту. Запропонована рекомендаційна система використовує публічний датасет вакансій міста Нью Йорк, проте це не єдина сфера в якій вона може бути застосованою. Основною вимогою для такої системи – є наявність або текстової частини в даних. Датасет що досліджувався, містить в собі набір об’яву вакансії, для кожної з яких відомо текстовий опис. Тому було застосовано метод TF-IDF для вилучення ознак з описів оголошень та метод cosine similarity для оцінки схожості вакансій. Результати оцінювання схожості стали основою для словника рекомендацій, в якому для кожної вакансії підібрані схожі вакансії та записані у порядку спадання оцінки схожості. Для генерації рекомендацій, користувач обирає одну з вакансій на основі критеріїв пошуку, а далі рекомендаціями будуть інші вакансії, з якими поточна має найбільшу вагу. Для зручності використання рекомендаційної системи, було створено інтерактивний веб-портал в який було інтегровано реалізовану систему.

Запропонована рекомендаційна система, виявилася ефективною, тестові генерації рекомендацій були схожими та задовольняли потреби користувачів. Проте, система має недоліки пов’язані з появою нових оголошень в системі, але ці недоліки легко виправити застосовуючи ефективнішу реалізацію методів порівняння контенту.

Список літератури

1. Aggarwal, C. C. (2016). *Recommender Systems The Textbook.* New York: Springer.
2. Asanov, D. *Algorithms and Methods in Recommender Systems.* Berlin: Berlin Institute of Technology.
3. Bertsekas., D. P. (1999). *Nonlinear programming.* Belmont: Athena Scientific Publishers.
4. Bishop, C. M. (2007). *Pattern recognition and machine learning.* Springer.
5. Burke., R. (2000). *Knowledge-based recommender systems.*
6. Chen, Y. W. (2001). Solving the Sparsity Problem in Recommender Systems Using Association Retrieval. *Journal of computers* .
7. D. Bridge, M. G. (2005). *Case-based recommender systems. The Knowledge Engineering Review.*
8. Dietmar Jannach, M. Z. (2011). *Recommender Systems An Introduction.* New York: cambridgeuniversitypress.
9. Dwork., C. (2011). *Encyclopedia of Cryptography and Security.* Springer.
10. E. Candes, X. L. (2011). Robust principal component analysisv. *Journal of the ACM*.
11. EKLUND, M. (2018). *Comparing Feature Extraction Methods and Effects of Pre-Processing Methods for Multi-Label Classification of Textual Data.* STOCKHOLM.
12. Golbeck, J. (2008). *Computing with social trust.* Springer.
13. Gunawardana, A. &. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research* .

14. Hosanagar., D. M. (2007). *Recommender systems and their impact on sales diversity.* ACM Conference on Electronic Commerce.

15. Kantor, F. R. (2011). *Recommender Systems Handbook.* New York: Springer.

16. L. M. de Campos, J. F.-L.-M. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning*.

17. Liu., B. (2007). *Web data mining: exploring hyperlinks, contents, and usage data.* New York: Springer.

18. Noah, S. M.-G. (без дати). *A Comprehensive Overview of Recommender System and Sentiment Analysis.* Malaysia, Yemen: Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Faculty of Applied Sciences, Department of Computer Science, Taiz University.

19. Ricci, F. (2002). *Travel recommender systems. IEEE Intelligent Systems.*

20. Ricci., F. L. (2005). *Case-based recommender systems: a unifying view.*

21. Rubin., R. L. (2002). *Statistical analysis with missing data.* Wiley.

22. S. Bhagat, G. C. (2011). *Social Network Data Analytics.* Springer.

23. Schapire, Y. F. (2011). *Experiments with a new boosting algorithm.* ICML Conference.

Додаток А. Код програмної реалізації словника рекомендацій

```
df = []

def cos(x, y):

    v1 = np.array(list(x.values()))

    v2 = np.array(list(y.values()))

    return(np.dot(v1, v2)/(np.linalg.norm(v1)*np.linalg.norm(v2)))

  def get_rec(vacancy):

    vac_list = set(df['Job ID'][df['Job ID'] != vacancy])

    top = dict()

    for vac in vac_list:

      top[vac] = cos(TF_IDF[vacancy], TF_IDF[vac])

    top = {k: v for k, v in sorted(top.items(), key=lambda item: item[1], reverse
=True)}

    lst = top.keys()

    st = set(lst)

    return list(lst)

  df['clear'] = df['Job Description'].str.lower().str.translate(str.maketrans('', '',
string.punctuation))

  Bag = dict(zip(df['Job ID'], [i.split() for i in df['clear']]))

  all_words = set().union(*Bag.values())

  IDF = dict()

  for word in all_words:
```

```
    count = sum([word in descr for descr in df['clear']])
    IDF[word] = math.log(len(df)/count)
TF_IDF = dict()
for key in Bag.keys():
    for word in all_words:
        DF = Bag[key].count(word)/len(Bag[key])
        if key in TF_IDF.keys():
            TF_IDF[key][word] = DF*IDF[word]
        else:
            TF_IDF[key] = {word : DF*IDF[word]}
rec_dict = dict()
for id in df['Job ID']:
    rec_dict[id] = get_rec(id)
return rec_dict
```