Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Розробка системи рекомендацій вакансій з використанням нейролінгвістичного програмування

Текстова частина до курсової роботи

за спеціальністю „Комп'ютерні науки" 122

Керівник курсової роботи

доцент

Ющенко Ю. О.

_____

"____" _____ 2023 р.

Виконала студентка КН-4

Васильєва Т. Є.

"____" _____ 2023 р.

Київ 2023

Календарний план

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітки
1.	Отримання завдання на курсову роботу.	13.10.2022	
2.	Огляд технічної літератури за темою роботи.	12.01.2023	
3.	Аналіз актуальності теми	15.01.2023	
3.	Дослідження теоретичної частини	29.03.2023	
4.	Написання теоретичної частини	29.03.2023	
5.	Кінцеві правки практичної роботи	20.04.2023	
6.	Написання практичної частини	30.04.2023	
7.	Подання роботи на кафедру для перевірки на плагіат	20.05.2023	

Зміст

Анотація **4**
Вступ **5**
1. Актуальність проблеми 5
2. Що таке рекомендаційна система? 5
3. Мета та структура роботи 6
1. Розділ 1. Рекомендаційні системи та NLP **7**
1.1. Рекомендаційні системи 7
1.2. Natural language processing 10
1.3. Висновки до розділу 12
2. Розділ 2. Огляд використаних технологій **13**
2.1. Фреймворк CRISP-DM 13
2.2. Spacy 18
2.3. Flask 21
2.4. Pandas 22
2.5. MongoDB 23
3. Розділ 3. Впровадження системи **25**
3.1. Структура проекту 25
3.2. Джерела даних 26
3.4. Відображення системи 29
4. Висновки **31**
5. Список використаної літератури **32**

Анотація

У даній роботі розроблено систему рекомендацій вакансій за навичками користувача, що має назву Jobs by Skills. Також було досліджено методи побудови рекомендаційних систем. Під час розробки були використані такі мови програмування та технології: Python, Flask, Spacy, MongoDB, Pandas. В наступних розділах цієї роботи будуть детально описані актуальність проблеми, методи побудови рекомендаційних систем, застосовані технології, методологія розробки та архітектура кінцевого продукту.

Вступ

1. Актуальність проблеми

Пошук роботи є однією з найбільш актуальних проблем у сучасному світі, адже часто люди змінюють роботу більше 3 разів у своєму житті. Постійний пошук гарного місця працевлаштування забезпечує постійний цикл процесу найму. Багато компаній, що займаються працевлаштуванням, винайшли рішення, яке допомагає людям знаходити роботу, а компаніям - кандидатів для їхніх вакансій, а саме дошки вакансій. Тут шукач шукає роботу, яка йому підходить, і подає заявку на неї. Оскільки існує багато дошок вакансій, кандидати, як правило, використовують інструменти, які надають їм кращі послуги, наприклад написання резюме, створення профілю роботи та рекомендація нових вакансій шукачеві. Претенденти на роботу стали більш наполегливими та ініціативними у пошуку нових можливостей, які відповідають їхнім навичкам. Однак компаніям, які орієнтуються на таких шукачів роботи, важко визначити навички шукача роботи та надати персоналізовані рекомендації щодо роботи.

2. Що таке рекомендаційна система?

Рекомендаційна система - програма, яка намагається передбачити, які об'єкти (фільми, музика, книги, новини, веб-сайти) будуть цікаві користувачеві, маючи певну інформацію про його профіль.

Потреба в системі рекомендацій час від часу зростала. Спочатку індустрії розваг почали використовували переваги цих систем (наприклад, згідно зі звітами бізнес-сайдерів Австралії, Netflix оцінює, що 20% їхніх

переглядів відео є результатом пошуку. У той час як 80% надходить від його системи рекомендацій). Потім системи рекомендацій були впроваджені в інтернет-магазини та онлайн-новини, але дуже небагато компаній намагалися застосувати їх у процесі найму.

3. Мета та структура роботи

Метою даної роботи є розробка системи рекомендації вакансій на основі навичок користувача.

Ця робота поділена на кілька розділів:

- **Розділ перший.** Містить опис методів впровадження рекомендаційних систем, інформацію щодо технології NLP, та деяких показників подібності, які використовуються для дослідження в області рекомендацій.
- **Розділ другий.** У цьому розділі розповідається про методи, інструменти та технології, які використовувалися під час написання роботи.
- **Розділ третій.** У цьому розділі розглядається впровадження системи.
- **В кінці роботи** наведені висновки та список використаної літератури.

1. Розділ 1. Рекомендаційні системи та NLP

Системи рекомендацій стають частиною багатьох бізнесів. Підприємства намагаються збільшити свої доходи, підвищуючи взаємодію користувачів, рекомендуючи нові товари на основі уподобань користувача. Сервіс Netflix став популярним у сфері розваг, завдяки стратегії впровадження системи рекомендацій у власну екосистему. Проте було проведено мінімальне дослідження у сфері пошуку роботи, тому дуже важливо переглянути відповідні наробки в галузі та існуючі технології.

1.1. Рекомендаційні системи

Як обговорювалося раніше, RecSys — це система, яка аналізує історію налаштувань користувачів і надає їм різні варіанти послуг, пов’язаних із вимогою. Системи рекомендацій виникли як незалежний напрямок досліджень у середині 1990-х. В останні роки, різко зріс інтерес до рекомендаційних систем. Існують різні підходи до їхньої побудови. Іноді використовуються тільки коефіцієнти подібності, наприклад, коефіцієнт Жаккара (як в практичній частині даної роботи), так і рекомендаційні алгоритми. Вони класифікуються за чотирма типами: Content-based filtering, Collaborative filtering, Rule-based filtering, та гібридний підхід. Розглянемо кожний із цих типів нижче.

- **Спільна фільтрація** (**collaborative filtering)**: ця техніка ґрунтується на оцінці елементу користувачем, і знаходженні подібності між різними користувачами, які дали подібні оцінки елементам. Основна операція, яка тут використовується, — це метод найближчих сусідів на основі пам’яті для групування користувачів, які мають подібні інтереси. У міру

зростання обсягу даних, буде зростати і затримка у створенні рекомендацій. Спільна фільтрація має перевагу над методами фільтрації на основі вмісту, але через характер процесу найму користувач не може оцінити вакансію. Це унеможливить використання даного методу для системи рекомендації вакансій, адже немає змоги створити матрицю подібності.

- **Фільтрація на основі вмісту (Content-based filtering)**: це найбільш суб'єктивний метод фільтрації. Фільтрування на основі вмісту також можна назвати рекомендуванням на основі атрибутів, оскільки воно використовує явно визначену властивість елемента. В цьому методі робиться припущення, що користувач віддає перевагу елементу з подібними властивостями. Фільтрація на основі вмісту рекомендує користувачеві елементи, властивості яких схожі на елемент, до якого користувач раніше виявляв інтерес. Недоліком цієї техніки фільтрації є тенденція до “надмірної спеціалізації” в пропонуванні елементу користувачу. Оскільки профілі користувачів базуються на атрибуті попереднього елемента, який вибрав користувач. Тим не менш, у сфері пошуку роботи, вакансії доступні лише протягом певного періоду часу, тому тенденція до надмірної спеціалізації в рекомендаціях одного і того ж предмета не буде проблемою в системі рекомендацій вакансій. У таких сферах, як розваги, уподобання користувачів змінюються залежно від різних факторів, але в сфері пошуку вакансій користувач прагне шукати роботу, де він може використати свої попередні навички. Нові рекомендації вакансій можна зробити, коли є зміни в уподобаннях користувача, тобто якщо користувач прагне змінити сферу своєї роботи, оновивши свої навички та сферу роботи. Іншим сценарієм нової

рекомендації є оновлення вакансій у базі даних. Тоді система буде ідентифікувати властивості вказаної вакансії, та ставити у відповідність з користувачами, що мають високий показник подібності.

- **Фільтрація на основі правил (Rule-based filtering)**: наразі в галузі електронної комерції використовується метод фільтрації на основі правил, щоб рекомендувати товар на основі демографічного регіону користувача, історії покупок та інших атрибутів, які можна використовувати для профілювання користувача. Недолік фільтрації на основі правил полягає в тому, що користувач надає особисту інформацію системі. Ці вхідні дані використовуються як опис профілю користувача або можуть розглядатися як уподобання користувача, визначені користувачем. Якщо користувач надасть неправдиві або неточні дані, отримані рекомендації можуть бути неточними.

- **Гібридна фільтрація (Hybrid Filtration)**: комбінація декількох технік фільтрації для покращення ефективності рекомендацій. Цей метод можна використовувати, щоб отримати кращі рекомендації та подолати проблеми, пов’язані з попередніми методами. Опитування, проведені Burke (2002) та Dhameliya та Desai (2019), виявили різні типи методів гібридної фільтрації, які можна використовувати шляхом інтеграції CF, CBF та RBF.
    1. Зважений: оцінка подібності, отримана від різних компонентів рекомендації, об’єднується чисельно, щоб отримати одну кращу рекомендацію.

2. Змішаний: рекомендації, отримані за допомогою різних методів рекомендацій, об’єднуються та представляються як одна рекомендація.
3. Перемикання: вибір одного з компонентів рекомендацій на основі сценаріїв, де він найкраще підходить.
4. Комбінація функцій: атрибути, отримані з різноманітних джерел, об’єднуються та додаються до рекомендаційного алгоритму.
5. Розширення функцій: один метод рекомендації використовується для обчислення набору атрибутів користувача або елемента, який потім є частиною вхідних даних для наступного методу рекомендації.

1.2. Natural language processing

Це часи, які можна вважати епохою даних. У цій великій кількості даних текстові дані займають більшу частку, причому більшість з них знаходяться в неструктурованій формі. Тому постає питання, як можна отримати інформацію з неструктурованих текстових даних або як змусити комп’ютер зрозуміти, про що йдеться в тексті? Для відповіді на ці питання використовується аналіз тексту, який надає змогу отримати корисну інформацію з текстових даних. Аналіз тексту можна виконати за допомогою таких методів, як обробка природної мови. Обробка природної мови — це процес пошуку інформації з неструктурованих даних.

Тоді постає завдання визначення подібності тексту. Існують різні підходи для виявлення подібності між текстом: алгоритм редагування відстані, техніку лексичного накладення. Проте більш надійною є техніка вбудовування слів. Слова представляються в семантичному просторі як

вектор. Тобто слова, семантично подібні, залишатимуться близькими в семантичному просторі. Щоб отримати терміни, які базуються на подібності між двома термінами, використовується найвідоміший метод під назвою word2vec, який є моделлю векторного простору, після цього можна використати косинусну подібність для вимірювання подібності між термінами. Цю модель також можна використовувати для визначення подібності між реченнями. Зазвичай word2vec приймає як вхідні дані великий набір слів, який називається corpus, і створює векторний простір. Після створення векторної просторової моделі можна використовувати метод вимірювання подібності, щоб визначити відстань або наскільки схоже слово, з яким ми порівнюємо. Щоб знайти подібність у векторному просторі, можна використовувати такі міри подібності, як косинусна подібність і подібність за Жаккаром.

1. **Коефіцієнт Жаккара**: коефіцієнт Жаккара — це метод порівняння елементів двох наборів даних, щоб визначити, які елементи є спільними, а які - різними. Це міра подібності для двох наборів даних із результатом від 0% до 100%. Два набори можна назвати схожими, коли результат близький до 100%. Формула для індексу Жаккара показана нижче:

$$\text{Індекс Жаккара} = \frac{\text{кількість елементів, спільних у двох наборах}}{\text{кількість елементів у двох наборах}}$$

Формула 1.1 Індекс Жаккара

2. **Косинусна міра подібності**: косинусна міра подібності також є мірою для визначення подібності між двома наборами

ненульового векторів. Це зважена векторна модель простору, яка використовується в процесі отримання інформації. Подібність вимірюється за допомогою правила евклідового косинуса, тобто шляхом використання простору внутрішнього добутку двох ненульових векторів, який вимірює косинус кута між двома векторами. Якщо кут між двома векторами дорівнює 0°, то косинус 0 дорівнює 1; Це означає, що два ненульові вектори подібні один до одного. Щоб зважити слова, використовується добре відома модель векторного простору word2vec.

$$\text{Косинусна міра подібності (A, B)} = cos(\theta) = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Формула 1.2 Косинусна міра подібності

Також можна обчислити косинусну відстань за допомогою наведеного нижче рівняння:

Косинусна відстань = 1− (Косинусна міра подібності).

Формула 1.3 Косинусна відстань

1.3. Висновки до розділу

Виходячи з усіх дослідницьких методологій і технік, розглянутих у цьому розділі для системи рекомендацій вакансій найбільше підходить метод фільтрації на основі вмісту. Метод спільної фільтрації не можна розглядати, оскільки він не відповідає цілям дослідження: набір даних користувача не

містить інформації про надану оцінку певної вакансії, тому неможливо створити матрицю подібності, необхідну для техніки CF (спільної фільтрації).

Варто зазначити, що у практичній частині даної роботи використовується тільки коефіцієнт Жаккара для отримання релевантних навичкам користувача вакансій.

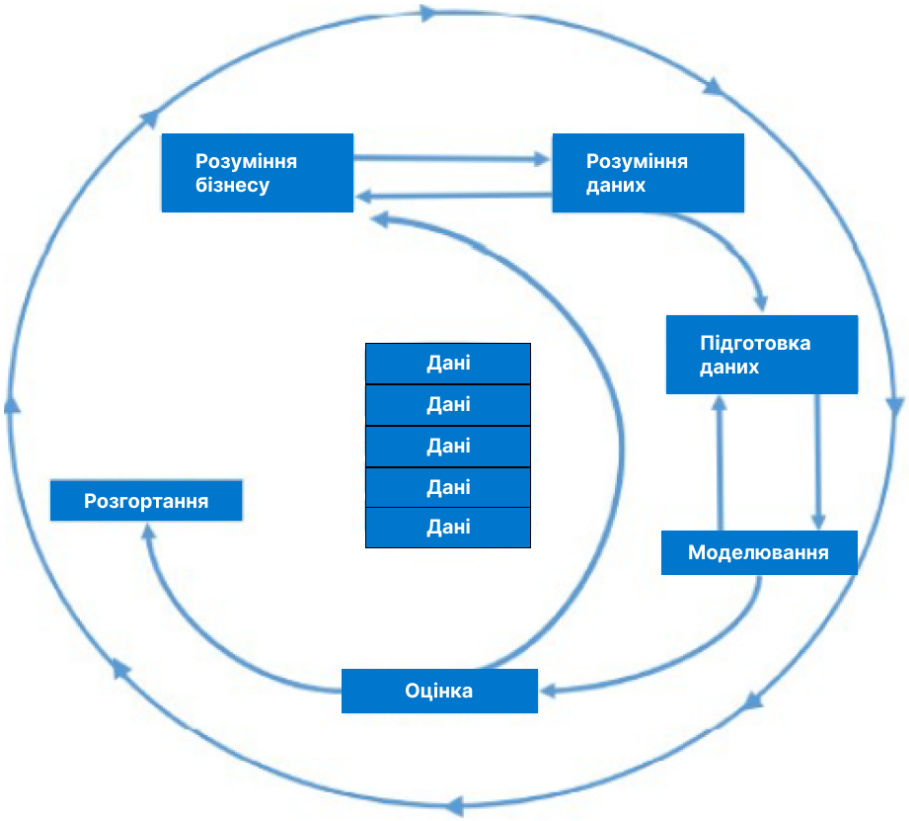
2. Розділ 2. Огляд використаних технологій

У цьому розділі розглядаються використані технології для розробки системи.

2.1. Фреймворк CRISP-DM

Міжгалузевий стандартний процес обміну даними (Cross-Industry Standard Process for Data Mining) представляє собою добре структурований підхід до проектів з аналізу даних. Цей фреймворк використовується різноманітними організаціями, оскільки є нейтральним для всіх галузей та додатків та забезпечує кращий результат від аналізу даних. Загальна модель CRISP-DM надає огляд завдань і етапів інтелектуального аналізу даних.

Модель CRISP-DM має шість етапів, які показані на малюнку нижче.



Мал. 2.1 Шість фаз моделі CRISP-DM

Після завершення фази, залежно від результатів, отриманих на цій фазі, буде вирішено перейти до наступної фази або повернутися до будь-якої стадії фази, щоб отримати бажаний результат. Таким чином, ці фази не мають суворої послідовності. Результати цих фаз будуть одним із факторів для прийняття рішення про те, чи переходити далі до наступної фази, чи повторно запустити ту саму фазу. Далі буде детально розглянутий кожен етап, визначивши завдання, які мають виконуватися на кожному етапі процесу інтелектуального аналізу даних.

Розуміння бізнесу: Це початковий процес CRISP-DM. На цьому кроці відбувається розуміння мети проекту. Крім того, критерії повинні бути

встановлені з точки зору бізнесу. Це дозволить визначити задачі аналізу даних для створення стратегії, щоб досягти мети. Дану інформацію можна отримати за допомогою таких етапів:

1. Оцінка ситуації.
2. Визначення цілей інтелектуального аналізу даних.
3. Створення плану проекту.

Розуміння даних: Розуміння даних — це процес, під час якого констатуються факти щодо даних для отримання більш повної інформації про них. У цьому процесі також формуються гіпотези щодо прихованих знань про мету проекту інтелектуального аналізу даних. Гіпотези створюються на основі досвіду та професійних припущень. Мають бути виконані такі етапи:

1. Збір вихідних даних
2. Опис даних
3. Дослідження даних
4. Перевірка даних

Підготовка даних: це процес, під час якого аналітик збирає дані, пов’язані з проектом, і готує їх для подальших етапів. Етап підготовки даних включає всі дії від початкової стадії необроблених даних до створення остаточного набору даних. Завдання підготовки даних можна виконувати кілька разів і в будь-якому порядку. Нижче наведені етапи, які необхідно виконати під час підготовки даних.

1. Вибір даних.
2. Очистка даних.
3. Конструювання даних.
4. Інтеграція даних.

5. Форматування даних.

Моделювання: серед доступних прийомів моделювання вибір того, що відповідає поставленому завданню. Обрана техніка моделювання може передбачати наявність даних у певному форматі. В такій ситуації, можливо, доведеться повернутися до моделі підготовки даних, перш ніж знову працювати над фазою моделювання.

1. Вибір техніки моделювання.
2. Створення тестового дизайну.
3. Конструювання даних.
4. Створення моделі.
5. Оцінка моделі.

Оцінка: на цьому етапі буде отримана робоча модель для аналізу даних. Однак перед тим, як приступити до розробки, потрібно перевірити модель на фактичних наборах даних виробничої системи та порівняти результати інтелектуального аналізу даних з бізнес-ціллю. На цьому етапі важливо визначити, чи відповідає модель вимогам користувача. Нижче наведено кроки, які необхідно виконати на етапі оцінки:

1. Оцінка результату.
2. Огляд процесу.
3. Визначення подальших кроків.

Розгортання: коли модель відповідає вимогам користувача на етапі оцінки, її потрібно розгорнути у виробничому середовищі. Це залучає усіх зацікавлених сторін проекту, головним чином користувачів і аналітиків. Групі

аналітиків необхідно надати користувачам інформацію про те, як вони можуть використовувати модель. Нижче наведено етапи, які мають бути виконані:

1. Планування розгортання.
2. Планування моніторингу та обслуговування.
3. Формування підсумкового звіту.
4. Огляд проекту.

Цей фреймворк розроблений для проблем, які вимагають ітераційного підходу. Він служить для кращого розуміння даних перед застосуванням їх до моделі. Впровадження системи здійснюватиметься на основі фреймворку CRISP-DM. Фреймворк CRISP-DM модифіковано до рівня, на якому він відповідає бізнес-вимогам розробки системи рекомендацій для шукача роботи. На малюнку нижче показано модифіковану структуру CRISP-DM для системи рекомендацій.

Мал. 2.2 Модифікована CRISP-DM для моделі рекомендаційної системи

Схема, запропонована для дослідження системи рекомендацій для шукачів роботи, є трирівневою. Рівень даних — це комбінація двох наборів даних, тобто даних користувача і даних списку вакансій. Рівень обробки складається з платформи Python/Intellij Idea, яка використовується для аналізу, моделювання та рекомендацій, а також бази даних. Також у дослідженні використовується третій рівень як рівень відображення системи, який складається з веб-фреймворку flask, який використовується для рекомендації найкращих вакансій за навичками користувача.

2.2. Spacy

spaCy є однією з популярних і простих у використанні бібліотек обробки природної мови в Python. spaCy допомагає створювати програми, які

можуть обробляти великі обсяги тексту та отримувати статистичні дані. Його можна використовувати для завдань, пов’язаних із вилученням інформації або системами розуміння природної мови, глибоким навчанням тощо. Такі відомі компанії, як Airbnb, Quora, Uber, використовують дану бібліотеку.

Основні особливості:

- Підтримує понад 64 мови.
- Вбудовані візуалізатори.
- Міцний і швидкий.
- Легко розширюється за допомогою спеціальних компонентів.
- Багатомовна підтримка, має моделі для систем з кількома мовами разом.

Ось деякі функції, які надає spaCy:

- Токенізація.
- Лематизація.
- Розпізнавання сутності.
- Розбір залежностей.
- Розпізнавання речень.
- Позначення частин мови.

Встановлення spaCy

spaCy можна встановити за допомогою менеджера пакунків python pip.

```
$ pip install spacy
```

Лістинг. 2.1 Встановлення spacy

Токенізація

У даній роботі використовується функція токенізації. Токенізація — це завдання поділу тексту на невеликі сегменти, які називаються токенами. Токенізація може відбуватися на рівні документа для створення токенів речень або токенізації речень для створення токенів слів, або токенізації слів для створення токенів символів.

Токенізатор зазвичай є початковим кроком пайплайну попередньої обробки тексту та служить як вхідні дані для наступних операцій NLP, таких як стемінг, лемматизація, аналіз тексту, класифікація тексту тощо.

У Spacy процес поділу тексту на сегменти слів і знаків пунктуації виконується в кілька етапів. Текст обробляється зліва направо.

- По-перше, токенізатор розділяє текст на пробіли подібно до функції split().
- Потім токенайзер перевіряє, чи відповідає підрядок правилам винятків токенайзера. Наприклад, «U.K.» завжди повинен залишатися цільним токеном.
- Далі він перевіряє наявність префікса, суфікса чи інфікса в підрядку, до них належать коми, крапки, дефіси чи лапки. Якщо знайдений збіг, підрядок розбивається на два токени.

У розробленій системі користувач вводить власні навички через пробіл. Для обробки тексту, спочатку імпортується бібліотека Spacy та завантажується англомовна модель spacy. Потім створюється об’єкт nlp, який буде використовуватись для аналізу тексту. Також створюється об’єкт doc, який буде містити оброблений текст. Отримані дані передаються у базу даних.

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
type(text)
dict = [token.text for token in doc]
```

Лістинг. 2.2 Токенізація

2.3. Flask

Flask — це невеликий і легкий веб-фреймворк Python, який надає корисні інструменти та функції, які полегшують створення веб-додатків на Python. Він надає можливість швидко створити веб-програму, використовуючи лише один файл Python. Flask також є розширюваним і не нав’язує певну структуру каталогів та не вимагає складного шаблонного коду перед початком роботи. Flask використовує механізм шаблонів Jinja для динамічного створення HTML-сторінок за допомогою знайомих концепцій Python, таких як змінні, цикли, списки тощо.

Переваги flask:

- Можливість експериментувати зі своїми модулями чи архітектурою.
- Підходить для невеликих проектів.
- Пропонує вбудований сервер розробки та швидкий налагоджувач.
- Легко масштабується для програм.
- Підтримка безпечних файлів cookie.

Flask можна встановити за допомогою менеджера пакунків python pip.

```
$ pip install flask
```

Лістинг. 2.3 Встановлення flask

На малюнку нижче показано використання flask у роботі. Причому відбувається підключення до серверу http://127.0.0.1:5000/

Мал. 2.3 Використання Flask

2.4. Pandas

pandas — це бібліотека програмного забезпечення, написана на мові програмування Python для обробки та аналізу даних. Зокрема, вона пропонує структури даних і операції для роботи з числовими таблицями та часовими рядами.

Переваги:

- Швидка і ефективна для обробки та аналізу даних.
- Можна завантажувати дані з різних файлових об’єктів.

- Легка обробка відсутніх даних (представлених як NaN) як у даних із плаваючою комою, так і без неї.
- Змінність розміру: стовпці можна вставляти та видаляти з DataFrame та об’єктів більшої розмірності.
- Злиття та об’єднання набору даних.
- Гнучка зміна форми та поворот наборів даних.
- Забезпечує функції часових рядів.
- Потужне групування за функціями для виконання операцій розділення, застосування та об’єднання наборів даних.

pandas можна встановити за допомогою менеджера пакунків python pip.

```
$ pip install pandas
```

Лістинг. 2.4 Встановлення pandas

2.5. MongoDB

MongoDB — це кросплатформна документоорієнтована база даних, яка працює на концепції колекцій і документів. MongoDB пропонує високу швидкість, високу доступність і високу масштабованість.

Переваги MongoDB:

- Підтримка ієрархічної структури даних.
- Підтримка асоційованих масивів, такі як словники в Python.
- Вбудовані драйвери Python для підключення програми python до бази даних. Приклад - PyMongo.
- Розроблений для великих даних.

- Розгортання MongoDB дуже просте.

Для встановлення MongoDB потрібно виконати наступну команду:

```
$ python -m pip install pymongo
```

Лістинг. 2.5 Встановлення pymongo

Створення бази даних для системи

Спочатку потрібно надати mongodb atlas url для підключення python до mongodb використовуючи pymongo:

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

uri =
"mongodb+srv://tanyavasylieva01:<password>@cluster
0.ktszquy.mongodb.net/?retryWrites=true&w=majority
"
```

Лістинг. 2.6 Підключення до mongodb

Далі створюється підключення, використовуючи MongoClient, що був імпортований у файл.

```
client =
MongoClient(uri,server_api=ServerApi('1'))
```

Лістинг. 2.7 Використання MongoClient

Потім відбувається створення бази даних jobs та запис оброблених даних вакансій у колекцію records з використанням методу insert_many().

```
db = client["jobs"]
db.records.insert_many(data)
```

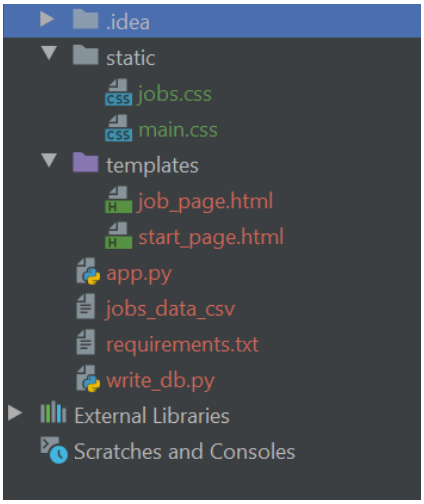
Лістинг. 2.8 Запис вакансій у базу даних

3. Розділ 3. Впровадження системи

У попередніх розділах було обговорено всю методологію для впровадження системи рекомендацій. У цьому розділі обговорюватиметься те, як було виконано впровадження за допомогою фреймворку CRISP-DM.

3.1. Структура проекту

Проект написаний у середовищі Intellij Idea. Папка static містить CSS файли, що мають стилі для відображення вмісту системи. Папка templates містить HTML-файли, які використовуються для розмітки вмісту системи у веб-браузері. Файл app.py містить код для отримання інформації від користувача, отримання релевантних вакансій на основі коефіцієнту Жакарра та їнього відображення за спаданням оцінки релевантності. А також підключення до серверу. Файл write_db.py містить код для підключення до бази даних, а також запису в базу датасету з вакансіями.



Мал. 3.1 Структура проекту

3.2. Джерела даних

В даній роботі використовуються дані вакансій та дані користувача про його навички.

Дані вакансій та обробка даних

Для отримання даних про вакансії використано датасет job_data_csv.csv з ресурсу Kaggle. У ньому наявні такі стовпці даних: _id, experience, location, salary, search_type, skills, title, url.

Мал. 3.2 Датасет з вакансіями

Перш, ніж почати роботу з датасетом, була виконана попередня його обробка та перетворення деяких даних типу string, у масив string значень, використовуючи pandas.

```
df = pd.read_csv("jobs_data_csv")

df['skills'] = df['skills'].apply(ast.literal_eval)
df['location'] = df['location'].apply(ast.literal_eval)

data = df.to_dict(orient="records")
```

Мал. 3.3 Робота з датасетом вакакнсій

Дані користувача та їх обробка

Дані користувача про його навички отримуються безпосередньо шляхом введення інформації від користувача. Вони обробляються з використанням NLP, а саме функції токенізації, яка була розглянута у попередньому розділі. Отримані дані передаються в базу даних.

```python
def retrieve_user_skills(text):
    nlp = spacy.load("en_core_web_sm")

    doc = nlp(text)

    type(doc)

    dict = [token.text for token in doc]

    return retrieve_vacancies(dict)
```

Мал. 3.4 Отримання інформації від користувача

3.3. Отримання релевантних вакансій

На малюнку нижче наведена функція, що відповідає за отримання вакансій, релевантних навичкам користувача на основі коефіцієнту Жаккара. Використання даного коефіцієнту дозволяє співставити введені навички користувача з навичками, вказаними у вакансії. Чим більше навичок співпадають, тим більш релевантною буде вакансія.

```python
def retrieve_vacancies(skills_list):
    length_skills_list = len(skills_list)
    num = mydb.find({"skills": {"$in": skills_list}})
    vacancies = []
    for i in num:
        vacancy_skills = i['skills']

        match = len([k for k, val in enumerate(vacancy_skills) if val in skills_list])

        total_length = len(vacancy_skills) + length_skills_list
        # ranking coefficient
        i['rank'] = match / total_length
        vacancies.append(i)

    return show_info(vacancies, skills_list, len(vacancies))
```

Мал. 3.5 Пошук релевантних вакансій в базі даних та обрахування коефіцієнту Жаккара

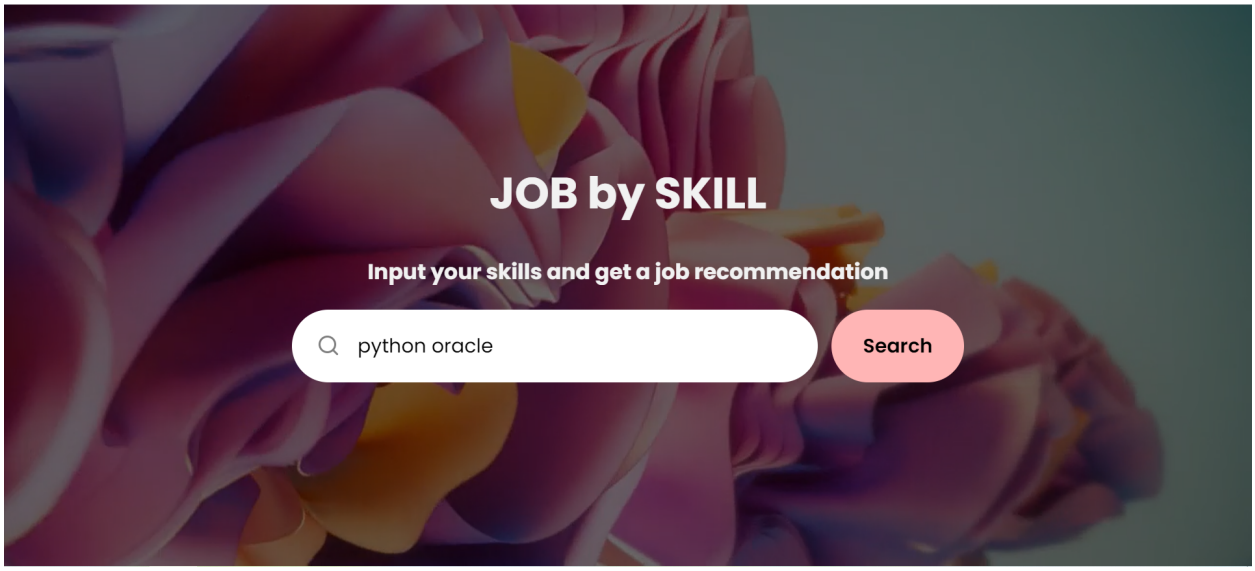
Спочатку отримуються всі релевантні вакансії, далі обраховується коефіцієнт релевантності та відбувається сортування вакансій за спаданням значення цього коефіцієнту.

```
def show_info(vacancies, vacancies_skills, vacancies_length):
    # sort jobs based on rank
    vacancies.sort(key=operator.itemgetter('rank'), reverse=True)
    return render_template('job_page.html', jobs=vacancies,
                           job_skills=vacancies_skills, job_len=vacancies_length)
```

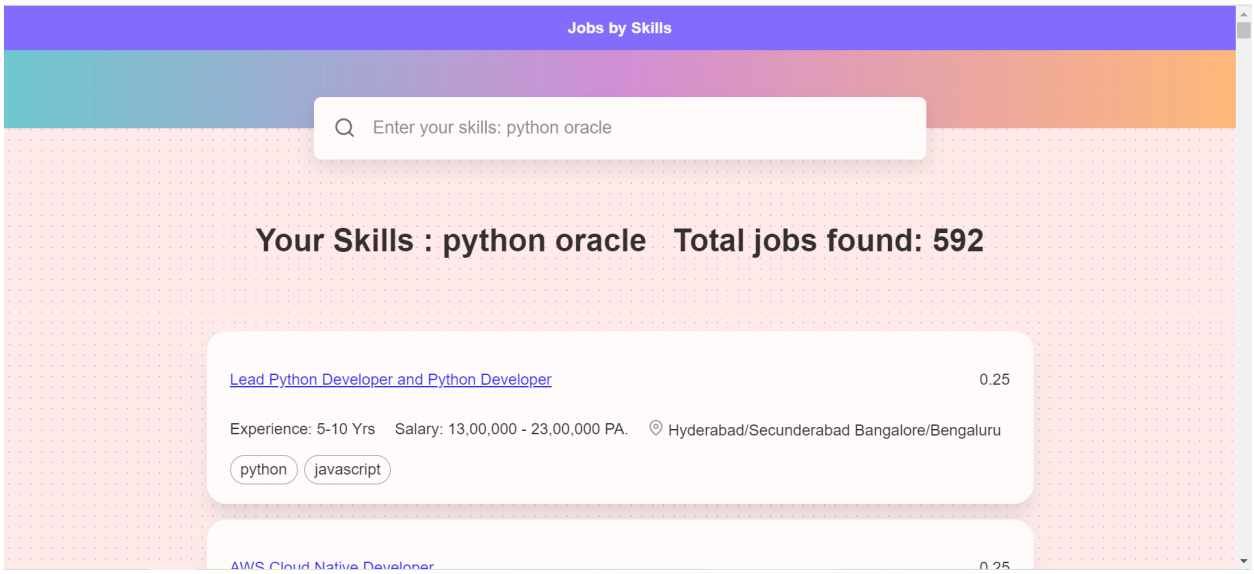
Мал. 3.6 Функція для відображення вакансій

3.4. Відображення системи

Для відображення системи у браузері використовуються HTML, CSS та Flask. Готовий прототип системи має дві сторінки та виглядає наступним чином:



Мал. 3.7 Початкова сторінка сервісу



Мал. 3.8 Сторінка з релевантними вакансіями, що відповідають запиту

4. Висновки

У даній роботі розроблено систему рекомендацій вакансій за навичками користувача, що має назву Jobs by Skills, досліджено особливості побудови рекомендаційних систем, продемонстровано процес розробки, особливості архітектури сервісу та його роботи.

Можливі покращення отриманої системи:

- Додати більш повний аналіз тексту користувача, а саме можливість вводити навички у будь-якому форматі.
- Розбити отриманий список вакансій на сторінки та додати нумерацію.
- Додати фільтрацію контенту.

Однак базовий концепт був реалізований, а саме досліджено та розроблено власну систему рекомендацій вакансій на основі навичок користувача.

5. Список використаної літератури

1. Jobs on Naukri.com [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/PromptCloudHQ/jobs-on-naukricom
2. How To Make a Web Application Using Flask in Python 3 [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3
3. Коефіцієнт Жаккара [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B5%D1%84%D1%96%D1%86%D1%96%D1%94%D0%BD%D1%82_%D0%96%D0%B0%D0%BA%D0%BA%D0%B0%D1%80%D0%B0
4. What is Flask Python [Електронний ресурс] – Режим доступу до ресурсу: https://pythonbasics.org/what-is-flask-python/
5. What is natural language processing (NLP)? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/natural-language-processing
6. How to Use Python with MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/languages/python
7. Introduction to Pandas in Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-to-pandas-in-python/
8. CRISP-DM: Towards a Standard Process Model for Data Mining [Електронний ресурс] – Режим доступу до ресурсу: http://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf