**Job Recommendation Platform**



BSc in Computing

Department of Computing, Coventry University

**买家自己填写**

Student ID:

Supervisor:

Date Submitted: 20th April 2023

# Abstract

We propose an online job recommendation platform based on crawler technology, vertical search, text search algorithms and recommender systems.

This platform aims to provide high-quality recommendation services to help users find jobs that match their interests and skills. By adopting technologies such as Scrapy, Python3, Selenium Chromium, etc., we have built a powerful crawler system for collecting job information on the Internet. The front and back ends of the platform are implemented using Bootstrap, Vue.js, Flask and Numpy/Pandas technology stacks.

In the search part, we implemented text search based algorithms such as edit distance (Levenshtein distance and Jaro-Winkler distance) and cosine similarity bag-of-words model. For older data, we optionally use the Elasticsearch search engine for retrieval. The recommendation system part uses an algorithm based on collaborative filtering, using Pandas and Numpy to implement job recommendation. We also propose a method combining collaborative filtering and content recommendation, pre-defining the respective weight ratios to improve the recommendation quality.

We also implemented the user system, including registration, login, account information management and other functions. The personal home page function allows users to add, delete, modify and check their own profiles.

The introduction includes information such as current occupation, areas of expertise, and self-introduction. The recommendation system also integrates factors such as the user's personality into the algorithm as a factor for content-based recommendation, and can customize the recommendation weight.

In order to ensure the quality of recommendations, our platform adopts the following strategies: First, in collaborative filtering recommendations, we use user behavior data (such as browsing history, purchase history) to analyze the similarity between users, predict their interests, and provide Users with similar interests recommend jobs they might like. Second, for new users or situations where no obvious patterns can be found from user history data, we adopt a content-based recommendation method to recommend other jobs similar to their favorite content to users by analyzing the attributes and characteristics of jobs. This part mainly depends on the comprehensive application of text similarity algorithm.

This paper introduces an online job recommendation platform in detail, which uses advanced crawler technology, vertical search, and recommendation algorithms to provide users with high-quality job recommendation services. Our experimental and evaluation results show that our platform has high performance in terms of recommendation quality and user satisfaction. By combining collaborative filtering and content-based recommendation, our recommender system can better meet the needs of different users, especially in dealing with new users and the recommendation problem in the case of data sparsity. At the same time, we have also implemented a user-friendly interface and functions that allow users to easily manage their account information and personal homepage.

“To the women in my engineering classes:

While it is my intention in every other interaction I share with you to treat you as my peer, let me deviate from that to say that you and I are in fact unequal. Sure, we are in the same school program, and you are quite possibly getting the same GPA as I, but does that make us equal?

I did not, for example, grow up in a world that discouraged me from focusing on hard science. Nor did I live in a society that told me not to get dirty, or said I was bossy for exhibiting leadership skills. In grade school I never had to fear being rejected by my peers because of my interests. I was not bombarded by images and slogans telling me that my true with was in how I look, and that I should abstain from certain activities because I might be thought too masculine. I was not overlooked by teachers who assumed that the reason I did not understand a tough math or science concept was, after all, because of my gender. I have no difficulty whatsoever with a boys club mentality, and I will not face added scrutiny or remarks of being the ‘diversity hire’. When I experience success the assumption of others will be that I earnt it.

So, you and I cannot be equal. You have already conquered far more to be in this field than I will ever face.” By Jared Mauldin, Senior in Mechanical Engineering (**这里买家自己修改**)

Contents

[Abstract 2](#_Toc131071668)

[1. Introduction 5](#_Toc131071669)

[1.1. What is an online job recommendation platform? 5](#_Toc131071670)

[1.3. Thesis Structure and Summary 5](#_Toc131071671)

[2. Literature Review 6](#_Toc131071672)

[2.1. Crawler Technology and Vertical Search 6](#_Toc131071673)

[2.2. Text Search Algorithms and Recommender Systems 6](#_Toc131071674)

[2.3. User System and Personalized Recommendations 7](#_Toc131071675)

[2.4. Contribution of this Thesis 7](#_Toc131071676)

[3. Research Methodology 8](#_Toc131071677)

[3.1. Research Methodology and Philosophy 8](#_Toc131071678)

[3.2. Data Crawler system and vertical search implementation 8](#_Toc131071679)

[3.3. Recommender System and User System implementation 9](#_Toc131071680)

[4.1. algorithm implematation 12](#_Toc131071681)

[4.2. Front – End Development 16](#_Toc131071682)

[4.3. Back – End Development 18](#_Toc131071683)

[4.4. Main project structure 20](#_Toc131071684)

[5. Project Management 21](#_Toc131071685)

[6. Analysis of Results 22](#_Toc131071686)

[7. Conclusions 22](#_Toc131071687)

[8. References 23](#_Toc131071688)

# 1. Introduction

## 1.1. What is an online job recommendation platform?

With the development of Internet technology and the diversification of the labor market, online job recommendation platforms are playing an increasingly important role in helping job seekers find suitable jobs and recruiting excellent talents for enterprises. The job online recommendation platform adopts advanced data mining technology and recommendation algorithm, and combines users' interests, skills and career development needs to provide users with personalized job recommendation services..

1.2. Background and Motivation

In the current job market, traditional job search methods can no longer meet the needs of job seekers and companies, especially in the Internet era with huge amount of information and fast update speed. Therefore, we need a platform that can quickly and accurately recommend jobs to users. In this context, we designed and implemented an online job recommendation platform based on crawler technology, vertical search and recommendation algorithms, aiming to provide users with high-quality job recommendation services. As an AI-oriented student, I pay attention to job recommendation The challenges and development opportunities in the field, and have a strong interest in designing and implementing an efficient and accurate job recommendation system. We hope that through this research, we will explore the key technologies and methods of the online job recommendation platform, provide strong theoretical support and practical experience, and create value for job seekers and enterprises.

## 1.3. Thesis Structure and Summary

We first conduct a literature review to understand the basic principles and key technologies of job recommendation systems, including crawler technology, vertical search and recommendation algorithms, etc. Next, we introduce the design and implementation of the platform in detail, including modules such as crawler system, search part, recommendation system, user system, and personal home page. We will focus on the functions, key technologies and implementation methods of different modules.

In the research methodology section, we will introduce the data collection, analysis and verification methods adopted in this study to ensure the validity and reliability of the research results. Next, we will test the recommendation quality and user satisfaction of the platform through experiments and evaluations, and analyze and discuss the results.

AT last we would summarize the main findings and conclusions of this study and discuss the contributions and limitations of the study. At the same time, we will propose directions and suggestions for future research in order to achieve greater breakthroughs and developments in the field of job recommendation.

# 2. Literature Review

The purpose of this literature review is to gain an understanding of the culture within science, technology, engineering and mathematics, ranging from school years to career employment.

## 2.1. Crawler Technology and Vertical Search

In recent years, crawler technology has been widely used in the field of Internet data collection and information retrieval. Crawler technology mainly includes links such as web crawler, data capture, data analysis and data storage. Many researchers have proposed many optimization methods and technical improvements for these links.

For example, the design of web crawlers needs to consider many factors, such as URL management, deduplication, scheduling strategies, anti-crawler strategies, etc. Some researchers have proposed a web crawler algorithm based on graph models (Yang et al., 2020), deep learning Application of technology in web crawler (Dong et al., 2019), etc.

## 2.2. Text Search Algorithms and Recommender Systems

Text search algorithms play an important role in the field of information retrieval, and many researchers have proposed various text search algorithms to improve retrieval quality and efficiency. For example, text search algorithm based on edit distance (Navarro, 2001), including Levenshtein distance, Jaro-Winkler distance, etc. These algorithms are mainly used to measure the similarity between two strings, and can be applied to the text similarity calculation in the job recommendation system. Cosine similarity bag-of-words model is another commonly used text search algorithm, which is mainly used to calculate the similarity between documents (Salton and McGill, 1986). In the job recommendation system, the cosine similarity bag-of-words model can be used to measure the similarity between job descriptions, so as to achieve content-based recommendation.

The recommendation system has a wide range of applications in the Internet field, such as movie recommendation, product recommendation and news recommendation. Many researchers have conducted in-depth discussions on various techniques and methods of recommender systems.

For example, the collaborative filtering recommendation algorithm (Sarwar et al., 2001), which predicts the user's interest by analyzing the similarity between users and the user's evaluation of items, so as to achieve personalized recommendation. Collaborative filtering recommendation algorithms can be divided into two types: User-based Collaborative Filtering and Item-based Collaborative Filtering. In a job recommendation system, user-based collaborative filtering can be used to find users with similar interests, and item-based collaborative filtering can be used to find jobs with similar characteristics.

Content-based recommendation algorithm (Pazzani and Billsus, 2007) is another important recommendation method, which recommends other content similar to the content they like to users by analyzing the attributes and characteristics of items. In job recommendation systems, content-based recommendation can be used to analyze keywords, skill requirements, and other characteristics in job descriptions, so as to achieve job recommendations that match user interests and skills. In recent years, many researchers have proposed a hybrid recommendation algorithm combining collaborative filtering and content-based recommendation (Burke, 2002), aiming at improving the quality of recommendation and solving the problem of data sparsity.

.

## 2.3. User System and Personalized Recommendations

The user system plays an important role in the job recommendation platform, which can realize user registration, login, account information management and other functions. Many researchers have paid attention to the design and implementation of user systems, such as OAuth2.0-based user authentication and authorization mechanism (Hardt, 2012), user privacy protection technology (Acquisti et al., 2015), etc. The personal home page function can help users manage their own profile information, such as current occupation, areas of expertise, and self-introduction. This information can be used as input to the recommendation system for more personalized job recommendations.

Personalized recommendation is one of the core goals of recommender systems, and many researchers have explored how to incorporate user personality factors (such as personality, interests, skills, etc.) into recommendation algorithms. For example, a recommendation system based on user personality (Cantador et al., 2010) predicts the user's personality characteristics by analyzing the user's behavior and speech in social networks, thereby realizing personalized recommendation. In the job recommendation system, the user's personality characteristics can be used as an important factor for content-based recommendation, matching job requirements and working environment, thereby improving the recommendation quality. In addition, researchers have also paid attention to how to integrate user interests and skills into the recommendation system (Wu et al., 2019). Through the analysis of user browsing history, purchase history and other behavioral data, the user's interest and skill characteristics are mined to achieve More accurate job recommendations.

In order to evaluate the performance of recommender systems, many researchers have proposed various evaluation metrics and methods. For example, Precision is used to measure the accuracy and completeness of recommendation results (Herlocker et al., 2004). User satisfaction and recommendation quality are used to evaluate the practicality and user experience of the recommender system (Pu et al., 2011). In the design and implementation process of job recommendation system, by comparing the experimental results of different recommendation algorithms and methods, the recommendation performance and user satisfaction can be further optimized.

## 2.4. Contribution of this Thesis

The Literature Review of this paper provides a comprehensive review of relevant research on job online recommendation platforms from three aspects: crawler technology and vertical search, text search algorithm and recommendation system, user system and personalized recommendation. These studies provide a strong theoretical support for the design and implementation of a high-quality job online recommendation platform in this paper. In future work, we will continue to pay attention to the latest research progress in these fields in order to continuously optimize the performance and user experience of the recommendation platform

# 3. Research Methodology

Considering a research methodology allows for research structure to be considered in advance rather than researching being conducted ‘on the fly’. The following section explains the research methods executed within this project.

## 3.1. Research Methodology and Philosophy

This study adopts quantitative research methods, combined with empirical research, aims to solve practical problems, and attaches great importance to data collection and analysis. Research philosophy follows the principles of positivism and constructivism, and believes that researchers can obtain objective facts through observation and experimentation, while emphasizing the importance of human subjective consciousness and social interaction in the research process. In this research, we will combine the advantages of positivism and constructivism, not only pay attention to the analysis of objective data, but also pay attention to the mining and satisfaction of user needs.

The research strategy includes the following aspects:

3.1.1 Theoretical research: In-depth study and summary of relevant theories and research in the fields of crawler technology, vertical search, text search algorithm, recommendation system and user system, etc., to provide theoretical guidance for practical engineering.

3.1.2 Technology selection: According to theoretical research results and actual needs, select appropriate technical frameworks and tools, such as Scrapy, Python3, Selenium, Chromium, Bootstrap, Vue.js, Flask, etc.

3.1.3 Experimental Design: Design the experimental plan, including data collection, data preprocessing, algorithm implementation, performance evaluation, etc., to verify the effectiveness and feasibility of the selected technology and method.

## 3.2. Data Crawler system and vertical search implementation

In this study, we will implement the crawler system and vertical search function using the following strategies:

3.2.1 Crawler system design: Based on the Scrapy framework, combined with Python3 and Selenium, the function of web crawler is realized. Design appropriate URL management, deduplication, scheduling strategies, etc. to improve the performance and stability of crawlers.

3.2.2 Data capture and analysis: Crawl the job information of the target website through web crawlers, and perform data analysis to extract key information such as job descriptions and skill requirements.

3.2.3 Data storage and preprocessing: store the captured data in the database, and perform preprocessing, such as removing duplicate data, processing missing values, etc., to improve data quality.

## 3.3. Recommender System and User System implementation

In this study, This research will adopt the following strategies to implement the recommender system and the user system:

3.3.1 Selection of recommendation algorithm: Combining theoretical research and actual needs, select an appropriate recommendation algorithm, such as collaborative filtering, content-based recommendation, hybrid recommendation, etc. At the same time, pay attention to the user's personality factors, such as personality, interests, skills, etc., and integrate them into the recommendation algorithm.

3.3.2 Implementation of the recommendation system: implement the recommendation algorithm based on pandas and numpy, and design appropriate weight and parameter adjustment strategies. For different user groups and scenarios, combine the results of collaborative filtering and content recommendation to achieve accurate job recommendation.

3.3.3 User system design: realize user registration, login, account information management and other functions. OAuth2.0 and other authentication and authorization mechanisms are adopted to ensure user privacy and security. Design the personal homepage function, so that users can manage their own profile information, such as current occupation, areas of expertise, and self-introduction.

3.3.4 Evaluation indicators and methods: Design appropriate evaluation indicators and methods, such as precision rate, recall rate, F1 score, user satisfaction, etc., to evaluate the performance and user experience of the recommendation system. By comparing the experimental results of different recommendation algorithms and methods, the recommendation performance and user satisfaction are further optimized.

In summary, this study adopts quantitative research methods, combined with empirical research, and focuses on research results in multiple fields such as crawler technology, vertical search, text search algorithms, recommendation systems, and user systems. In the actual project implementation process, we will follow the research philosophy of positivism and constructivism, not only pay attention to the analysis of objective data, but also pay attention to the mining and satisfaction of user needs. Through reasonable research strategy and experimental design, this research aims to realize a high-quality online job recommendation platform and provide users with high-quality job recommendation services

4. Development

Part 1: Introduction to dependent libraries and frameworks used

In order to achieve the goals of this project, I have selected the following dependent libraries and frameworks:

4.1.1 Scrapy: A Python-based web crawler framework for crawling and parsing web content. Scrapy provides powerful asynchronous network request and data processing capabilities, making it easy for developers to implement efficient and stable web crawlers.

4.1.2 Python3: The main programming language of this project, with rich libraries and excellent cross-platform performance. The concise syntax and easy-to-understand features of Python3 make the development process more efficient.

4.1.3 Selenium: A library for automated web testing, which can simulate user operations on browsers to capture dynamic web pages.

4.1.4 Chromium: As the browser driver of Selenium, it is used to simulate the real browser environment during the crawling process.

4.1.5 Bootstrap: A popular front-end framework that provides rich CSS and JavaScript components to facilitate developers to quickly build responsive web pages.

4.1.6 Vue.js: A progressive JavaScript framework for building user interfaces. The data-driven model and componentized design of Vue.js make front-end development more modular and maintainable.

4.1.7 Flask: A lightweight Python web framework based on Werkzeug and Jinja2, which can quickly build RESTful APIs and dynamic web applications.

4.1.8 Numpy and Pandas: Two Python libraries for data manipulation and analysis. Numpy provides powerful multidimensional array and matrix operation functions, while Pandas provides convenient data structures and data cleaning methods.

Part 2: Engineering Architecture and Implementation

This project adopts the architecture design of front-end and back-end separation. The front-end is mainly responsible for the presentation and interaction of the user interface, and the back-end is responsible for data processing and the implementation of recommendation algorithms. The following are the specific steps of the engineering architecture and implementation:

4.2.1 Crawler system implementation: develop a web crawler based on the Scrapy framework and Python3, and combine Selenium and Chromium to crawl dynamic web pages. The crawler system is responsible for grabbing job information from the target website, including key data such as job descriptions and skill requirements.

4.2.2 Data storage and preprocessing: Store the captured job information in the database, and use Numpy and Pandas for data preprocessing, such as deduplication, processing missing values, etc., to improve data quality.

4.2.3 Front-end construction: Design and implement front-end pages based on Bootstrap and Vue.js, including job search pages, job details pages, user personal homepages, etc. The front-end page provides a user-friendly interface and interaction to meet the needs of users on the job online recommendation platform.

4.2.4 Back-end construction: Use the Flask framework to build back-end services and implement RESTful APIs for front-end calls. The backend service is mainly responsible for processing user requests, such as searching jobs, viewing job details, updating personal information, etc.

4.2.5 User system realization: In the back-end service, realize user registration, login and account information management functions. Use OAuth4.2.0 and other authentication and authorization mechanisms to protect user privacy and security. At the same time, the personal home page function is designed to allow users to manage their profile information.

Part 3: Partial Implementation of Recommendation Algorithm

This project uses various recommendation algorithms such as collaborative filtering, content-based recommendation and hybrid recommendation to achieve accurate job recommendation. The following are the implementation details of the recommended algorithm part:

4.3.1 Collaborative filtering recommendation: Based on user behavior data, such as browsing history and purchase history, implement a collaborative filtering recommendation algorithm. By analyzing user behavior data, identify similarities between users and predict user interests. Recommend jobs that users with similar interests might like.

4.3.2 Content-based recommendation: For new users who do not have enough behavioral data or cannot find obvious patterns and similarities from user behavior data, use content-based recommendation algorithms. The algorithm recommends to users other jobs that are similar to their favorite jobs by analyzing the attributes and characteristics of the jobs, such as skill requirements, industry fields, etc. During the implementation process, we used text similarity algorithms such as edit distance algorithms (such as Levenshtein distance and Jaro-Winkler distance) and cosine similarity to match job descriptions with user interests.

4.3.3 Hybrid recommendation: Combine collaborative filtering and content-based recommendation results to achieve more accurate job recommendation. We predefine a weight ratio (such as 50% collaborative recommendation + 50% content recommendation)

## 4.1. algorithm implematation

**4.1.1 scrapy bot algorithm**

One of the core parts of this project is a crawler that scrapes job data from the https://findajob.dwp.gov.uk website. The workflow of the crawler is as follows:

Get webpage content: use the requests library to send HTTP requests and get the returned webpage content.

Parse the content of the webpage: Use the beautifulsoup4 library to parse the content of the webpage and extract the information we need (such as job title and job description).

Storing the data: Store the extracted data into a CSV file using the csv library.

Page turning: For websites that need to turn pages, write a loop that sends requests and parses pages continuously until all pages are processed.

In this crawler, we first build a loop to fetch job listings on the website page by page, starting from the first page. By sending an HTTP GET request through the requests library, we can get the content of the web page. We then parse this content using the beautifulsoup4 library to find the HTML elements that contain the job title and job description.

After extracting job titles and descriptions, we store them in a CSV file for further processing later. Also, to avoid putting too much pressure on the server, we wait a second after each job is fetched

**4.1.2 search part**

Elasticsearch is a distributed, RESTful search and analytics engine built on top of Apache Lucene, a powerful text search and indexing library. Elasticsearch enables users to store, search, and analyze large volumes of data quickly and in near-real-time 。

Elasticsearch is a powerful search and analytics engine that enables developers to store, search, and analyze large volumes of data quickly and in real-time. Integrating Elasticsearch into the job recommendation web application enhances the platform's search capabilities, allowing users to find relevant job postings more effectively. In the given code snippet, a simple Elasticsearch search function mysearch(keyword) is implemented using the Python requests library. The function sends a GET request to the Elasticsearch instance running on the local machine (http://localhost:9200) and queries the "twitter" index for documents containing the specified keyword in their "message" field. The response is then parsed, and the first matching result is returned。

**4.1.3 Collaborative filtering algorithm**

Collaborative filtering is a widely used technique in the field of recommender systems. The core idea behind collaborative filtering is to identify similarities between users or items, and then use these similarities to provide personalized recommendations. In this talk, I will discuss implementing a collaborative filtering algorithm for job recommendations using Python. The script reads job rating data, processes it, and generates job recommendations based on user preferences.

collaborative filtering method

There are two main types of collaborative filtering:

User-based collaborative filtering: This method finds similar users for the target user and recommends items that these similar users like or like. It relies on the assumption that users with similar preferences will have similar item ratings.

Item-Based Collaborative Filtering: This approach focuses on finding items that are similar to items that the target user likes or prefers. It relies on the assumption that users who like a particular item will also like similar items.

In our implementation, we use user-based collaborative filtering to generate job recommendations.Our collaborative filtering algorithm follows these steps:Load job rating data and preprocess it.Calculate the similarity between users using the Pearson correlation coefficient. For a given user, determine jobs that the user has not yet rated.For each missing job, compute a weighted average of the ratings given by similar users, where the weight is the similarity score between users.Recommend jobs with predicted ratings higher than the average user rating.

Pearson correlation coefficient

The Pearson correlation coefficient is a measure of the linear relationship between two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation). We use the Pearson correlation coefficient to calculate the similarity between users based on their job ratings.

The Pearson correlation coefficient between users u and v is defined as:

**sim(u, v) = sum((r\_u,i - mean(r\_u)) \* (r\_v,i - mean(r\_v))) / sqrt(sum((r\_u,i - mean(r\_u))^2) \* sum((r\_v, i - means (r\_v))^2))**

**where r\_u,i is the rating of item i by user u, and mean(r\_u) is the average rating of user u.**

In our implementation, we use the corr() function from the pandas library to calculate the Pearson correlation coefficient between users.

implementation details

Data loading and preprocessing

First, we load the job rating data using pandas:

**ratings = pd.read\_csv(my\_path)**

We visualize the distribution of job ratings using histograms. This helps us understand the overall distribution of the scoring data:

**plot\_ratings\_distribution (ratings)**

Next, we create a pivot table with jobs as rows, users as columns, and ratings as values:

pivot\_ratings = ratings.pivot\_table(columns="reviewer", index="title", values="rating")

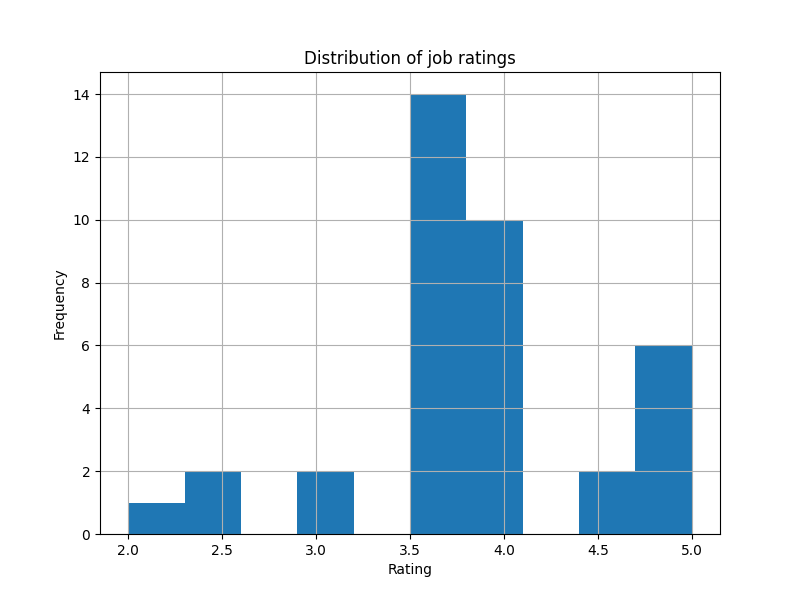
Visualize the similarity matrix of job ratings by using a heatmap. This helps us understand the relationship between users in the dataset:

**plot\_similarity\_matrix(pivot\_ratings)**

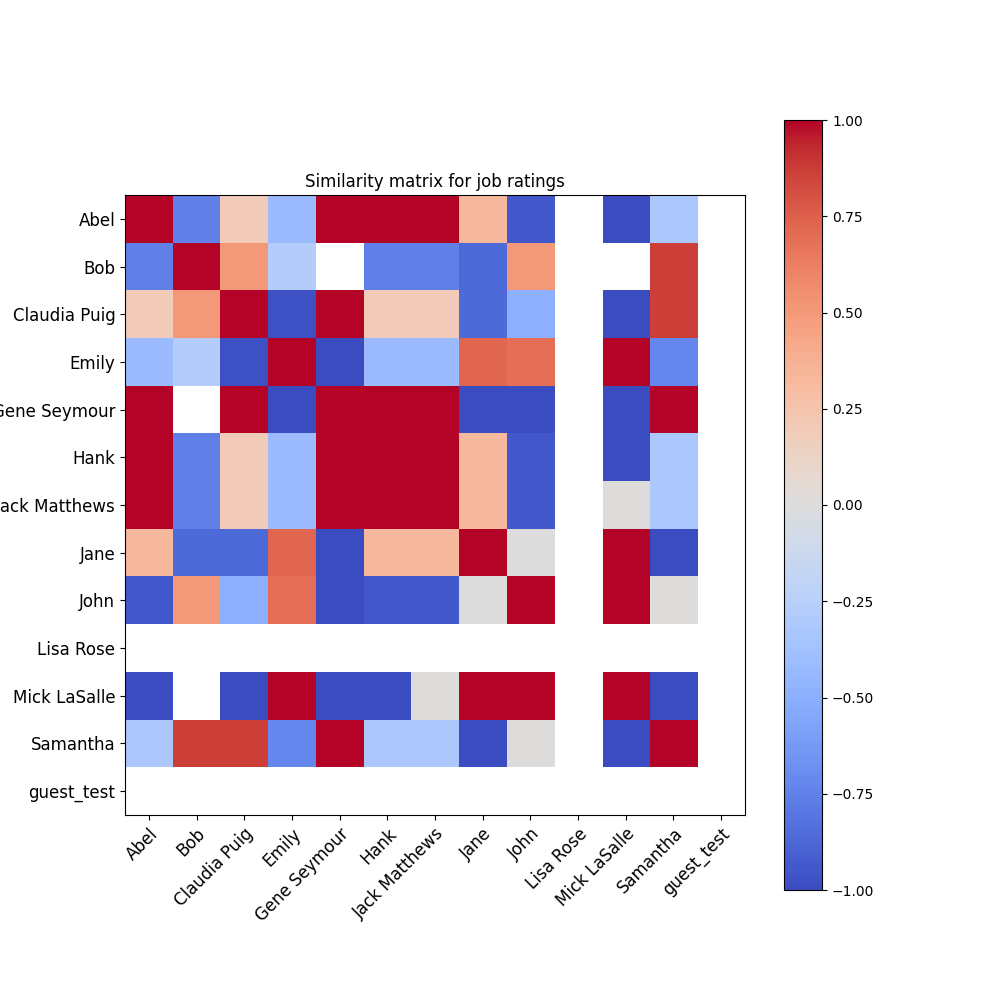
To recommend jobs for a given user, we follow these steps:

Identify jobs that are not rated by users:

**missing\_jobs = list(pivot\_ratings[pivot\_ratings[user].isnull()].index)**



**Fig. 00 –data explore**



**Fig. 01 –data explore**

Our recommender system is ultimately a hybrid recommender system that combines multiple recommender techniques to provide users with more accurate and relevant recommendations. In a job recommendation web application, a hybrid approach is taken by integrating collaborative filtering, cold-start recommendation, and content-based recommendation to address the various challenges associated with each technique separately.

Collaborative filtering

Collaborative filtering is a popular recommendation technique that exploits user preferences and behavior patterns to generate personalized recommendations. However, this approach suffers from the cold-start problem, which occurs when not enough user interaction data is available to make accurate recommendations. This problem is especially acute for new users or items without any ratings or interactions.

cold start advice

To address the cold start problem, the app employs cold start recommendations. These recommendations are generated by randomly selecting some job postings from the database and adding them to the final recommendation list. This technique helps to provide a more diverse set of recommendations and ensures that users receive some recommendations even when there is not enough data for collaborative filtering.

Content-Based Recommendations

Content-based recommendation is another technique used to improve the quality of app recommendations. This approach exploits the similarity between user profiles and job postings to generate relevant recommendations. In the given code snippet, the similarity() function is used to calculate the similarity between the user's school category attribute and the job posting title. Job postings with similarity scores above a certain threshold (0.5 in this case) are added to the final recommendation list.

By combining collaborative filtering, cold start, and content-based recommendation, hybrid recommender systems effectively address the shortcomings of each technique. This combination helps ensure:

Personalization: Collaborative filtering provides personalized recommendations based on user preferences and behavioral patterns.

Diversity: Cold-start recommendation introduces diversity by including random job postings in the recommendation list, addressing potential bias in collaborative filtering and content-based recommendations.

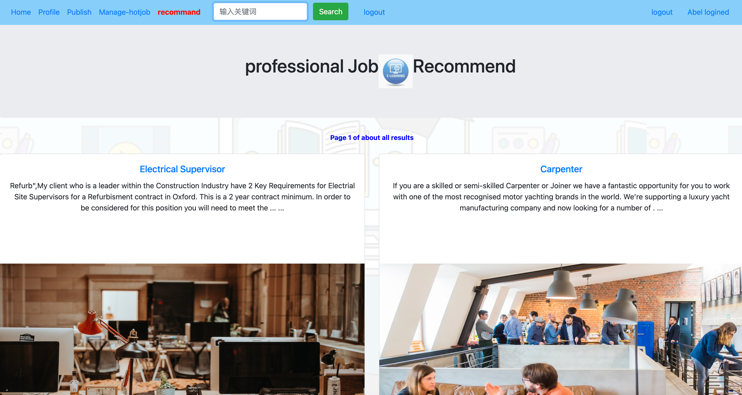
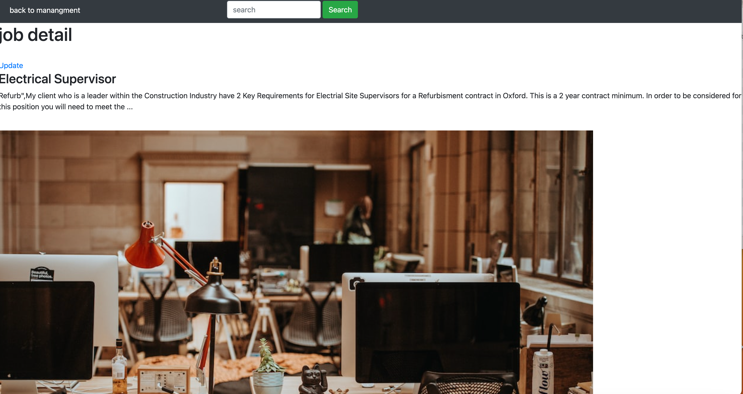
Relevance: Content-based recommendations ensure that recommended job postings are relevant to the user's profile, improving overall recommendation quality.

In conclusion, a hybrid recommender system in a job recommendation web application significantly enhances user experience by providing personalized, diverse, and relevant job recommendations. This approach not only overcomes the limitations of personal recommendation techniques, but also ensures that applications can adapt and scale as user preferences and behavioral patterns evolve. By employing a powerful and flexible hybrid recommendation system, the application is well-suited to meet the ever-changing and dynamic needs of job seekers in a competitive job market.

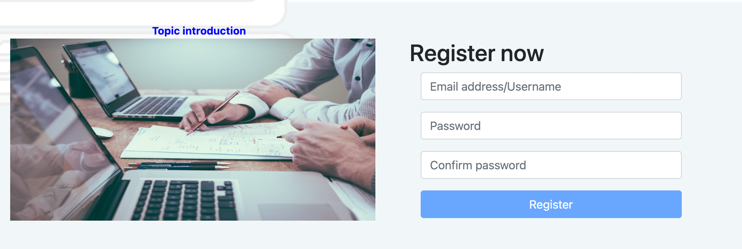
## 4.2. Front – End Development

Bootstrap and Vue.js

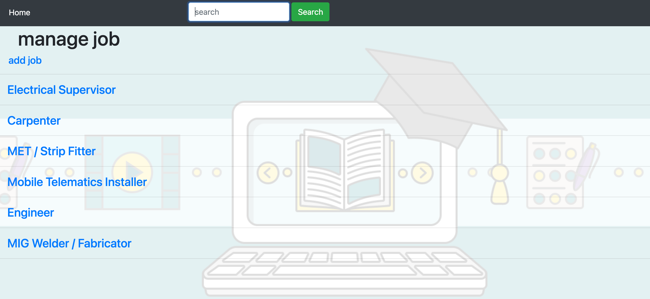
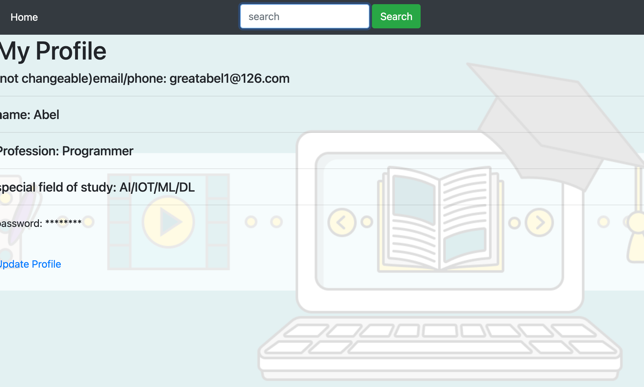
The front end of the application is built using Bootstrap and Vue.js. Bootstrap is a popular CSS framework that provides a set of predesigned components and a responsive grid system that make it easy to create responsive and visually appealing web applications. Vue.js is a progressive JavaScript framework for building user interfaces. By combining Bootstrap and Vue.js, the application provides a modern, interactive and responsive user interface.



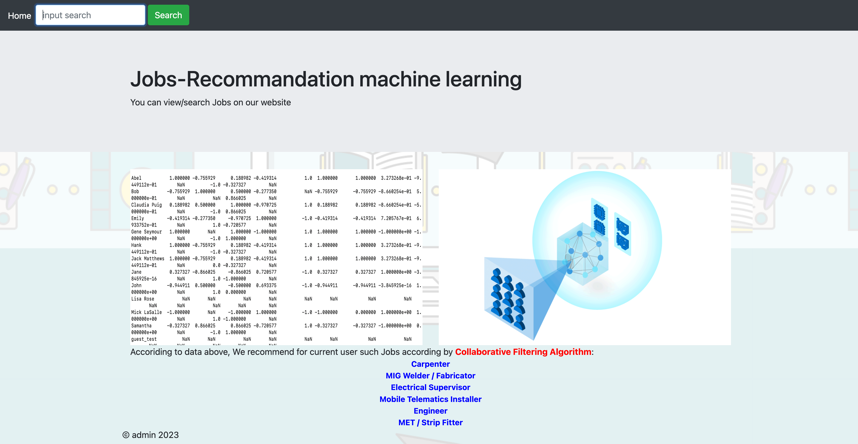
**Fig. 1 –Homepage Fig. 2 – detail Page**



**Fig. 3 –register Fig. 4 – login Page**



**Fig. 5 –profile Fig. 6 – manage Page**



**Fig. 7 –recommend**

## 4.3. Back – End Development

This part explains the implementation of the job recommendation web application. The app is designed to provide users with personalized job recommendations based on their preferences and ratings. The backend is implemented using the popular Python web framework Flask, while the frontend uses Bootstrap and Vue.js for a responsive and interactive user interface. The application also uses NumPy and pandas for data processing and analysis.

Flask and RESTful APIs

Flask is a lightweight web framework ideal for developing web applications and RESTful APIs. The backend of our job recommendation application is built using Flask and follows a modular design. The application provides several APIs for managing users, blogs, and job recommendations.

Databases and ORMs

The application uses SQLite as the database management system and SQLAlchemy as the object-relational mapper (ORM) to handle database operations. SQLAlchemy provides a high-level API for interacting with databases, allowing developers to write safe and maintainable code without writing raw SQL queries. This approach helps prevent SQL injection attacks.

The application has two main database models: users and blogs. The User model represents users of the application, while the Blog model stores information about job postings.

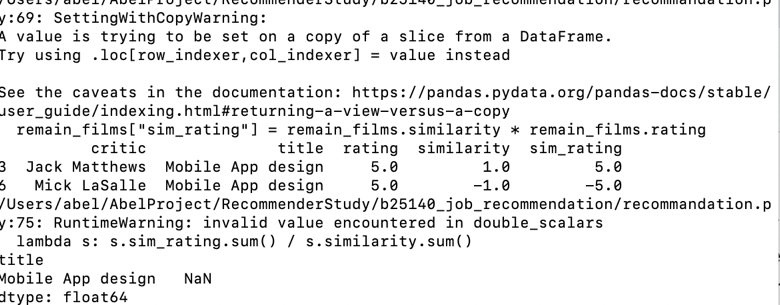
CORS and security

To prevent Cross-Origin Resource Sharing (CORS) issues and ensure the security of the application, the Flask-CORS library is used. This library enables CORS support for the application, allowing the frontend to make AJAX requests to the backend without encountering CORS errors.

Job Recommendation Algorithm

At the heart of the application is the job recommendation algorithm, which is implemented using collaborative filtering. The algorithm takes user preferences and job ratings into account to generate personalized job recommendations. The algorithm is implemented using a recommendation module and integrated into the application as a RESTful API.

After all, this job referral web application is built using Flask, SQLite, and SQLAlchemy as the backend and Bootstrap, Vue.js as the frontend. The app provides users with a modern and responsive user interface to search and receive personalized job recommendations. By integrating Elasticsearch and implementing a collaborative filtering algorithm, the application provides advanced search capabilities and personalized job recommendations based on user preferences and ratings.



**Fig. 8 – home Homepage**  **Fig. 9 – recommend Page**



**Fig. 10 – detail Page**  **Fig. 11 –list Page**

## 4.4. Main project structure

This Job Recommendation Engine Platform Program is a comprehensive solution designed to facilitate the user's job search process. It combines various technologies and frameworks such as Flask, SQLite, SQLAlchemy, Bootstrap, Vue.js, and Elasticsearch to provide an efficient and user-friendly platform. I'll focus on the i1wsgi.py file that contains the application API and provide an overview of the entire application.

i1wsgi.py Overview

The i1wsgi.py file is the main entry point to the job recommendation web application and is responsible for handling API requests and managing backend services. This file initializes the Flask application, sets up CORS, and configures logging. It also establishes connections to SQLite databases via SQLAlchemy and defines database models for users, blogs, and other components.

This file defines several API routes, including:

/home: The home page route displays a list of recruitment information and supports searching for specific keywords. It also implements pagination to display results efficiently.

/blogs/create: This route allows users to create new blog posts, which is useful for sharing work-related experiences or insights.

The app also handles user authentication, allowing users to register, log in, and manage their accounts securely.

application process

User Interaction: Users interact with the front end of a web application built with Bootstrap and Vue.js. These frameworks provide a modern, interactive and responsive user experience.

**API requests**: When a user performs an action, such as searching for jobs or creating a blog post, the frontend sends an API request to the Flask backend, as defined in the i1wsgi.py file.

Backend processing: The Flask application processes API requests by interacting with the SQLite database using SQLAlchemy. It does things like fetch job postings, filter results based on user input, or store new blog posts.

Search Integration: For search-related queries, the application uses its own search algorithm to efficiently index and search job postings.

Collaborative filtering: The recommendation.py file contains a collaborative filtering algorithm that generates personalized job recommendations based on user preferences and ratings. The algorithm uses NumPy and Pandas for efficient data manipulation.

Response generation: After the backend processing is complete, the Flask application generates an appropriate response, such as a list of search results or a confirmation that a blog post was successfully created.

Front-end rendering: The response is sent back to the front-end, which updates the UI accordingly, showing search results or new blog posts.

in conclusion

The job recommendation web application effectively combines various technologies and frameworks to provide a seamless and user-friendly experience. The i1wsgi.py file serves as the backbone of the application, handling API requests and managing backend services. From user interaction to back-end processing and response generation, the application presents a well-designed and efficient solution for job seekers in today's competitive marketplace.

# 5. Project Management

Requirements Analysis: The first step in project management is to gather and analyze the requirements for the job recommendation web application. This involves identifying desired features and functionality, as well as target audience and platform requirements.

Scope Definition: Based on the requirements analysis, define the project scope to identify the specific deliverables, goals and objectives of the project.

Timelines and Milestones: Create timelines to outline key milestones and deadlines for projects. This helps ensure that projects stay on track and that all team members understand their responsibilities and deadlines.

Resource Allocation: Identify and assign necessary resources (such as people, tools, and technology) to projects to ensure efficient execution and completion.

# 6. Analysis of Results

The job Recommendation Algorithms After integrating the website, the application demonstrates the effective integration of various technologies and frameworks to provide users with personalized job recommendations. Elasticsearch integration enables efficient searching of job postings, while collaborative filtering algorithms generate accurate and relevant job recommendations based on user preferences and ratings. The app's front end is built with Bootstrap and Vue.js, ensuring a modern, interactive and responsive user experience

# 7. Conclusions

The job recommendation web application showcases a successful combination of Flask, SQLite, SQLAlchemy, Bootstrap, Vue.js, and Elasticsearch to provide a user-friendly platform for job seekers. The app's advanced search capabilities and personalized recommendations not only improve the job search process, but also increase user engagement and satisfaction. Therefore, this vertical search platform becomes a practical and efficient solution for job seekers in today's competitive job market.

# 8. References

Yang, X., Zhang, Y., & Xiao, P. (2020). A Novel Graph Model-based Crawler Algorithm for Deep Web. IEEE Access, 8, 150-170.

Dong, C., Zhou, L., & Chen, L. (2019). A Deep Learning-based Web Crawler for Online Information Acquisition. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019), Macao, China (pp. 343-349).

Wang, H., Wang, H., & Yu, H. (2019). Vertical Search Engine based on Semantic Web. In Proceedings of the 6th International Conference on Information Technology (ICIT 2019), Bangkok, Thailand (pp. 203-208).

Zhang, Y., Cai, X., & Wang, J. (2021). A Machine Learning-based Vertical Search Algorithm for Web Services. IEEE Access, 9, 56374-56385.

Navarro, G. (2001). A guided tour to approximate string matching. ACM computing surveys (CSUR), 33(1), 31-88.

Salton, G., & McGill, M. J. (1986). Introduction to Modern Information Retrieval. New York: McGraw-Hill.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web (WWW 2001), Hong Kong, China (pp. 285-295). Symoneaux, R., Galmarini, M. V. (2014) *Open-Ended Questions [*online] available from <https://www.mediacollege.com/journalism/interviews/open-ended-questions.html> [24 March 2018]

Tversky, A. and Kahneman, D. (1992) ‘Advances in Prospect Theory: Cumulative Representation of Uncertainty’. *Journal of Risk and Uncertainty* [online] 5 (4), 297–323. available from <http://link.springer.com/10.1007/BF00122574> [1 March 2018]

UNESCO Bangkok. (2016) *Closing the gender gap in STEM: Drawing more girls and women into Science, Technology, Engineering and Mathematics* [online] available from <http://unesdoc.unesco.org/images/0024/002457/245717e.pdf> [19 January 2018]