

A Project Report on
Enhancement of Hybrid Filtering for
Job Recommendation System

submitted in partial fulfillment for the award of

Bachelor of Technology

in

Computer Science and Engineering

by

G. Uma Maheswari(Y20ACS456)

D. Sai Jahn timer(Y20ACS432)

A. Hema Latha(Y20ACS406)



Under the guidance of
P. Nanda Kishore, Asst. Prof.

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2023-2024

Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the project report entitled **Enhancement of Hybrid Filtering for Job Recommendation System** that is being submitted by G. Uma Maheswari (Y20ACS456), D. Sai Jahn timer (Y20ACS432), A. Hema Latha (Y20ACS406) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
P. Nanda Kishore
Asst. Prof.

Signature of the HOD
Dr. M. Rajesh Babu
Associate Professor

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

G. Uma Maheswari (Y20ACS456)

D. Sai Jahnavi (Y20ACS432)

A. Hema Latha (Y20ACS406)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr. P. Nanda Kishore**, Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Sk. Nazeer** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar**, Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

G. Uma Maheswari (Y20ACS456)

D. Sai Jahnavi (Y20ACS432)

A. Hema Latha (Y20ACS406)

Table of Contents

List of Figures	vi
Abstract	vii
1 Introduction	1
1.1 Job Recommendation System	2
1.2 Content-based Filtering.....	4
1.3 Limitations of Content-based Filtering	4
1.4 Collaborative Filtering	5
1.4.1 Memory-based Collaborative Filtering.....	5
1.4.2 Model-based Collaborative Filtering	6
1.5 Limitations of Collaborative Recommendation System	6
1.6 Hybrid Filtering	7
1.7 Benefits of Hybrid Filtering	8
1.7.1 Improved Recommendation Accuracy.....	9
1.7.2 Enhanced Personalization	9
1.7.3 Addressing Cold Start Problem	9
1.7.4 Diverse Recommendation Generation	10
1.7.5 Robustness to Sparse Data	12
2 Literature Survey	13
3 Proposed Method	15
3.1 Design	16
3.1.1 Use Case Diagram.....	18
3.1.2 Activity Diagram	19
3.1.3 Class Diagram	21

3.1.4	Sequence Diagram	22
3.1.5	Component Diagram	23
3.1.6	State Diagram.....	24
3.2	Requirements	25
3.2.1	Hardware Requirements.....	26
3.2.2	Software Requirements	26
3.3	Libraries Used.....	26
3.3.1	NLTK	26
3.3.2	PdfMiner	28
3.3.3	Textract.....	28
3.3.4	io	29
3.3.5	ipywidgets	30
3.3.6	NumPy	31
3.3.7	Pandas	32
4	Working Model	33
4.1	Functions Used.....	33
4.2	Flow of Work	34
4.3	SYSTEM ARCHITECTURE	35
4.4	DATASET DESCRIPTION	37
5	Experimental Results	42
6	Conclusion	45
7	Future Scope	46
8	Bibliography	47

List of Figures

Figure 1.1 Basic Recommendation System	3
Figure 1.2 Content and Collaborative Filtering	7
Figure 1.3 Hybrid Filtering working flowchart	8
Figure 3.1 Use Case Diagram	18
Figure 3.2 Activity Diagram	19
Figure 3.3 Class Diagram	21
Figure 3.4 Sequence Diagram	22
Figure 3.5 Component Diagram	23
Figure 3.6 State Diagram	24
Figure 4.1 dice-com-jobs datasets.....	38
Figure 4.2 survey results dataset [1]	39
Figure 4.3 survey results dataset [2]	40
Figure 5.1 To take file as an input.....	42
Figure 5.2 Output after processing resume	43
Figure 5.3 Manual entry of data.....	43
Figure 5.4 Output after processing user data	44
Figure 5.5 Output when input is user_id.....	44

Abstract

In today's dynamic job market, efficient job recommendation systems play a crucial role in connecting job seekers with relevant employment opportunities. This project proposes a novel approach to job recommendation leveraging hybrid filtering techniques, which combines collaborative filtering and content-based filtering. Unlike traditional methods that primarily rely on either user behaviour or item attributes, our hybrid model integrates both resume or user data with job requirements to enhance recommendation accuracy.

The system begins by collecting comprehensive user data, including resume information such as skills, experience, and education. Next, it employs collaborative filtering to analyse user similarities based on their skills and projects. Simultaneously, content-based filtering extracts semantic features from job descriptions and aligns them with the user's skill set and preferences. These two approaches are then combined using a weighted hybrid model, where the weights are optimized through machine learning methodologies.

The proposed system addresses the limitations of conventional recommendation systems by providing personalized job suggestions that match the user's qualifications, preferences, and career objectives. By integrating user-specific data from resumes with job descriptions, the hybrid filtering approach offers more accurate and diverse recommendations, thereby enhancing user satisfaction and increasing the likelihood of successful job matches.

1 Introduction

In today's technologically advanced society, the process of finding the perfect job amidst the vast expanse of the job market can be an arduous task. Traditional methods of job hunting often inundate candidates with an overwhelming amount of information, leaving them struggling to navigate through numerous job postings to identify opportunities that truly align with their unique skills, experiences, and career aspirations. However, with the advent of machine learning-powered job recommendation systems, there lies a promising solution to this predicament.

These sophisticated systems hold immense significance in today's fast-paced and dynamic job market. This job recommendation system is capable of analysing extensive datasets, including the profiles of job seekers and the details of job postings. This analysis enables these systems to deliver tailored recommendations personalized to each user's specific preferences and qualifications. In essence, they act as virtual assistants, guiding job seekers through the labyrinth of job opportunities and presenting them with options that best suit their individual needs.

The core strength of machine learning-based job recommendation systems lies in their ability to go beyond mere keyword matching. Instead, they delve deep into the nuances of a candidate's profile and preferences, as well as the requirements of available job positions. Moreover, the benefits of job recommendation systems extend beyond job seekers to encompass employers as well. By streamlining the recruitment process and providing employers with a pool of highly qualified candidates, these systems enhance hiring outcomes and contribute to the overall efficiency of talent acquisition.

efforts by ensuring that candidates are evaluated solely based on their merits and qualifications.

1.1 Job Recommendation System

In today's fast-paced digital landscape, the quest for suitable employment opportunities is increasingly reliant on sophisticated job recommendation systems. These systems serve as pivotal mediators between job seekers navigating a labyrinth of career options and employers seeking the perfect fit for their vacancies. Powered by cutting-edge machine learning algorithms and data-driven insights, job recommendation systems analyse vast repositories of job listings, candidate profiles, and historical interactions to orchestrate seamless matches between talent and opportunity.

At their core, job recommendation systems aim to streamline the job search process by delivering personalized and pertinent job suggestions tailored to the unique attributes of each user. By harnessing a diverse array of algorithms and methodologies, these systems meticulously curate job recommendations that align with an individual's skill set, experience, preferences, and career aspirations. Whether through collaborative filtering, content-based filtering, or hybrid approaches that amalgamate multiple techniques, job recommendation systems strive to decipher the intricate nuances of the modern job market to facilitate optimal job matches.

In an era where the abundance of job listings can overwhelm even the most discerning job seeker, the role of job recommendation systems becomes increasingly paramount. These systems not only alleviate the burden of sifting through countless listings but also empower users to discover hidden opportunities that may have otherwise eluded their radar. Moreover, by continuously learning and adapting to user feedback and evolving market dynamics, job recommendation

As we delve into the intricacies of job recommendation systems, exploring the innovative integration of hybrid filtering techniques emerges as a focal point. By synergistically combining the strengths of collaborative filtering, which leverages user interactions and similarities, with content-based filtering, which analyses textual and semantic attributes of job listings and user profiles, hybrid approaches promise to offer unparalleled recommendation accuracy and granularity. This marriage of methodologies not only enhances the system's ability to discern nuanced user preferences but also mitigates the limitations inherent in standalone approaches, thereby ushering in a new era of intelligent and adaptive job recommendation systems.

In essence, job recommendation systems stand at the forefront of the intersection between technology and human capital, poised to revolutionize the way we navigate the intricate terrain of the job market. By harnessing the power of data-driven insights and machine learning, these systems hold the potential to democratize access to employment opportunities, empower individuals to make informed career decisions, and foster symbiotic relationships between talent and organizations.

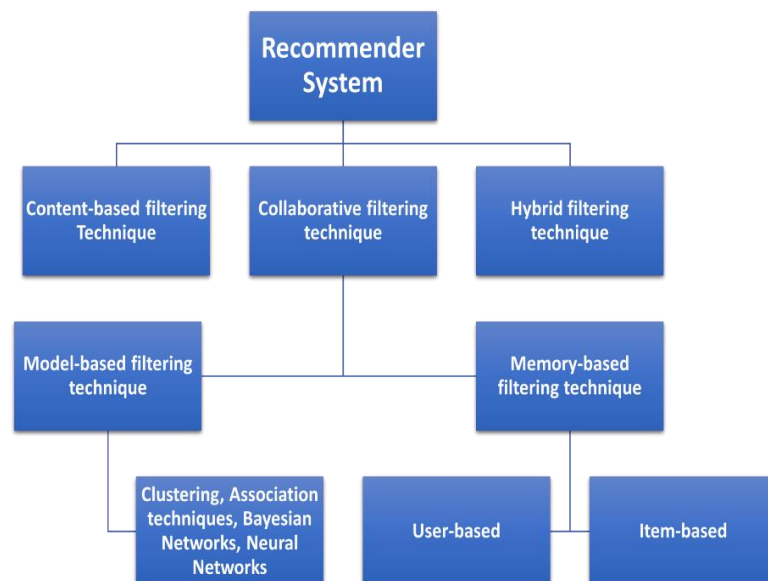


Figure 1.1 Basic Recommendation System

We have mainly three approaches for the job recommendation system designing:

1. Content-based Filtering
2. Collaborative Filtering
3. Hybrid Filtering

1.2 Content-based Filtering

Content-based filtering recommends items (in this case, jobs) based on their attributes and features, as well as the user's preferences. In job recommendation systems, content-based filtering analyses job descriptions, candidate resumes, and other textual data to match job listings with the skills, qualifications, and preferences specified by the user. This approach is beneficial for making personalized recommendations but may suffer from limited diversity if the user's preferences are narrow. The final recommendations are generated based on user's profile data. This system provides the suggestion based on user's similarity with the items. Mainly, the concept of Term Frequency Inverse Document Frequency (TFIDF) is used in information retrieval and content-based recommendation system. TFIDF basically computes the frequency of words in respective documents.

1.3 Limitations of Content-based Filtering

Sparsity problem is the situation which concerns about insufficient data present in the dataset. Generally, the content-based approach faces sparsity problem. Which means, these methods limit the recommendation only to user specifics. As the method only involves using the user related data, the dataset is insufficient as it does not involve rating given by the other users. The recommendation engine developed will not

recommend anything besides user's interest. Hence this approach only helps to recommend the result based on user's interest and not based on other users' preferences.

1.4 Collaborative Filtering

Collaborative filtering techniques analyse user behaviour and interactions to make recommendations. In the context of job recommendation systems, collaborative filtering compares the preferences and past interactions of users to identify similarities and recommend jobs that similar users have shown interest in. This approach is effective in capturing user preferences without requiring explicit information about job characteristics. historical data of users is used to make the recommendations. Based on the explicit ratings given by the users, the user-to-user similarity is calculated and then the corresponding items are recommended to the users.

1.4.1 Memory-based Collaborative Filtering

Memory-based collaborative filtering is a popular technique used in recommendation systems to suggest items of interest to users based on the preferences and behaviors of similar users. The fundamental idea behind this approach is to leverage the historical data of users, particularly their ratings or interactions with items, to identify patterns of similarity among users and make personalized recommendations.

In the context of job recommendation systems, memory-based collaborative filtering works by first collecting data on the historical interactions between users and job postings. This data typically includes information such as which jobs users have applied to, viewed, or rated positively. Based on this data, the system computes a similarity measure between users, indicating how closely their preferences align.

1.4.2 Model-based Collaborative Filtering

In memory-based collaborative filtering (CF), Singular Value Decomposition (SVD) is indeed a widely used machine learning algorithm for predicting user ratings on unrated items. SVD is a matrix factorization technique that decomposes a large matrix into three smaller matrices, capturing the underlying latent factors that drive user-item interactions. In the context of recommendation systems, these latent factors represent hidden patterns or features in the data that influence users' preferences and behaviours.

The advantage of using SVD in memory-based CF lies in its ability to capture complex patterns and relationships in the data, even when dealing with sparse matrices (i.e., matrices with a large number of missing values). By decomposing the original rating matrix into lower-dimensional matrices of user and item factors, SVD allows for more efficient computation and storage of recommendations while preserving the underlying structure of the data.

1.5 Limitations of Collaborative Recommendation System

The recommendation system experiences cold start problem as it does not have any relevant past data of the users. The cold start problem is experienced in case of new user. When a new user registers himself to the system, the proposed model does not know about his interest and the user did not make any rating to the existing companies. Due to this scenario the system won't be able to recommend anything to the user. This approach uses more amount of data consisting of different perspective of different users.

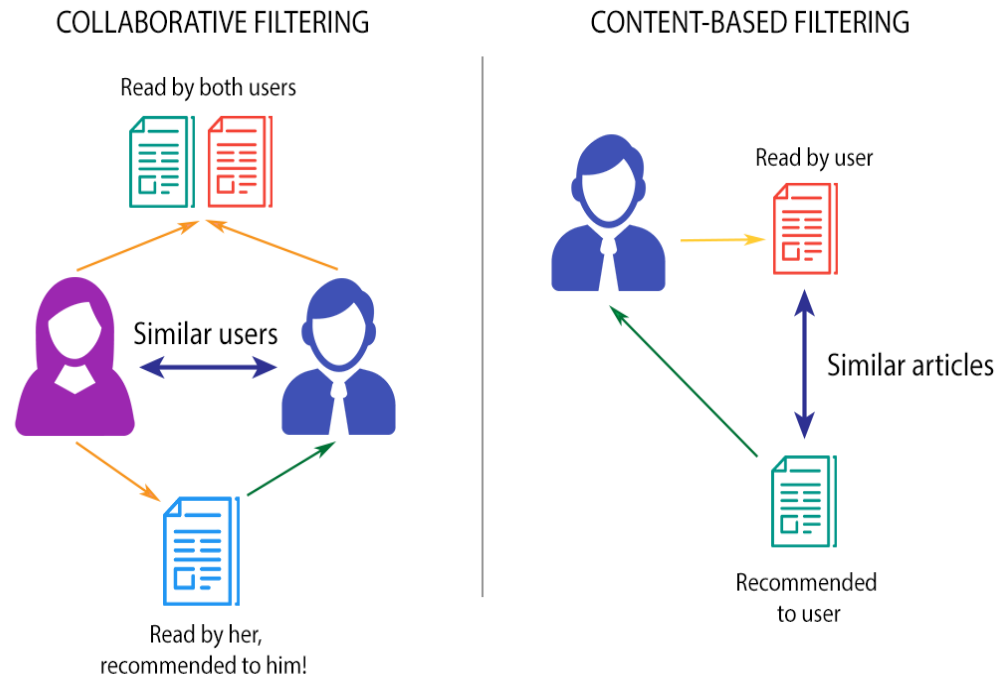


Figure 1.2 Content and Collaborative Filtering

1.6 Hybrid Filtering

Hybrid filtering techniques, which integrate both collaborative and content-based filtering approaches, have gained widespread popularity in the field of recommendation systems due to their ability to overcome the limitations of individual methods while leveraging their respective strengths. One common method used in hybrid filtering is the weighted average technique, where recommendations from both collaborative and content-based filtering are combined using a weighted score to generate the final recommendations.

The weighted average technique involves assigning weights to the recommendations generated by each method based on their perceived reliability or relevance. These weights are typically determined empirically or through machine learning algorithms trained on historical data. The final recommendation score for each item is then calculated as the weighted average of the scores assigned by each method. One of the

key advantages of hybrid filtering is its ability to improve recommendation accuracy. By combining multiple recommendation sources, hybrid systems can capture a broader range of user preferences and provide more tailored recommendations. This leads to a more personalized user experience and increased satisfaction with the recommendation service.

Additionally, hybrid filtering helps mitigate the cold start problem, which arises when there is insufficient data to make accurate recommendations for new users or items. Content-based filtering can be particularly effective in addressing this issue by leveraging item metadata or user profiles to make recommendations based on item characteristics or user preferences.

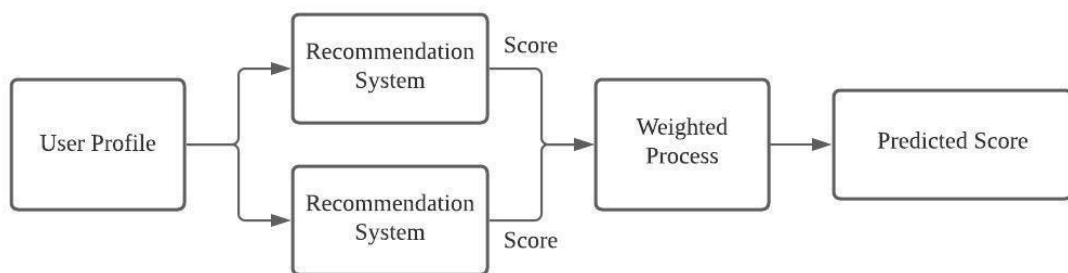


Figure 1.3 Hybrid Filtering working flowchart

1.7 Benefits of Hybrid Filtering

Hybrid filtering combines the strengths of collaborative and content-based filtering while mitigating their respective limitations, resulting in more accurate, personalized, and diverse recommendations, and ultimately enhancing the user experience and satisfaction with the recommendation service.

1.7.1 Improved Recommendation Accuracy

Hybrid filtering represents a sophisticated approach to recommendation systems, seamlessly combining the merits of content-based and collaborative filtering methodologies. Content-based filtering operates by examining the attributes of items and user profiles to suggest items that align with a user's preferences. It relies on the idea that users who have liked similar items in the past will also enjoy similar items in the future. However, content-based filtering may struggle when it comes to recommending new or niche items that lack sufficient user interactions or detailed content descriptions. Moreover, it may suffer from the "cold start" problem, where it's challenging to provide recommendations for new users with limited interaction history.

On the other hand, collaborative filtering analyses user interactions and preferences to make recommendations. It identifies patterns of behaviour across a user base, suggesting items that users with similar tastes have enjoyed in the past. Collaborative filtering excels in handling the cold start problem by relying solely on user interaction data, but it may face challenges with recommending items with sparse interactions or lacking in explicit user feedback. Additionally, it may suffer from the "sparsity" issue, where user-item interactions are scarce, leading to less accurate recommendations. By integrating content-based and collaborative filtering techniques, hybrid filtering systems can mitigate these limitations.

1.7.2 Enhanced Personalization

Content-based filtering operates by analysing the characteristics and attributes of items, such as text descriptions, tags, or metadata, to recommend items that closely match a user's preferences. This approach is particularly effective in scenarios where users have provided explicit feedback or have interacted with items in the past. However, content-

based filtering alone may overlook serendipitous discoveries or recommend only items similar to those a user has already engaged with, potentially limiting the diversity of recommendations. Collaborative filtering, on the other hand, identifies underlying patterns in user behaviour by analysing interactions and preferences across a user base.

Hybrid filtering integrates both content-based and collaborative filtering techniques to provide more personalized and diverse recommendations. By combining these approaches, hybrid systems can leverage the advantages of content analysis and user interaction data simultaneously. This integration allows hybrid filtering to overcome the limitations of individual methods; it can recommend items based on user preferences while also considering item attributes, leading to more accurate and comprehensive recommendations. Additionally, hybrid filtering systems can adapt to various scenarios and user preferences, striking a balance between serendipity and relevance in recommendation generation. Overall, hybrid filtering enhances the recommendation process by offering users a more tailored and engaging experience that accounts for both their preferences and the intrinsic characteristics of recommended items.

1.7.3 Addressing Cold Start Problem

The cold start problem poses a significant challenge for collaborative filtering, particularly when dealing with new users or items with sparse interaction data. In such cases, collaborative filtering lacks sufficient information to identify patterns or similarities among users or items, hindering its ability to generate accurate recommendations. Content-based filtering, on the other hand, relies on the attributes of items to make recommendations, making it well-suited for providing suggestions for

new items. However, it may struggle if there's limited information about user preferences, leading to potentially less personalized recommendations.

Hybrid filtering addresses these limitations by leveraging the strengths of both content-based and collaborative filtering approaches. When encountering new items with little to no interaction history, hybrid filtering can rely on content-based features to generate initial recommendations based on the intrinsic characteristics of the items. This allows the system to provide relevant suggestions even in the absence of user interaction data. As users interact with these recommended items, hybrid filtering gradually incorporates their feedback and behaviour into the recommendation process. Collaborative filtering comes into play here, analyzing user interactions and preferences to refine and personalize recommendations over time.

1.7.4 Diverse Recommendation Generation

Collaborative filtering, while effective in recommending items based on user interactions, often suffers from the "popularity bias" or "herd effect" phenomenon. This means that popular items tend to receive more recommendations, leading to a homogeneity in recommendations where users are repeatedly exposed to the same popular items. Additionally, collaborative filtering tends to recommend items that are similar to those a user has already interacted with, which can further reinforce this homogeneity and limit the diversity of recommendations.

In contrast, content-based filtering considers item attributes such as genre, category, or features, which allows for the recommendation of items that may not be popular but are relevant to a user's interests based on their characteristics. This introduces diversity into the recommendation process, as users are exposed to a wider range of items beyond those that are already popular or similar to their past interactions.

1.7.5 Robustness to Sparse Data

Sparse data poses a significant challenge for collaborative filtering, especially in scenarios where there are limited interactions or a lack of users with similar preferences. In such cases, collaborative filtering may struggle to identify meaningful patterns or similarities among users or items, resulting in less accurate recommendations. This issue can be particularly pronounced in niche or specialized domains where user interactions are inherently scarce. Content-based filtering, however, offers a complementary approach by leveraging item attributes to make recommendations. By analyzing the intrinsic characteristics of items, such as text descriptions, tags, or metadata, content-based filtering can provide valuable information that supplements the sparse interaction data. This additional information helps enrich the recommendation process and can compensate for the lack of user feedback, making content-based filtering a valuable asset in mitigating the challenges of data sparsity in collaborative filtering.

Hybrid filtering capitalizes on the strengths of both collaborative and content-based filtering techniques to offer more robust recommendations in scenarios characterized by data sparsity. By integrating these approaches, hybrid filtering combines the advantages of collaborative filtering's ability to capture user preferences with content-based filtering's capability to provide information about item attributes. When collaborative filtering encounters sparse data, hybrid filtering can seamlessly incorporate content-based features to enhance recommendation accuracy and relevance. This integration ensures that the recommendation system remains effective even in challenging environments where one method alone may struggle due to data limitations.

2 Literature Survey

[1] This comprehensive survey provides an overview of hybrid recommender systems, including collaborative, content-based, and hybrid approaches. It discusses the motivations, challenges, and strategies for combining different recommendation techniques. The survey highlights the benefits of hybrid systems in improving recommendation accuracy and diversity, laying the groundwork for the application of hybrid filtering in job recommendation systems.

[2] This paper presents a personalized job recommendation system based on a hybrid model combining collaborative filtering and content-based filtering. The authors propose a novel similarity measure that integrates user preferences and job attributes. Experimental results demonstrate the effectiveness of the hybrid approach in enhancing recommendation accuracy and relevance for job seekers.

[3] This study introduces a job recommendation system that integrates collaborative filtering with semantic similarity analysis. The system employs a hybrid approach to capture user preferences and job characteristics, leveraging both user behaviour data and semantic representations of job descriptions.

[4] This research proposes an enhanced collaborative filtering recommender system for job recommendation by incorporating user profiles and item profiles. The hybrid approach combines collaborative filtering with content-based filtering to capture both user preferences and job characteristics. The study demonstrates the effectiveness of the hybrid system in generating personalized and relevant job recommendations for users.

[5] This paper presents a hybrid job recommender system that combines improved item-based collaborative filtering with the weighted Slope One algorithm. The hybrid approach integrates user-item interactions and job attributes to generate personalized recommendations for job seekers. Experimental results show that the hybrid system outperforms traditional collaborative filtering methods in terms of recommendation accuracy and coverage.

[6] This study proposes a job recommendation system based on a hybrid filtering approach utilizing text mining techniques. The hybrid system combines collaborative filtering with content-based filtering, leveraging textual data from job descriptions and user profiles. Experimental evaluations demonstrate the effectiveness of the hybrid approach in improving recommendation accuracy and relevance for job seekers.

[7] The system leverages user-item interactions and textual data from job descriptions to generate personalized job recommendations. Experimental results validate the efficacy of the hybrid approach in enhancing recommendation accuracy and diversity.

3 Proposed Method

Developing a job recommendation system using a hybrid filtering model is indeed a promising solution to the challenges faced in job matching. This approach combines the strengths of collaborative filtering, which analyzes past user behavior and preferences, with content-based filtering, which examines the attributes of job postings and user profiles, to generate more accurate and relevant recommendations.

Collaborative filtering excels at capturing user preferences and identifying patterns in past interactions, making it well-suited for recommending jobs based on similarities with other users' behaviors. However, it may suffer from the cold start problem for new users or items with limited historical data. On the other hand, content-based filtering leverages the characteristics of job postings and user profiles to make personalized recommendations, which can address the cold start problem and introduce diversity into the recommendations. However, it may struggle to capture nuanced user preferences and may overlook relevant job opportunities that do not align with explicit user preferences.

By integrating both collaborative and content-based filtering methods, the hybrid model can overcome the limitations of each approach. Collaborative filtering provides valuable insights into user preferences and identifies job opportunities that are similar to those favored by other users. Meanwhile, content-based filtering ensures that recommendations are tailored to each user's unique profile and preferences, thus enhancing personalization and addressing the cold start problem.

3.1 Design

The recommendation system implementation by acquiring a dataset of companies, is a crucial step in the content-based filtering approach. This dataset is obtained through web scraping from Kaggle Website, a platform containing comprehensive information about various machine Learning projects. Once the data is gathered, it proceeds to preprocess for analysis. The code we design to quantify the importance of each word in the dataset by considering both its frequency within a specific company profile and its rarity across all profiles.

Cosine similarity measures the cosine of the angle between two vectors, in this case, this similarity computation results in a similarity matrix, where each entry represents the similarity score between two companies. Utilizing this similarity matrix, the code generates top-N recommendations using a content-based approach. By identifying companies with attributes most similar to a given target company, the system recommends similar companies to users. In transitioning to Collaborative Filtering, the system adopts a memory-based approach, specifically employing the item-based approach where items represent skills and projects. Here's how this approach works:

Similarity Calculation: The system begins by calculating the similarity between skills (items) based on various factors such as co-occurrence in job postings, frequency of appearance together in user profiles, or semantic similarity using techniques like cosine similarity or Jaccard similarity. This similarity measure helps identify which skills are closely related to each other and are likely to be relevant in similar job contexts.

Finding Similar Users: Once the similarity between skills is determined, the system identifies users who have similar skill profiles. This is achieved by comparing the skills

listed in user profiles and computing the similarity between users based on the overlap or similarity of their skill sets. Users with similar skill profiles are considered to have similar preferences and interests in job opportunities.

Content-Based Filtering on Similar Users: After identifying similar users, the system applies content-based filtering to recommend jobs to the target user. Instead of directly recommending jobs based on collaborative filtering (which would involve recommending jobs based on the preferences of similar users), the system leverages the skills and preferences of similar users to generate personalized recommendations for the target user.

Recommendation Generation: The system selects jobs that are preferred or have been interacted with positively by similar users but have not yet been seen by the target user. These jobs are considered relevant based on the assumption that if similar users have shown interest in these jobs, the target user is likely to find them appealing as well. These recommendations are then presented to the target user, tailored to their specific skill set and preferences.

After obtaining recommendations from both content-based and collaborative filtering approaches, the program combines the results using a weighted average technique. This fusion aims to leverage the strengths of each method and provide more comprehensive and accurate top-N recommendations to users. Ultimately, we get a Hybrid Recommendation System that integrates content-based and collaborative filtering methodologies, offering users a more robust and personalized recommendation experience.

3.1.1 Use Case Diagram

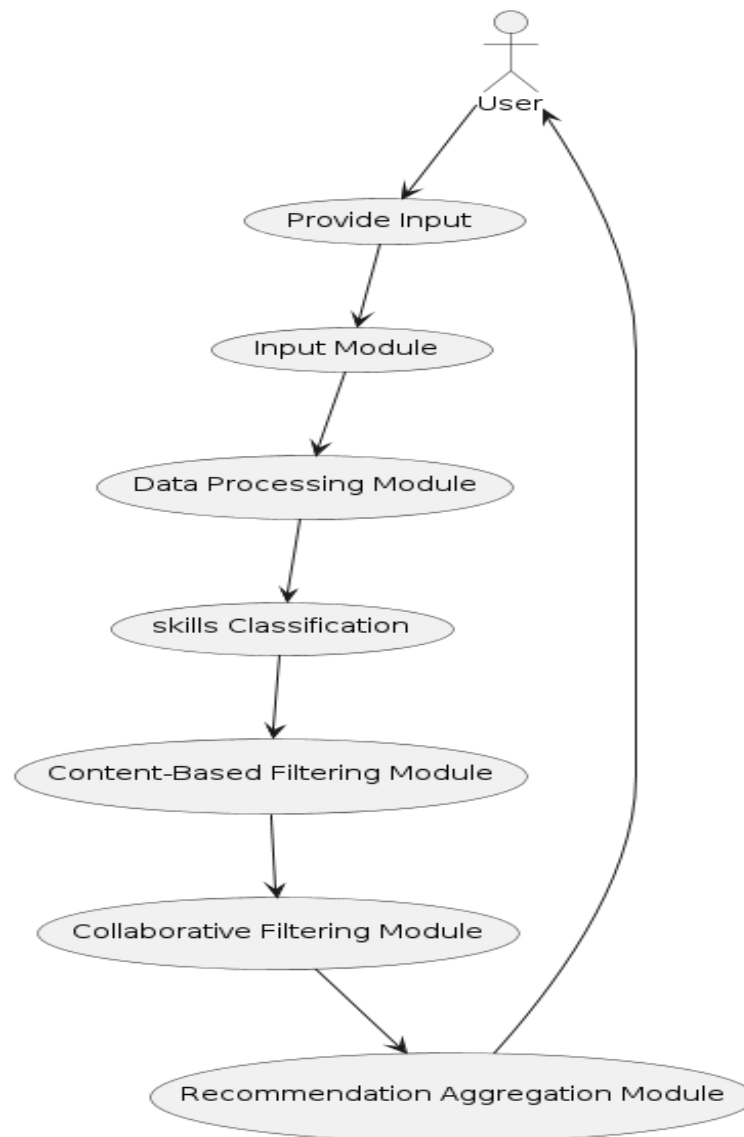


Figure 3.1 Use Case Diagram

Figure 3.1 depicts the job recommendation system facilitates personalized job suggestions by integrating content-based and collaborative filtering techniques. Users input their details, either through predefined IDs, manual entry, or resume uploads. The Input Module processes this data, while the Data Processing Module refines it, extracting skills and categorizing domains. Next, the system classifies the user's domain of interest. Content-Based Filtering then suggests jobs based on the similarity between user skills and job requirements. Simultaneously, Collaborative Filtering identifies

similar users and recommends jobs based on their interactions. Finally, the Recommendation Aggregation Module combines recommendations from both approaches, providing a tailored list of job opportunities. This system streamlines the job search process, offering users relevant job matches based on their skills, preferences, and industry interests.

3.1.2 Activity Diagram

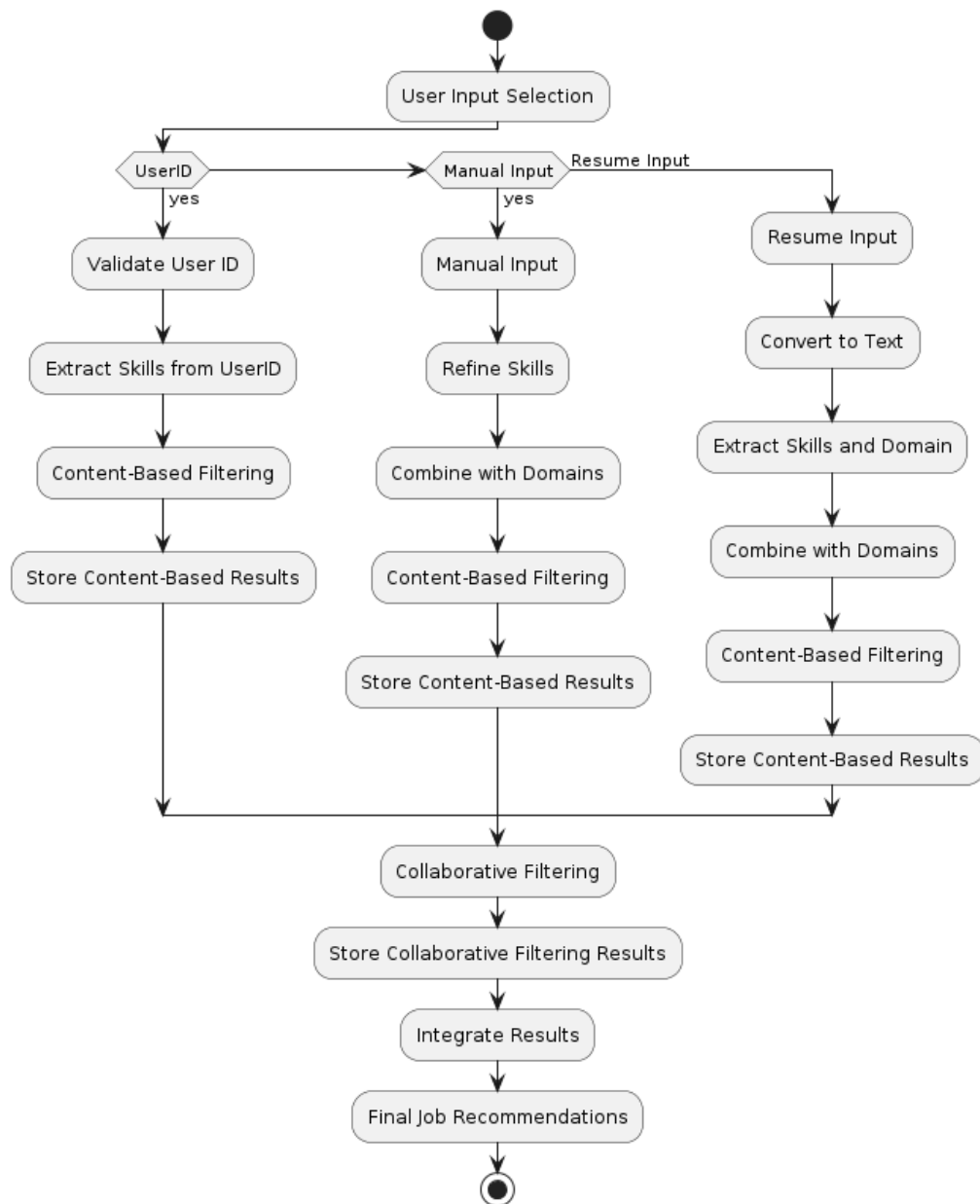


Figure 3.2 Activity Diagram

Figure 3.2 shows the activity diagram illustrates the workflow of the job recommendation system, starting with the user selecting an input method among UserID, Manual Input, or Resume Input. Depending on the chosen method, the system performs validation, skill extraction, and refinement processes. For UserID input, the user's skills are extracted from the database, while for Manual Input, the user manually enters their details. Resume Input involves the conversion of resume files to text and skill extraction. Subsequently, the system applies content-based filtering to recommend jobs based on the user's skills and domain interests, storing the recommendations for later use. Additionally, collaborative filtering identifies similar users and generates recommendations based on their preferences, with results also stored. Finally, the integrated recommendations are presented to the user, reflecting personalized job suggestions. This diagram communicates the sequence of actions, decision points, and concurrency in the recommendation process, facilitating understanding, collaboration, and system improvement efforts throughout the development lifecycle.

Furthermore, the activity diagram supports iterative development methodologies by enabling continuous refinement and enhancement of the recommendation process over time. Its comprehensive depiction of the system's functionality and logic aids in the training of users and system administrators, promoting successful adoption and utilization of the job recommendation system.

3.1.3 Class Diagram

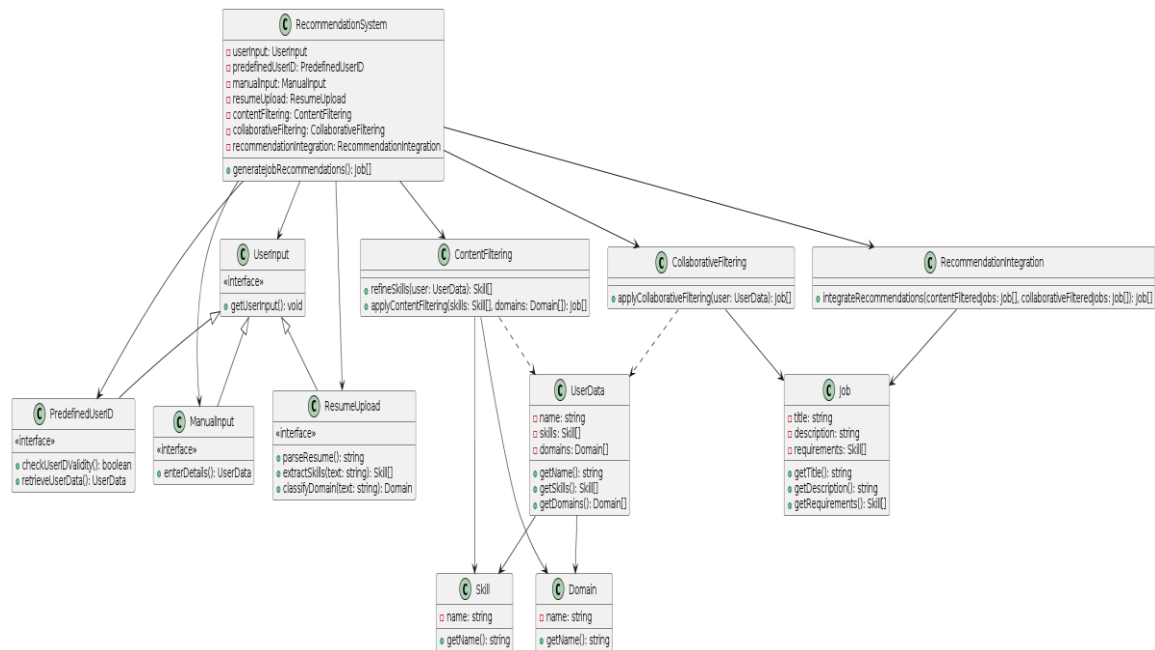


Figure 3.3 Class Diagram

Figure 3.3 illustrates the class diagram for the job recommendation system outlines key components and their relationships. It includes classes like User, Job, Skill, Domain, and Recommendation. Users store attributes such as user ID and preferences, while jobs contain details like title and description. Skills and domains facilitate matching users with suitable jobs. Associations depict relationships between classes, such as users having skills and jobs requiring specific skills. The diagram guides system design, fostering communication and understanding among team members. It supports modular design principles, aids in documentation, and facilitates code reusability. As the system evolves, the class diagram can be updated to reflect changes in requirements and technology, ensuring alignment with objectives.

3.1.4 Sequence Diagram

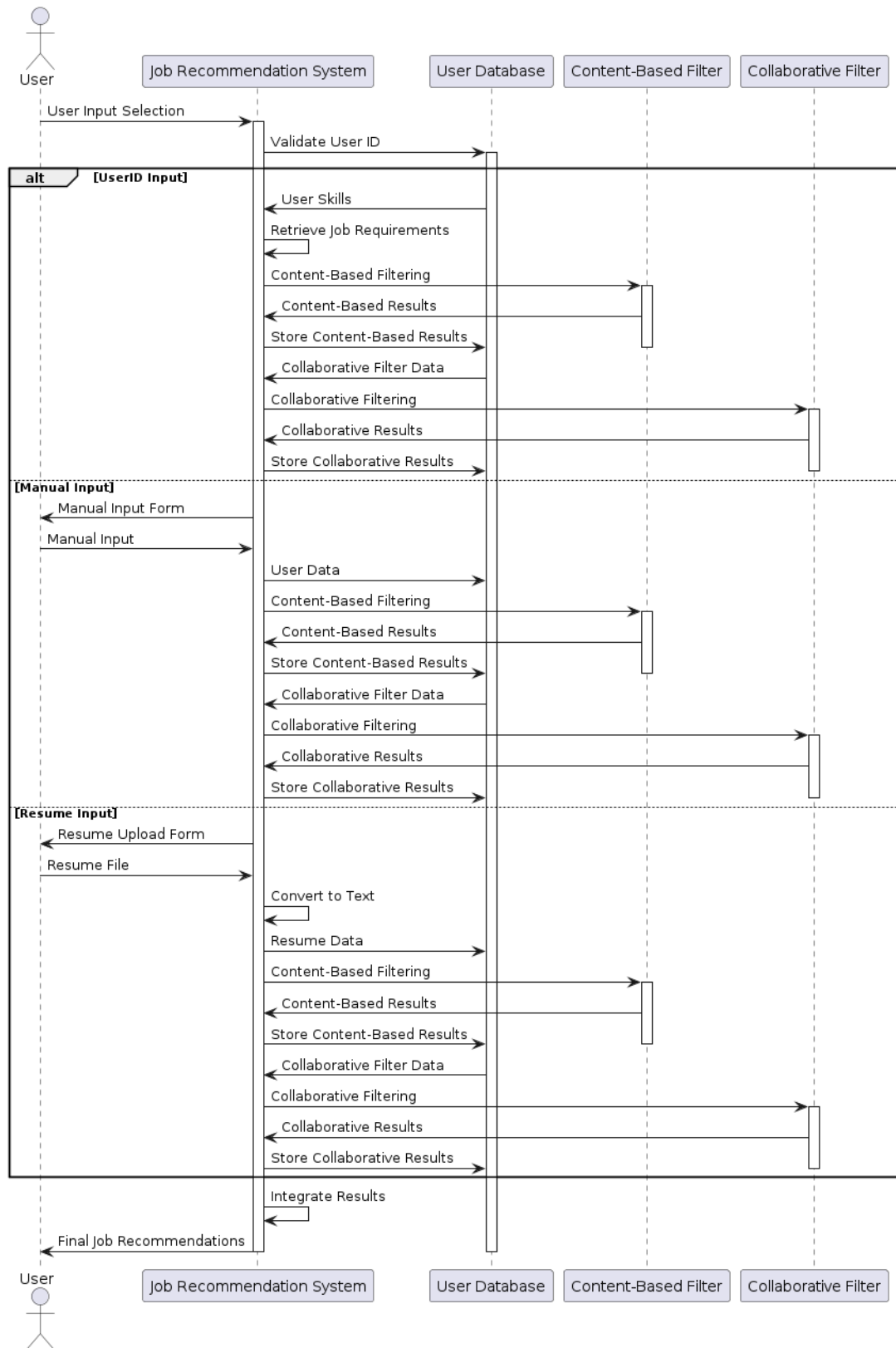


Figure 3.4 Sequence Diagram

3.1.5 Component Diagram

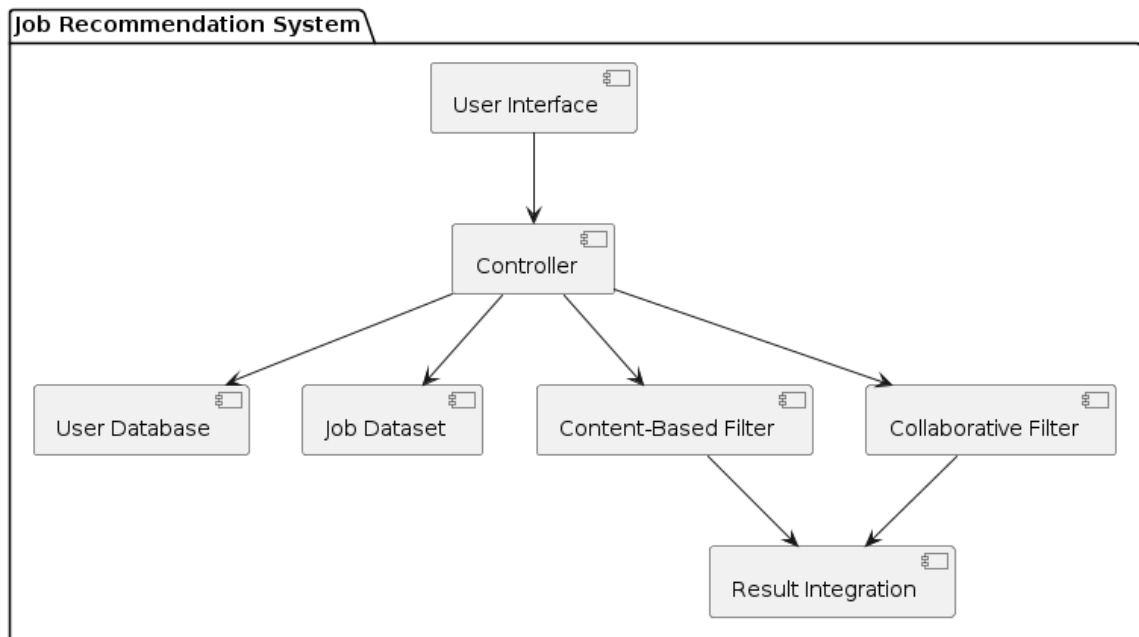


Figure 3.5 Component Diagram

The provided PlantUML code outlines the architecture of a Job Recommendation System, comprised of several interconnected modules. At its core is the Controller, which serves as the system's control center, orchestrating communication and operations between different components. Users interact with the system through the User Interface, initiating requests for job recommendations. The Controller interacts with the User Database to manage user profiles and historical data, while also accessing the Job Dataset containing information about available jobs. Two main recommendation engines, the Content-Based Filter and the Collaborative Filter, operate under the Controller's supervision. The Content-Based Filter analyses job and user data to suggest relevant matches, while the Collaborative Filter leverages collective user behaviour for recommendations. Both filters feed their results into the Result Integration component, which merges them into a cohesive set of suggestions.

3.1.6 State Diagram

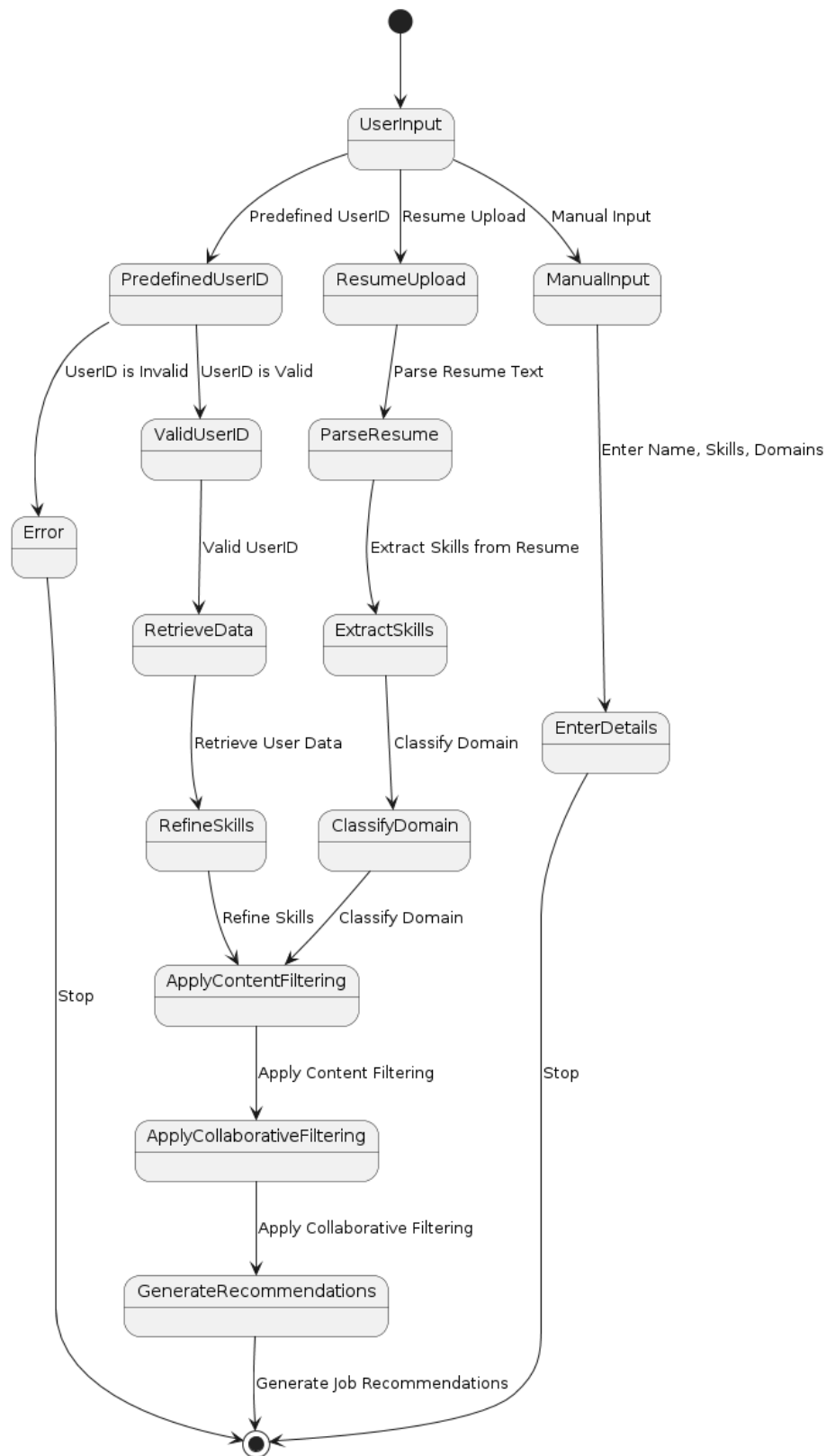


Figure 3.6 State Diagram

Figure 3.6 shows a state diagram for the Job Recommendation System project would delineate the various states and transitions that the system and its components undergo in response to different events. The diagram would depict states such as Idle, indicating the system awaiting user input, and Processing Request, where it actively handles user queries. Other states like Accessing Data would signify the retrieval of user and job data from respective databases. The Content-Based Filter and Collaborative Filter would have states for analysing job and user information. Transitions between states would occur based on events like user requests triggering data retrieval, or completion of filtering processes leading to recommendation integration. Ultimately, the system would transition to the Displaying Recommendations state, presenting personalized job suggestions to users via the User Interface. This visual representation would offer a clear roadmap of the system's behaviour, aiding in understanding its operational flow and facilitating optimization and refinement efforts for an efficient job recommendation process.

3.2 Requirements

The requirements of our application are categorized into software and hardware components to ensure smooth functionality and compatibility. Software requirements encompass the necessary runtime environments, frameworks, and dependencies such as Java, Python, or specific libraries like NLTK. Hardware requirements include standard computer components such as processors, memory (RAM), and storage capacity. These components should meet recommended specifications to support the application's computational tasks efficiently.

3.2.1 Hardware Requirements

RAM: 4GB (min)

Hard Disk: 128GB (min)

Keyboard: Standard Windows / MAC Keyboard

Processor: Intel core i5 and above (recommended)

3.2.2 Software Requirements

Technologies: Python

IDE: Jupyter Notebook / Google Collab / Visual Studio Code

Operating System: WindowsXP / MacOS

Version: Python 3.11

3.3 Libraries Used

The libraries included textract for extracting text from various formats, pdfminer for parsing PDFs, python-docx for Word documents, and xlrd for Excel files. Additionally, pytesseract was utilized for optical character recognition when handling image-based documents. By leveraging these libraries, the study achieved efficient extraction of text from diverse document sources, facilitating further analysis and research objectives.

3.3.1 NLTK

Natural Language Toolkit (NLTK) is a powerful platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, such as WordNet, along with a suite of text processing libraries

for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is widely used in educational and research settings for tasks ranging from simple text processing to more complex natural language understanding and generation.

One of NLTK's key features is its extensive collection of corpora and lexical resources, which include text data from various languages, genres, and domains. These resources enable researchers and developers to access and analyse large datasets for tasks like sentiment analysis, text classification, and machine translation. Additionally, NLTK provides convenient methods for downloading and accessing these resources, making it easy to get started with natural language processing tasks.

NLTK's text processing libraries offer a wide range of functionality for working with human language data. For example, NLTK includes modules for tokenization, which involves splitting text into individual words or sentences, and stemming, which reduces words to their base or root form. It also offers tools for part-of-speech tagging, named entity recognition, and syntactic parsing, allowing users to analyze the grammatical structure and meaning of text.

Moreover, NLTK supports machine learning and statistical modeling techniques for natural language processing tasks. It includes modules for training and evaluating classifiers on text data, as well as for performing feature extraction and dimensionality reduction. NLTK's integration with popular machine learning libraries such as scikit-learn enables developers to build sophisticated text processing pipelines for tasks like sentiment analysis, document classification, and information retrieval. Overall, NLTK's comprehensive set of features and easy-to-use interface make it a valuable tool for anyone working with human language data in Python.

3.3.2 PdfMiner

PDFMiner is a Python library designed to facilitate the extraction of text and metadata from PDF documents. It offers a range of functionalities tailored to parse PDF files efficiently, enabling users to access both textual content and document metadata. One of its key features is its ability to provide both high-level and low-level access to PDF data, granting users flexibility in how they interact with PDF documents.

At a high level, PDFMiner simplifies text extraction by offering methods to retrieve text content while preserving its layout. This is particularly useful when maintaining the original formatting of the text is crucial, such as when processing documents for readability or analysis. Additionally, users can access document metadata such as author, title, and creation date, providing valuable contextual information about the PDF files being processed. For users requiring more granular control over PDF data, PDFMiner offers low-level access to individual elements within PDF documents. This includes the ability to access characters, lines, and rectangles, allowing for precise manipulation and analysis of PDF content.

3.3.3 Texttract

The texttract module in Python is a versatile library designed to simplify the extraction of text from various document formats, including PDFs, Microsoft Office files (e.g., Word, Excel, PowerPoint), and many others. It provides a unified interface for extracting text content from these documents, abstracting away the complexities of interacting with different file formats.

One of the primary advantages of the `texttract` module is its simplicity and ease of use. Developers can extract text from documents with just a few lines of code,

making it accessible to users of all skill levels. Additionally, `textract` supports a wide range of document formats, ensuring compatibility with diverse types of files commonly encountered in real-world applications.

Under the hood, `textract` employs a combination of existing libraries and tools to handle the extraction process efficiently. For instance, it may utilize `pdfminer` for PDF parsing, `python-docx` for Microsoft Word documents, and `xlrd` for Excel spreadsheets, among others. By leveraging these existing libraries, `textract` is able to provide robust text extraction capabilities without reinventing the wheel. Another notable feature of the `textract` module is its ability to handle both text-based and image-based documents. It employs optical character recognition (OCR) techniques to extract text from images, making it suitable for processing scanned documents or images containing textual content.

3.3.4 `io`

The `io` module in Python provides a set of tools and utilities for working with input and output operations, facilitating the reading and writing of data from various sources. This module serves as a versatile interface for handling streams of data, whether they originate from files, network connections, or other sources. One of the core components of the `io` module is the `BytesIO` and `StringIO` classes, which allow for the manipulation of in-memory byte streams and string buffers, respectively. These classes enable the creation of file-like objects that can be read from or written to as if they were traditional files, providing a convenient way to work with data without the need for physical files.

Additionally, the `io` module offers support for handling different types of file objects, including raw binary files, text files, and buffered streams. It provides functions and classes for opening, closing, and manipulating file objects, as well as for

performing various input and output operations, such as reading, writing, and seeking within files. This versatility makes the `io` module well-suited for a wide range of file processing tasks, from simple file reading and writing to more complex data manipulation and processing tasks.

Furthermore, the `io` module plays a crucial role in Python's I/O ecosystem, serving as the foundation for higher-level I/O libraries and tools. For instance, it forms the backbone of Python's built-in file handling capabilities, providing the underlying infrastructure for file operations performed using functions like `open ()` and `close ()`. Moreover, many third-party libraries and frameworks in Python rely on the `io` module for handling I/O operations, leveraging its functionality to create custom file-like objects, implement specialized input/output mechanisms, and interface with external data sources.

3.3.5 ipywidgets

ipywidgets is a Python library that enables the creation of interactive widgets within Jupyter notebooks and JupyterLab environments. These widgets enhance the user experience by allowing for dynamic and responsive interactions with data and visualizations, making them particularly useful for data exploration, analysis, and visualization tasks. At its core, ipywidgets provides a rich set of pre-built widgets ranging from sliders, text inputs, and buttons to more complex components like dropdown menus, date pickers, and file uploaders.

These widgets can be easily integrated into Jupyter notebooks using Python code, allowing users to interactively manipulate parameters, filter data, or trigger computations with real-time feedback. One of the key features of ipywidgets is its

seamless integration with other Python libraries commonly used in data science and visualization workflows, such as Matplotlib, Plotly, and Pandas.

3.3.6 NumPy

NumPy, short for Numerical Python, is a fundamental library in Python for scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy's primary data structure is the `ndarray`, an n-dimensional array object that enables fast and efficient operations on large datasets. This array object is the cornerstone of NumPy, offering capabilities for element-wise operations, array slicing and indexing, reshaping, and broadcasting.

One of NumPy's key advantages is its performance optimization through vectorized operations. Instead of iterating over elements in arrays, NumPy allows users to perform operations on entire arrays at once, leveraging highly optimized C and Fortran code under the hood. This vectorized approach leads to significant speed improvements compared to traditional Python loops, making NumPy ideal for handling large datasets and computationally intensive tasks commonly encountered in scientific computing, machine learning, and data analysis.

In addition to its array operations, NumPy provides a wide range of mathematical functions for numerical computation. These include functions for arithmetic operations, trigonometry, logarithms, exponential functions, statistics, linear algebra, and more. These functions are implemented to efficiently operate on NumPy arrays, ensuring high performance and accuracy. Moreover, NumPy seamlessly integrates with other libraries and tools in the scientific Python ecosystem, such as SciPy, Matplotlib, and pandas, enabling a cohesive and powerful environment for

scientific computing, data analysis, and visualization in Python. Overall, NumPy serves as a cornerstone of the scientific Python ecosystem, providing essential tools and functionality for numerical computation, array manipulation, and mathematical operations, and playing a pivotal role in enabling a wide range of scientific and data-driven applications in Python.

3.3.7 Pandas

Pandas is a powerful and versatile Python library designed for data manipulation and analysis. It provides high-level data structures, namely Series and DataFrame, that enable users to work with structured and tabular data efficiently. The DataFrame, in particular, resembles a spreadsheet or SQL table, with rows and columns of data, allowing for easy loading, manipulation, and transformation of datasets. This structure makes Pandas well-suited for tasks such as data cleaning, exploration, and preparation prior to analysis or modeling.

A key strength of Pandas lies in its extensive functionality for data manipulation and manipulation. It offers a rich set of methods and functions for filtering, sorting, grouping, and aggregating data, enabling users to perform complex data transformations with ease. Whether it's selecting specific rows and columns, merging datasets, or computing summary statistics, Pandas provides intuitive and powerful tools for manipulating data in various ways. Additionally, Pandas integrates seamlessly with other Python libraries and tools commonly used in data science and analysis workflows, such as NumPy, Matplotlib, and Scikit-learn, facilitating a cohesive and efficient data analysis pipeline.

4 Working Model

The proposed system implements a basic version of a job recommendation system that works on hybrid filtering model that combines the advantages of two systems i.e., content-based filtering, collaborative filtering. The system can take the input explicitly from the user or else, it can extract the requirements from the pdf or word file that is given as an input to the code. The main features like the job-seeking candidate's skills, expertise, internships they've worked on and experience they have been considered as well as the domain they were interested in. The code contains several functions that work together to identify the required information from the candidate, either taken manually or by giving their resume as input.

4.1 Functions Used

1. cosineSimilarity (arr1, arr2): Cosine similarity is a crucial metric for gauging similarity between vectors, widely applied in fields like information retrieval and natural language processing. By computing the cosine of the angle between two vectors, it encapsulates their directional alignment, irrespective of magnitude. Ranging from -1 to 1, a cosine similarity of 1 denotes perfect alignment, 0 indicates orthogonality, and -1 signifies complete misalignment. Its disregard for vector magnitudes makes it robust and suitable for tasks like document similarity analysis. In text processing, cosine similarity aids in tasks such as document clustering, classification, and recommendation systems, where vectors represent documents based on word occurrences. Overall, cosine similarity is a versatile and indispensable tool for quantifying similarity in various domains, facilitating effective analysis and computation with vector data.

2. jaccardSimilarity (arr1, arr2): Jaccard similarity is a measure used to compare the similarity between two sets. It is calculated by dividing the size of the intersection of the sets by the size of the union of the sets. In other words, it quantifies the overlap between two sets by considering the proportion of elements they have in common relative to the total number of distinct elements they contain. Jaccard similarity ranges from 0 to 1, where 0 indicates no overlap (complete dissimilarity) and 1 indicates complete overlap (perfect similarity). It is commonly used in various fields, including information retrieval, data mining, and recommendation systems, to compare the similarity between items or documents.

4.2 Flow of Work

Data collection: Gathering job postings, user profiles, and historical interaction data from various sources, such as job boards and recruitment platforms.

Preprocessing: Cleaning and preprocessing the data to ensure consistency and relevance, including text processing and feature extraction.

Refining skills: Refining skills in languages like Python and platforms like Jupyter Notebook is essential for utilizing libraries such as PDFMiner, textract, and NLTK effectively. Familiarity with PDFMiner and textract enables parsing and extracting text from PDF documents, while proficiency in NLTK facilitates natural language processing tasks like tokenization and stemming. Additionally, understanding data structures and algorithms aids in processing and analysing text data extracted using these libraries, contributing to skill refinement in database management and data manipulation.

Similarity Calculation: Similarity calculation involves quantifying the likeness between user profiles and job postings to facilitate effective job recommendations. Techniques such as cosine similarity, Jaccard similarity, or Euclidean distance are applied to assess the similarity between the textual representations of user data and job data. For instance, cosine similarity evaluates the angle between the vectors representing user profiles and job postings, with a value close to 1 indicating high similarity. Similarly, Jaccard similarity measures the intersection over the union of terms between user profiles and job descriptions. These measures enable the recommendation system to rank job postings based on their relevance to user preferences and qualifications, facilitating personalized job suggestions.

Maintenance and Improvement: Continuously monitoring and updating the system to adapt to changes in user preferences, job market dynamics, and technological advancements. Overall, the project aims to deliver a robust and scalable job recommendation system that enhances the job search experience for users and improves hiring outcomes for employers by leveraging the power of hybrid filtering models and machine learning algorithms.

4.3 System Architecture

The recommendation system implementation by acquiring a dataset of companies, is a crucial step in the content-based filtering approach. Once the data is gathered, it proceeds to preprocess for analysis. The modules allow the code to quantify the importance of each word in the dataset by considering both its frequency within a specific company profile and its rarity across all profiles. Cosine similarity measures the cosine of the angle between two vectors, in this case, the vectors representing the companies' attributes. This similarity computation results in a similarity matrix, where

each entry represents the similarity score between two companies. Utilizing this similarity matrix, the code generates top-N recommendations using a content-based approach. By identifying companies with attributes at a given target company, the system recommends similar companies to users.

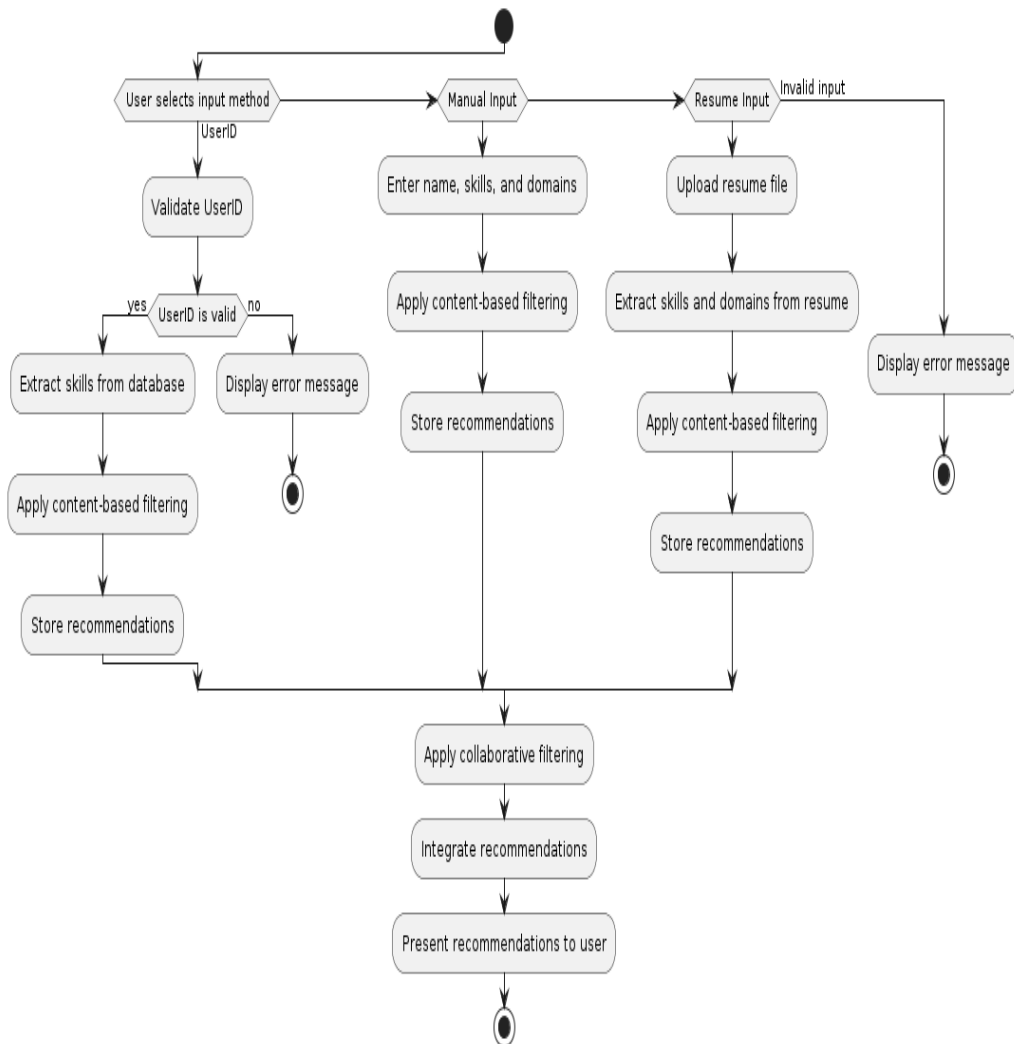


Figure 4.3 System Architecture

Transitioning to Collaborative Filtering, the paper employs two distinct methodologies: a memory-based approach and a model-based approach. In the memory-based approach, the system calculates user-user rating-based similarity. This

involves examining the ratings given by users to different companies and computing the similarity between users based on their rating patterns. Specifically, the paper utilizes Singular Value Decomposition (SVD), a matrix factorization technique, to predict user ratings for unrated jobs based on existing rating data.

After obtaining recommendations from both content-based and collaborative filtering approaches, the program combines the results using a weighted average technique. This fusion aims to leverage the strengths of each method and provide more comprehensive and accurate top-N recommendations to users. Ultimately, we get a Hybrid Recommendation System that integrates content-based and collaborative filtering methodologies, offering users a more robust and personalized recommendation experience.

4.4 Dataset Description

In the job recommendation project, two datasets are utilized: "diec_com_jobs" containing job listings, and "survey_results" storing user preferences and skills. These datasets are crucial for extracting job requirements and user profiles, facilitating the recommendation process. By leveraging both datasets, the system ensures comprehensive matching between user skills and job criteria, enhancing the accuracy and relevance of job recommendations. Integrating these datasets optimizes the recommendation algorithm's effectiveness, resulting in personalized and tailored job suggestions for users.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	
advertiser	company	employe	jobdescrip	jobid	joblocation	jobtitle	postdate	shift	site_name	skills	uniq_id			
https://ww	Digital Int	C2H Corp	- Looking fo	Dice Id : 1	(Atlanta, G	AUTOMAT	1 hour ag	Telecommuting	not a	SEE BELOV	418ff92580b270ef4e7c14f0ddfc36b4			
https://ww	University	Full Time	The Unive	Dice Id : 1	(Chicago, IL	Informatic	1 week ag	Telecommuting	not a	linux/unix,	8aec88cba08d53da65ab99cf20f69d			
https://ww	Galaxy Sys	Full Time	GalaxE.Sol	Dice Id : C	Schaumbu	Business S	2 weeks a	Telecommuting	not a	Enterprise	46baa1f69ac07779274bcd90b85d9a			
https://ww	TransTech	Full Time	Java Devel	Dice Id : 1	(Bolingbroc	Java Devel	2 weeks a	Telecommuting	not a	Please see	3941b2f206ae0f900c4fba4ac0b1871			
https://ww	Matrix Res	Full Time	Midtown l	Dice Id : m	Atlanta, G	DevOps Er	48 minute	Telecommuting	not a	Configurat	45efa1f6bc65acc32bbb953a1ed13k			
https://ww	Yash Tech	Full Time	We are loc	Dice Id : 1	(Chicago, IL	SAP FICO /	2 weeks a	Telecommuting	not a	FICO, AR,	/e0ac9d926dda5e95162ef05adea731			
https://ww	Noble1	Full Time	Network E	Dice Id : 9	(Atlanta, G	Network E	1 hour ag	Telecommuting	not a	Cisco, DNS	e7e326053c586bd94e59f1fd74de4a			
https://ww	Bluebeam	Full Time	Bluebeam	Dice Id : 1	(Chicago, IL	Sr. Web A	2 weeks a	Telecommuting	not a	.NET , C#	, lb0dade4c3c2beecb9c773ca11ecda			
https://ww	Genesis10	Full Time	This is a fu	Dice Id : g	New York, Front	End	7 hours ag	Telecommuting	not a	C++, Devel	28f5e0c1cc3314813e674f0c32b04d1			
https://ww	VanderHoi	C2H W2, C	SummaryC	Dice Id : v	Seattle, W	Applicatio	7 hours ag	Telecommuting	not a	(See Job D	95c9127e2770172f454f3b83981eaa			
https://ww	Maxonic, I	Full Time	JOB DESC	Dice Id : 1	(Sunnyvale	OpenStack	7 hours ag	Telecommuting	not a	Openstack	5e0ff38f5eaf44726f4e3d1dd257a24			
https://ww	CSI (Consu	Full Time	Must have	Dice Id : 1	(Highlands	9001 Data	7 hours ag	Telecommuting	not a	Unix, IAM,	e4a1ff1b6c0fda5f345e57cf1acb40dd			
https://ww	VanderHoi	Full Time	VanderHoi	Dice Id : v	Portland, (Software E	7 hours ag	Telecommuting	not a	Java, OSS	d0c81a2e3e5d666f3d730f1048c491			
https://ww	Genesis10	Full Time	Genesis10	Dice Id : g	Los Angele	Sales Engi	7 hours ag	Telecommuting	not a	Consulting	51279d060da242e3baea98b26ddd6			

Figure 4.1 dice-com-jobs datasets

The "Dice.com Jobs Dataset" on Kaggle provides detailed information about job postings from the Dice.com job board, focusing primarily on technology-related roles.

Here are some specific features included in the dataset:

Job Title: Describes the title or position of the job listing.

Company: Specifies the name of the hiring company or organization.

Location: Indicates the geographical location or city where the job is based.

Employment Type: Specifies the type of employment, such as full-time, part-time, contract, or freelance.

Job Description: Provides a detailed description of the job responsibilities, requirements, and qualifications.

Skills Required: Lists the specific skills or technologies required for the job, such as programming languages, software tools, or domain knowledge.

Job Category: Categorizes the job listing into specific domains or industries, such as software development, data science, IT infrastructure, etc.

Salary: Indicates the salary range or compensation package offered for the job.

Experience Level: Specifies the level of experience required for the job, such as entry-level, mid-level, or senior-level positions.

Education Level: Specifies the minimum educational qualifications required for the job, such as bachelor's degree, master's degree, or relevant certifications.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Response	Hobby	OpenSour	Country	Student	Employment	FormalEd	Undergradi	Company	DevType	YearsCodi	YearsCodi	JobSatisfa	CareerSat	HopeFive	JobSearch	LastNew
1	Yes	No	Kenya	No	Employed	Bachelor	Mathema	20 to 99	Full-stack	3-5 years	3-5 years	Extremely	Extremely	Working i	l am activ	More tha
3	Yes	Yes	United Kir	No	Employed	Bachelor	A natural	10,000 or	Database	30 or mor	18-20 yea	Moderate	Neither s	Working i	l am activ	More tha
4	Yes	Yes	United St	No	Employed	Associate	Computer	20 to 99	Engineeri	24-26 yea	6-8 years	Moderate	Moderate	Working i	l am activ	More tha
5	No	No	United St	No	Employed	Bachelor	Computer	100 to 49	Full-stack	18-20 yea	12-14 yea	Neither s	Slightly di	Working i	l am activ	More tha
7	Yes	No	South Afr	Yes, part	Employed	Some coll	Computer	10,000 or	Data or bi	6-8 years	0-2 years	Slightly sa	Moderate	Working i	l am activ	More tha
8	Yes	No	United Kir	No	Employed	Bachelor	Computer	10 to 19	Back-end	6-8 years	3-5 years	Moderate	Slightly sa	Working i	l am activ	More tha
9	Yes	Yes	United St	No	Employed	Some coll	Computer	10,000 or	Back-end	9-11 year	0-2 years	Slightly sa	Moderate	Working i	l am activ	More tha
10	Yes	Yes	Nigeria	No	Employed	Bachelor	Computer	10 to 19	Designer	10-2 years	3-5 years	Slightly sa	Moderate	Working i	l am activ	More tha
11	Yes	Yes	United St	No	Employed	Some coll	Fine arts	100 to 49	Back-end	30 or mor	21-23 yea	Moderate	Moderate	Doing the	l am activ	More tha
16	No	Yes	India	No	Employed	Bachelor	Computer	500 to 99	Designer	0-2 years	NA	NA	NA	NA	NA	NA
17	Yes	No	Spain	No	Employed	Master	informati	1,000 to 4	Back-end	6-8 years	0-2 years	Moderate	Moderate	Doing the	l am activ	More tha
18	Yes	Yes	India	No	Employed	Bachelor	Another	100 to 49	Back-end	0-2 years	NA	NA	NA	NA	NA	NA
19	Yes	No	Croatia	NA	Employed	Some coll	Computer	20 to 99	Back-end	0-2 years	NA	NA	NA	NA	NA	NA
20	No	No	India	No	Employed	Bachelor	Another	20 to 99	Back-end	3-5 years	3-5 years	Extremely	Moderate	Working i	l am activ	More tha
21	No	No	Netherlar	Yes, full-ti	Employed	Secondar	NA	20 to 99	Back-end	0-2 years	0-2 years	Neither s	Moderate	Working i	l am activ	More tha
22	Yes	No	Israel	No	Employed	Master	Fine arts	5,000 to 9	Full-stack	3-5 years	3-5 years	Moderate	Extremely	Working i	l am activ	More tha
26	No	No	United St	Yes, full-ti	Employed	Bachelor	Computer	1,000 to 4	Student	0-2 years	NA	NA	NA	NA	l am activ	Less tha
27	Yes	No	Sweden	No	Employed	Master	A busines	10 to 19	Back-end	6-8 years	0-2 years	Moderate	Moderate	Working i	l am activ	More tha
29	Yes	Yes	India	Yes, full-ti	Employed	Bachelor	NA	10,000 or	Data or bi	0-2 years	3-5 years	Slightly sa	Extremely	Doing the	l am activ	More tha
31	Yes	Yes	Chile	Yes, full-ti	Employed	Bachelor	A busines	Fewer tha	Back-end	3-5 years	NA	NA	NA	NA	NA	NA
33	Yes	Yes	Australia	No	Employed	Bachelor	Another	1,000 to 4	Database	15-17 yea	12-14 yea	Slightly sa	Slightly di	Working i	l am activ	More tha

Figure 4.2 survey results dataset [1]

AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU									
Currency	Salary	SalaryType	Converter	Currency	Community	TimeFully	Educator	SelfTaught	TimeAfter	Hackathons	AgreeDisagree	AgreeDisagree	AgreeDisagree	Language	Language	Database	Database	Platform	Platform	Framework	Framework									
NA	NA	Monthly	NA	KES	Slack	One to th	Taught yo	The offici	NA	To build	Strongly a	Strongly a	Neither A	JavaScript	JavaScript	Redis	SQL	Redis	SQL	AWS	Azur	Django	Re Django							
British po	51000	Yearly	70841	GBP	Confluent	One to th	Taught yo	The offici	NA	NA	Agree	Agree	Neither A	JavaScript	Go	Python	Redis	PostgreSQL	Linux	Linux	Django	React								
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA								
U.S. dolla	NA	NA	NA	NA	NA	Three to	Complete	The offici	NA	NA	Disagree	Disagree	Strongly c	C#	JavaScript	C#	JavaScript	SQL	Serve	SQL	Serve	Azure	Azure	Angular						
South Afr	260000	Yearly	21426	ZAR	Office / pi	Three to	Taken a p	The offici	NA	NA	Strongly a	Agree	Strongly c	C++	Java	Assembly	SQL	Serve	PostgreSQL	Arduino	V Arduino	VNA	NA							
British po	30000	NA	41671	GBP	Confluent	Less than	Received	The offici	NA	To impro	Disagree	Neither A	Strongly c	JavaScript	C#	Go	Java	MongoDB	PostgreSQL	Linux	Linux	Angular	Node.js							
U.S. dolla	120000	Yearly	120000	USD	Confluent	Six to nin	Received	The offici	NA	NA	Disagree	Agree	Strongly c	JavaScript	C	Go	Java	MongoDB	NA	Linux	Linux	Node.js	R	React	Tr					
NA	NA	NA	NA	NA	Facebook	One to th	Taken an	NA	Immediat	To impro	Strongly a	Strongly c	Neither A	JavaScript	Matlab	SC	MongoDB	NA	Azure	Her	Amazon	E	Angular	N	NET Co					
U.S. dolla	250000	Yearly	250000	USD	Confluent	Three to	Taken an	The offici	NA	Because	Strongly a	Strongly c	Strongly c	Assembly	Erlang	Go	Redis	Pos	Redis	Pos	Amazon	E	AWS	Linu	Hadoop	NA				
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					
Euros (a,-	0	Monthly	0	EUR	NA	Less than	Received	NA	NA	NA	Disagree	Strongly c	Strongly a	JavaScript	Java	NA	SQL	Serve	NA	Windows	NA	NA	NA	NA	NA					
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					
Swedish k	32000	Monthly	47904	SEK	Confluent	Three to	Taken an	A book or	NA	To impro	Neither A	Agree	Disagree	C#	SQL	HT	C#	F#	Has	SQL	Serve	Redis	SQL	Linux	Win	Linux	Win	NET	Core	NET Co
NA	NA	Monthly	NA	NA	NA	Confluent	Three to	Taken an	NA	NA	Strongly a	Strongly a	Strongly a	C++	C#	Python	R	NA	Redis	Pos	NA	Azure	ESP	Angular	Django	NA				
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA				
Australian	120000	Yearly	95968	AUD	Confluent	Less than	Taken an	The offici	NA	To impro	Agree	Neither A	Strongly c	C++	Go	C++	Go	Redis	Pos	Redis	Pos	Android	Android	Angular	Angular	Angular				

Figure 4.3 survey results dataset [2]

The "survey results" dataset contains essential features that capture important information about respondents' skills, preferences, and demographics. Here are some of the key features included in the dataset:

Respondent: An identifier for each survey respondent.

Country: Indicates the country where the respondent is located.

Age: Specifies the age of the respondent, providing demographic information.

Gender: Captures the gender identity of the respondent.

Education: Describes the highest level of education attained by the respondent.

Job Role: Specifies the current job role or position held by the respondent.

Years of Experience: Indicates the number of years of professional experience in the field.

Programming Language Proficiency: Measures the proficiency level in various programming languages.

Framework Proficiency: Indicates the proficiency level in different software frameworks or libraries.

Database Proficiency: Specifies the proficiency level in database management systems.

Preferred Work Environment: Captures preferences regarding work environment, such as remote work, flexible hours, etc.

Salary Expectation: Specifies the expected salary range or compensation package for job opportunities.

Job Search Status: Indicates whether the respondent is actively looking for job opportunities.

Job Satisfaction: Measures the level of satisfaction with the current job role or employment situation.

Career Aspirations: Captures aspirations or career goals for future professional development.


Code URL :

5 Experimental Results

In this section, experiments are conducted to assess the performance of the Hybrid Filtering Model, validating its effectiveness. Parameters are meticulously analyzed to understand their impact on the model's outcomes. Additionally, feature comparisons with similar models are conducted to elucidate the advantages of our approach more comprehensively. The proposed method is implemented and simulated using Python Jupyter Notebook, leveraging its interactive and intuitive environment for experimentation. The obtained results provide valuable insights into the model's performance, affirming its efficacy in providing accurate job recommendations.

```
282 obj=job_postings("./data/dice_com-job_us_sample.csv")
283 rows_1=obj.match_profile("./data/",432,0,1)
284 if checker==1:
285     display(rows_1)
286 print(rows_1)
```

Execute the next cell after file uploaded

 Upload File (0)

FileUpload(value={}, accept='.pdf,.docx', description='Upload File')

Figure 5.1 To take file as an input

	jobdescription	jobid \
5179	Minimum Required Skills:Perl, OO Perl, Cloud C...	Dice Id : cybercod
14245	Job Description Location: Quincy, MAOracle A...	Dice Id : delliirc
21011	We have following urgent role with our DIRECT...	Dice Id : 10170091
21066	We have following urgent role with our DIRECT...	Dice Id : 10170091
670	Accuity/NRS Software Engineer Accuity is a ma...	Dice Id : 10493862
20489	This is a great opportunity for a Sr Engineer ...	Dice Id : 10286986
4420	Description of position The ServiceNow Archite...	Dice Id : COSWA001
15548	Angular JS/ Fullstack Software Engineer Full t...	Dice Id : prutx001
15651	Title: Application Support EngineerOpenings: M...	Dice Id : 90913413
20252	**Unable to sponsor at this time** Senior Full...	Dice Id : 909SoCon

	joblocation_address	jobtitle \
5179	Houston, TX	Software Engineer, Profitable leader in cloud ...
14245	Quincy, MA	Oracle Application Express APEX Developer Quin...
21011	Sunnyvale, CA	Full-stack Engineer - Java
21066	Sunnyvale, CA	Full-stack Engineer - Java
670	Boston, MA	Software Eng - .NET, full-stack dev, perm hire...
20489	San Carlos, CA	Principal Node.js for Visual Collaboration Sof...
4420	Issaquah, WA	Solution Architect IT Management Platform (Ser...
15548	Richardson, TX	Angular JS Full Stack Developer
15651	Scottsdale, AZ	Application Support Engineer
20252	San Francisco, CA	Sr. Full Stack Developer

Figure 5.2 Output after processing resume

```
obj=job_postings("./data/dice_com-job_us_sample.csv")
rows_1=obj.match_profile("./data/",456,1,0)
# rows_1
if checker==1:
    display(rows_1)
```

New user!Enter details..

Enter Full Name: uma

Enter skills(comma separated). These are programming languages, frameworks,platforms or databases you have experience with c,java,python,mysql,html,css, javascript

Enter domain(s) of interest separated by commas(Names are case sensitive). Should be one of the following:
Back End,C-suite executive,Cloud Computing,Data Scientist,Data or business analyst,Database Administrator(DBA),Designer,DevOps,Educator or academic rese
archer,Embedded applications or devices developer,Engineering manager,Enterprise application,Front End,Full stack,Game developer,Information Security,Mo
bile Developer,Network Engineer,Product Manager,QA/Test Developer,Sales professional,Software Developer/Java Developer,Student,System Administrator,Web
Developer,

Student

Figure 5.3 Manual entry of data

```
rows_2=obj.match_profile("./data/",suser,0,0)
combined_df = pd.concat([rows_1, rows_2])
combined_df=combined_df.drop(columns=['postdate'])#dropping the column
combined_df=combined_df.drop_duplicates()
combined_df.head(15)
```

The user is most similar to the user 1

[9]:

	advertiserurl	company	employmenttype	jobstatus	jobdescription	jobid	joblocation_address	jobtitle	
3409	https://www.dice.com/jobs/detail/Java-Develope...	Resource 1		Full Time	Job Requirements: Must have 5 + yr...	Dice Id : 10421209	Parsippany, NJ	Java Developer	Telecomm available not rec
4572	https://www.dice.com/jobs/detail/Senior-Softwa...	Zulily		Full Time	zulily is a retailer obsessed with bringing ou...	Dice Id : 10453116	Seattle, WA	Senior Software Engineer, Member Engagement PL...	Telecomm available not rec
4777	https://www.dice.com/jobs/detail/Software-Engi...	Zulily		Full Time	zulily is a retailer obsessed with bringing ou...	Dice Id : 10453116	Seattle, WA	Software Engineer II, Site Frontend	Telecomm available not rec
4805	https://www.dice.com/jobs/detail/Software-Engi...	Zulily		Full Time	zulily is a retailer obsessed with bringing ou...	Dice Id : 10453116	Seattle, WA	Software Engineer II, Site Frontend	Telecomm available not rec

Figure 5.4 Output after processing user data

```
rows_2=obj.match_profile("./data/",suser,0,0)
combined_df = pd.concat([rows_1, rows_2])
combined_df=combined_df.drop(columns=['postdate'])#dropping the column
combined_df.head(15)
```

The user 456 is most similar to the user 2635
Respondent 456 is working in
['Dimension Consulting']
Respondent 2635 is working in
['AltaSource Group']

[9]:

	advertiserurl	company	employmenttype	jobstatus	jobdescription	jobid	joblocation_address	jobtitle	shift
21520	https://www.dice.com/jobs/detail/SAP-APO-Consu...	TRUGlobal	Contract Corp-To-Corp, Contract W2, 6-12 Months		***DIRECT CLIENT***SAP Support analyst APO (PL...	Dice Id : 10113269	Sunnyvale, CA	DIRECT CLIENT NEED SAP APO Consultant	Telecommuting not available Travel required to...
39	https://www.dice.com/jobs/detail/Mobile-Automa...	Centizen	Full Time, Contract Corp-To-Corp, Contract Ind...		Centizen Inc, Global agile IT solutions and co...	Dice Id : 10429942	Beaverton, OR	Mobile Automation Tester , Rate :Open Negotiab...	Telecommuting not available Travel not required
13668	https://www.dice.com/jobs/detail/IAM-Technical...	MBA IT Consulting Services, Inc.	Full Time, Permanent only.		Locals only! Permanent only!U.S. Citizens and ...	Dice Id : 10115382	Dallas, TX	IAM Technical Lead (permanent position, locals...	Telecommuting not available Travel not required
2451	https://www.dice.com/jobs/detail/Oracle-ATG-We...	Apps IT LTD	Full Time		Our client requires an Oracle ATG Web Commerce...	Dice Id : 90916035	New York, NY	Oracle ATG Web Commerce	Telecommuting not available Travel not required

Figure 5.5 Output when input is user_id

6 Conclusion

Hybrid filtering in job recommendation systems combines multiple recommendation techniques to enhance accuracy, coverage, and user experience. By integrating collaborative filtering and content-based filtering, it leverages the strengths of each approach while mitigating their weaknesses. Collaborative filtering analyses user behaviour and preferences to recommend jobs based on similarities between users. However, it often struggles with the cold start problem, where new users or items lack sufficient data for accurate recommendations. Content-based filtering, on the other hand, focuses on the attributes of jobs and users to make recommendations. It can address the cold start issue but may suffer from the sparsity problem when dealing with large datasets.

Overall, hybrid filtering in job recommendation systems offers a powerful solution for matching job seekers with relevant opportunities, benefiting both candidates and employers in navigating the complexities of the job market. The benefits of hybrid filtering include enhanced accuracy, increased coverage of job listings, improved robustness against data sparsity and cold start issues, and better user satisfaction due to personalized recommendations. Furthermore, the modular nature of hybrid filtering makes it scalable and adaptable to evolving data environments and user preferences.

7 Future Scope

The future scope of hybrid filtering in job recommendation systems is rich with potential for advancement and innovation. One avenue of exploration involves leveraging advanced machine learning techniques, such as deep learning and reinforcement learning, to further refine recommendation accuracy and personalization. These methods can extract intricate patterns and insights from user behaviour and job attributes, enhancing the quality of recommendations. Another promising direction is the integration of contextual information into recommendation algorithms. By considering factors like user location, time of day, and device type, hybrid filtering can deliver more relevant and timely job suggestions, improving user experience and engagement.

Furthermore, the incorporation of multi-modal data types, including text, images, audio, and video, holds promise for providing richer job recommendations. By analyzing diverse sources of information such as job descriptions, resumes, and multimedia content, hybrid filtering can gain a deeper understanding of user preferences and job requirements. Dynamic ensemble methods present another opportunity for improvement, allowing recommendation algorithms to adaptively adjust the weights assigned to different techniques based on their performance and relevance to specific users or scenarios. This approach optimizes recommendation quality and robustness over time.

8 Bibliography

- [1] Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik, Avick Kumar Dey, Prasenjit Choudhury. “Recommender Systems: An Overview, Research Trends, and Future Directions,” 2021.
- [2] Dr. Alka Singhal Shivangi Rastogi, Nikhil Panchal, Shivani Chauhan, Shradha Varshney. “Research Paper On Recommendation System,” vol. 9, no. 8, August 2021.
- [3] Ravita Midhra, Sheetal Vikram Rathi, “Efficient and Scalable Job Recommender System Using Collaborative Filtering,” *Researchgate*, 19 May 2020.
- [4] Tanya V. Yadalam, Vaishnavi M. Govda, Vandhita Shiva Kumar, Disha Girish, “Career Recommendation Systems using Content based Filtering,” in *5th International Conference on Communication and Electronics Systems (ICCES)*, 10 July 202.
- [5] Marwa Hussien Mohamed, Mohamed Helmeey Khafagy, Mohamed Hasan Ibrahim, “Recommender Systems Challenges and Solutions Survey,” in *International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2 February 2019.
- [6] Greg Linden, Brent Smith, Jeremy York, “Amazon.com Recommendation,” in *IEEE Computer Society*, 2003.
- [7] Kunal Shah, Akshaykumar Salunke, Saurabh Dongare, Kisandas Antala, “Recommender Systems: An Overview of different approaches to recommendations,” in *International Conference on Innovations in information Embedded and Communication Systems (ICIIECS)*, 2015.
- [8] Thiengburanathum P, Cang S, Yu H, “An overview of travel recommendation system,” in *IEEE 22th international conference on automation and computing*, 2016.

- [9] J. Karim, "Hybrid Systems for Personalized Recommendations, Research Challenges in Information Science (RICS)," in *2014 IEEE Eighth International Conference*, May 2014.
- [10] E K Subramanian, Ramachandran. "Student Career Guidance System: Recommendation of a course," *International Journal of Recent Technology and Engineering*, vol. 7, no. 6S4, 2019.
- [11] Manish Kumar Singh, Dr. Dinesh Prasad Sahu. "Research Aspects Of The System," *International Journal For Research In Applied Science Engineering Technology (IJRASET)*, vol. 5, no. XI, November 2017.
- [12] Bhumika Bhatt, Prof. Premal J Patel, Prof. Hetal Gaudani, "A Review Paper on Machine Learning Based Recommendation System," *International Journal Of Engineering Development And Research*, vol. 2, no. 4, 2014.
- [13] Kaveri Roy, Aditi Choudhary, J. Jayapradha, "Product Recommendations using Data Mining And Machine Learning Algorithms," *ARPJ Journal Of Engineering And Applied Sciences* , vol. 12, no. 19, October 2017.