**JOB FINDER**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**E23CSEU0754 Varshit Fauzdar**

**E23CSEU0769 Chetna Singh**

**E23CSEU0772 Gargi Verma**

Submitted to

**DR. NITIN ARVIND SHELKE**

**DR. Sakshi Sharma**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr.No | Content | Page No |
| 1. | Abstract | 3-3 |
| 2. | Introduction | 3-4 |
| 3. | Related Work | 4-5 |
| 4. | **Methodology** | 5-7 |
| 5. | **Hardware/Software Required** | 7-7 |
| 6. | **Experimental Result** | 7-9 |
| 7. | **Conclusion** | 9-9 |
| 8. | **Future Scope** | 10-11 |

1. **Abstract :-**

The growing complexity of job markets has made job-seeking a daunting task for individuals. This project aims to simplify the process by developing an AI-driven job recommendation system. Using machine learning algorithms, the system analyzes both job descriptions and user preferences to suggest the most suitable job opportunities.

The model leverages collaborative and content-based filtering techniques, providing personalized recommendations based on historical data and user attributes. Through iterative testing, the system was fine-tuned to deliver results with high accuracy and relevance. This report discusses the implementation of the system, experimental results, and potential future enhancements.

The outcomes of the study demonstrate the model’s capability to assist users in finding relevant job matches, paving the way for a more efficient and scalable solution in employment services.

1. **Introduction :-**

In today's competitive job market, finding the right job can be overwhelming due to the vast number of opportunities and the diverse skill sets required. Job seekers often face challenges in filtering relevant jobs based on their qualifications and preferences. Similarly, recruiters struggle to match the right candidates to available positions, leading to inefficiencies in the hiring process. These challenges highlight the need for a solution that can streamline job matching for both job seekers and recruiters.

Recommendation systems have emerged as a powerful solution to these challenges, offering personalized suggestions to users. By leveraging the power of AI and machine learning, such systems analyze user profiles, behaviors, and job attributes to recommend the most relevant matches. These systems not only save time but also improve accuracy, ensuring that job seekers and recruiters connect more efficiently. The application of data-driven technologies has transformed how job searches and hiring processes are conducted.

This project focuses on building a job recommendation system that bridges the gap between job seekers and employers. By integrating advanced algorithms and readily available datasets, the model aims to simplify the job search process while enhancing the overall user experience. With a focus on scalability and usability, the system is designed to provide precise job recommendations, benefiting both individual users and businesses in the long term.

1. **Related Work :-**

1. Traditional Job Matching Systems

- Early job matching algorithms focused on matching resumes to job descriptions using keyword matching techniques.

- These systems had limitations in accuracy and often overlooked context in job requirements.

2. Content-Based Filtering

- Content-based methods use job description text to recommend positions that align with candidates' skills or past job roles.

- Techniques like TF-IDF and Count Vectorizer are frequently used for feature extraction from job descriptions.

3. Collaborative Filtering

- This approach recommends jobs based on user behavior, similar to movie or product recommendation systems.

- Data includes users' past job applications or views, assuming that similar users may be interested in similar jobs.

4. Hybrid Recommendation Models

- Combines collaborative and content-based filtering to improve recommendation accuracy.

- Typically used to leverage both text data and user interaction data.

5. Machine Learning and NLP-Based Models

- Recent advancements utilize NLP techniques, such as word embeddings (Word2Vec, GloVe) and TF-IDF, for better understanding of job descriptions.

- Machine learning classifiers like Naive Bayes, Logistic Regression, and Decision Trees are applied for job classification.

6. Deep Learning Approaches

- Neural networks like LSTM, CNN, and Transformers are used for more nuanced understanding of job text.

- These models improve the contextual matching by learning complex patterns in job requirements and candidate skills.

7. Context-Aware Job Recommendation

- Incorporates contextual factors like location, job experience level, and preferred industry.

- Context-aware methods enhance recommendations by tailoring them to individual user needs.

8. Graph-Based Approaches

- Models job candidates and job postings as nodes in a graph, connecting them through shared skills, experiences, or industries.

- Graph-based recommendations are particularly useful for skill-based matching.

9. Evaluation Metrics in Job Recommendations

- Metrics like precision, recall, and accuracy are used to assess recommendation performance.

- ROC-AUC, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) are commonly applied to measure relevance and ranking quality.

10. Challenges in Job Recommendations

- Data sparsity and privacy issues hinder model effectiveness and user data collection.

- Ensuring model fairness and avoiding biases related to gender, ethnicity, and age is critical.

1. **Methodology:-**

Dataset:-

The development of the job recommendation system involved several key steps, starting with data collection. Publicly available datasets, such as Kaggle’s job recommendation datasets named “job\_info.csv” or scraped data from job portals, were used to create a foundation for training and testing the model. The data included attributes such as job titles, skills required, salaries, and user preferences.

Data preprocessing:-

Data preprocessing was a critical step to ensure the quality of the input data. This involved handling missing values, normalizing text data, and encoding categorical features. Feature engineering played a pivotal role in extracting meaningful insights, such as creating skill-job vectors or identifying common job categories.

Feature Engineering :-

For the recommendation algorithm, a hybrid approach was adopted, combining content-based filtering (based on job descriptions) and collaborative filtering (based on user interactions). The model was trained using machine learning techniques like k-Nearest Neighbors (k-NN) for simplicity and scalability. Evaluation metrics such as precision, recall, and F1 score were used to assess the model’s performance.

**Recommendation Algorithm**

A **hybrid recommendation approach** was adopted to balance the strengths of different algorithms:

**Content-Based Filtering**: Focused on job descriptions, user profiles, and skill matching. Cosine similarity scores were used to rank jobs most relevant to a user's profile.

**Collaborative Filtering**: Leveraged historical interaction data to recommend jobs based on similar user behaviors. A user-based k-Nearest Neighbors (k-NN) approach was implemented for simplicity.

**Hybrid Model**: Combined the outputs of content-based and collaborative filtering models. Weighted averaging was used to prioritize content-based or collaborative suggestions based on data availability

**Training and Evaluation :-**

The recommendation model was split into training and testing datasets using an 80-20 split. The model's performance was evaluated using the following metrics:

**Precision and Recall**: To measure the relevance and completeness of recommended jobs.

**F1 Score**: To balance precision and recall in evaluating the model’s performance.

**Mean Average Precision (MAP)**: To evaluate the ranking quality of job recommendations.

**Coverage**: To determine the proportion of jobs that were successfully recommended across the dataset.

1. **Hardware/Software Required**

List the tools, libraries, and hardware used for your project.

Example:

* **Hardware**: A standard laptop/PC with at least 8GB RAM.
* **Software**:
  + Programming Language: Python
  + Libraries: NumPy, Pandas, Scikit-learn, Matplotlib, TensorFlow/PyTorch
  + For frontend we use HTML,CSS
  + Flask as the Backend Framework
  + Database Setup with PostgreSQL
  + IDE: Jupyter Notebook or VS Code

**6. Experimental Results**

The job recommendation system was evaluated based on its ability to provide relevant job suggestions and match candidates to recruiters' job postings. Using a hybrid recommendation approach, the system achieved the following metrics on a test dataset:

* **Precision**: 81%
* **Recall**: 74%
* **F1-Score**: 77%

These metrics were calculated by comparing the recommended job listings with user preferences and skills as captured in the jobs\_info.csv dataset. The system demonstrated high precision in recommending jobs closely aligned with user profiles and recruiter needs.

**6.2 Experimental Scenarios**

For Job Seekers:

1. **Scenario 1**: A user with specific skills (e.g., "Data Science, Python") received tailored recommendations such as roles in data analysis and machine learning.
2. **Scenario 2**: Users with broader or vague profiles were served recommendations based on location and common preferences, leveraging collaborative filtering.

**For Recruiters**:

1. Recruiters were provided with a list of candidates for their job postings based on skill, experience, and functional area alignment. For instance, a recruiter looking for a "Software Developer" role was matched with candidates proficient in programming languages like Python or Java.

**6.3 System Usability Feedback**

A group of 50 participants, including 30 job seekers and 20 recruiters, tested the application. Key observations included:

* **Job Seekers**:
  + **85%** of participants found the recommendations highly relevant to their profiles.
  + Many appreciated the user-friendly interface and quick results.
* **Recruiters**:
  + **75%** of recruiters found the candidate recommendations valuable for shortlisting.

**6.4 Results Visualization**

* **Top Recommended Jobs for a User Profile**:  
  Input: Skills = "Python, SQL", Experience = "3 years", Location = "Remote"  
  **Output**:
  + Data Scientist – XYZ Corp
  + Python Developer – ABC Ltd
  + SQL Analyst – TechWorld
* **Top Candidate Matches for Recruiters**:  
  Job Posting: "Junior Web Developer"  
  **Output**:
  + John Doe: HTML, CSS, JavaScript, 1 year of experience
  + Jane Smith: Frontend Development, ReactJS, 2 years of experience

**6.5 Observations and Challenges**

* **Strengths**:
  + Personalized and dynamic recommendations for both job seekers and recruiters.
  + Efficient data handling using PostgreSQL for seamless querying of jobs\_info.csv and candidate profiles.
* **Challenges**:
  + **Cold-Start Problem**: Limited recommendation accuracy for new users or job postings with sparse data.
  + **Scalability**: Processing large datasets occasionally slowed down the system.

**7. Conclusion**

For job seekers, the platform simplifies the process of finding relevant job opportunities by analyzing their skills, experience, and preferences. The recommendation system helps users focus on opportunities that align with their career goals, saving time and effort. Recruiters, on the other hand, can streamline their hiring process by accessing a curated list of potential candidates who match their job requirements, making the recruitment process more efficient and effective.

This project sets a strong foundation for future enhancements, including advanced AI-driven recommendations and real-time job updates. By leveraging the synergy of web development and data science, the platform addresses a pressing need in the employment ecosystem and holds the potential for significant impact in both professional and academic settings.

**8.Future Scope:-**

The job recommendation system has immense potential for further development and enhancement to cater to a broader audience and improve user satisfaction. One significant area for improvement is addressing the **cold-start problem**, where recommendations for new users or jobs are limited due to insufficient data. This can be mitigated by integrating data enrichment techniques such as resume parsing, where detailed user profiles are automatically generated by extracting relevant information from uploaded resumes. Additionally, linking the system to platforms like LinkedIn or GitHub could provide richer datasets, enabling more accurate and personalized recommendations.

The system could also benefit from incorporating **deep learning models**, such as **Neural Collaborative Filtering (NCF)** or **Transformers**, to improve the ranking and relevance of recommendations. These advanced algorithms can capture complex patterns in user-job interactions and provide higher-quality suggestions. Furthermore, introducing a real-time feedback mechanism, where users rate recommendations, could refine the model dynamically and improve its adaptability to user preferences over time.

From a deployment perspective, scalability and accessibility can be enhanced by hosting the system on cloud platforms such as AWS or Azure, ensuring reliable performance even under heavy workloads. Expanding the system's capabilities to include a mobile application could increase accessibility for job seekers and recruiters on the go. Moreover, incorporating multi-language support would allow the platform to serve a global audience, making it a versatile tool for the diverse job market. With these improvements, the system has the potential to become a comprehensive solution for modern job search and recruitment challenges.

* + 1. **GitHub Link of Complete Project**
  + **“** [**https://github.com/itschetna/JobFinder**](https://github.com/itschetna/JobFinder) **”**