IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AI Resume Parser And Job Recommendation System

[1]Amrita Sinha, [2]Deepak Kumar Thakur L, [3]Chandana C K, [4]Sanjana S,

[1]Assistant Professor,[2]Student,[3]Student,[4]Student,

[1] Department of Information Science,

[1]T John Institute of Technology, Bengaluru, India

Abstract: The growing number of job listings and the wider range of skill requirements have made it difficult for job seekers to find opportunities that match their qualifications. This project introduces an AI-driven Job Recommendation System aimed at simplifying the process of connecting candidates with suitable jobs. The system uses Natural Language Processing (NLP) and Machine Learning (ML) techniques to effectively analyze resumes and job descriptions.

Resumes are processed with PyResparser, which extracts key details such as skills, experience, and education. At the same time, job data are prepared and transformed using the TF-IDF (Term Frequency-Inverse Document Frequency) method, allowing the system to assess how similar candidate profiles are to job requirements. The Nearest Neighbors algorithm is then used to find the most relevant job suggestions.

A Flask-based web interface enables users to upload their resumes and receive personalized job recommendations in real time. The system also includes automated data scraping to keep the job database current, ensuring that recommendations are relevant and accurate. This approach removes the need for manual browsing through many job portals and offers a more efficient, user-friendly solution. Experimental results show that the system greatly improves the speed and accuracy of job matching, making it a useful tool for job seekers and recruitment platforms alike.

Index Terms -Resume Parsing, Job Recommendation, Natural Language Processing, Machine Learning, Flask Application, Tf-Idf vectorizer, Automation, Job matching, Web scrapping.

I. INTRODUCTION

In today's fast-changing digital world, the job-hunting process has moved from traditional methods to AI-driven systems. Job seekers now have to sort through a large number of online listings. This makes it hard for them to find positions that match their skills and qualifications. At the same time, recruiters face challenges in finding suitable candidates who meet specific job requirements. To address this issue, the Job Recommendation System was created as an AI-based tool that streamlines the process of analyzing resumes and suggesting relevant job roles to users.

This project uses Natural Language Processing (NLP) and Machine Learning (ML) techniques to improve the recruitment process. The system extracts important information from resumes, such as skills, education, and experience, using libraries like PyResparser and SpaCy. It then compares this information with job descriptions collected from online platforms. By using the TF-IDF vectorization technique and the Nearest Neighbors algorithm, the model finds similarities between candidate profiles and job postings. This generates personalized job recommendations.

Built on the Flask framework, the system features a user-friendly web interface. Users can upload their resumes and quickly get job matches. This saves time and enhances job search experience. In addition, the project includes data cleaning and preprocessing techniques to ensure that both resumes and job descriptions are analyzed accurately. Overall, this system acts as a helpful link between job seekers and employers by using artificial intelligence. It shows how data-driven methods can transform recruitment by providing efficiency, relevance, and personalized job recommendations.

The rest of this paper is organized as follows. Section II presents a literature review. Section III describes the proposed methodology and system design. Section IV discusses the results. Finally, Section V concludes the paper and describe future work.

II. Literature Review

In this section, we review various existing works related to resume parsing and job recommendation systems. These studies mainly use Natural Language Processing (NLP), Machine Learning (ML), and clustering techniques to improve the efficiency and automation of recruitment systems.

The authors in [8] proposed a job recommendation system that relies on the K-Means clustering algorithm. Their research focused on recent graduates with little experience, assisting them in finding relevant positions by analyzing factors such as salary, location, and skills. Experimental evaluation showed that the system achieved a user satisfaction rate of 87.6%, confirming that clustering is an effective recommendation method.

In [3], the authors addressed the job recommendation challenge with a supervised machine learning approach. By examining historical job transition data, they trained a predictive model to identify potential future job opportunities for candidates. The results showed that machine-learned recommendation models significantly outperform traditional heuristic techniques.

A hybrid resume parsing method was introduced in [1], where Modified K-mers and the Firefly Algorithm (FFA) extracted and matched key resume elements. This combination improved feature identification, leading to better accuracy and flexibility for job matching compared to standard NLP systems.

The study in [10] presented a Resume Parser and Summarizer that extracts and condenses essential candidate information. This model automates data extraction and summarizes resumes to highlight important skills and experiences. The framework demonstrated high accuracy and efficiency in HR screening tasks.

In [7], a Bottom-Up approach to job recommendation was proposed. The authors analyzed large employment datasets and incrementally built a recommendation model using collaborative filtering techniques. Their results showed that data-driven insights can greatly enhance the personalization and accuracy of job recommendations.

The research in [5] developed a Resume Parser and Analyzer using NLP to automate the candidate screening process. The system extracted relevant details such as experience and qualifications while ranking resumes by relevancy. It also provided feedback and suggested courses to help candidates improve their profiles.

In [12], the authors designed a resume parsing system that uses Machine Learning and NLP. Their method extracted structured data from unformatted resumes and ranked candidates based on their skills and experience. The proposed system improved the efficiency of shortlisting and enhanced the recruitment decision-making process.

The authors in [2] presented a resume parser that employs Natural Language Processing techniques to extract structured data from raw text. By using tokenization, lemmatization, and entity extraction, they successfully identified key resume attributes like education and experience. The proposed system improved information consistency and reduced manual recruitment efforts.

Another study in [6] discussed an automated job recommendation system developed to match candidate profiles with job postings effectively. The authors evaluated both user-based and item-based collaborative filtering algorithms and ultimately chose the one that delivered the most accurate and relevant results. The system significantly cut down job search time and boosted user satisfaction.

The authors in [9] proposed an NLP-based Resume Parser Analysis that can handle various and complex resume formats. Their system used NLP models to extract critical details and rank candidates effectively. The inclusion of ML improved overall parsing accuracy and lessened the recruiter's workload.

In [11], a resume analysis system was proposed that uses Machine Learning and Natural Language Processing for automated recruitment. The model extracted features, compared candidate profiles, and generated job recommendations. The system showed strong performance and reduced the time needed for HR screening.

Finally, the research in [4] focused on creating a resume parser using NLP for structured data extraction. The system allowed users to upload resumes, review parsed outputs, and store them in a database. The authors emphasized the importance of linguistic preprocessing to enhance parsing quality and support efficient candidate evaluation.

From the reviewed works, it is evident that NLP and ML play a crucial role in changing recruitment automation. While these systems can effectively parse resumes and recommend suitable jobs, they still encounter challenges like managing different formats, understanding context, and adaptive matching. Therefore, the proposed research focuses on developing an integrated AI-based Resume Parser and Job Recommendation System that uses NLP and ML techniques to improve candidate-job alignment and streamline the overall hiring process.

III. METHODOLOGY

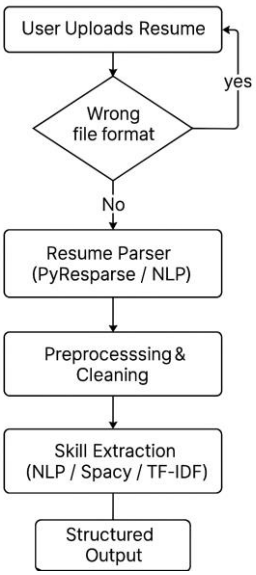

Fig.1. Resume Parser Methodology

The flow diagram in Fig.1 illustrates the workflow of the proposed *AI Resume Parser System*, which automatically extracts and organizes information from resumes. The process begins when the user uploads a resume file through the application interface. The system first validates the uploaded document to ensure that it is in a supported format such as PDF, DOC, or DOCX. If the format is invalid, the user is prompted to re-upload a valid file. Once a valid file is provided, the process proceeds to the parsing stage.

The resume is then processed using tools such as *PyResparser* and Natural Language Processing (NLP) techniques that extract raw text and details including the candidate's name, contact information, education, work experience, and skills. The extracted text is subsequently cleaned to remove special characters, stopwords, and redundant spaces, ensuring uniformity and consistency.

Next, the system performs skill extraction using NLP models such as *SpaCy* and the Term Frequency–Inverse Document Frequency (TF–IDF) method. These techniques identify and categorize relevant skills and keywords from the resume into groups such as technical, managerial, and soft skills. Finally, the processed data is transformed into a structured format (e.g., JSON or CSV), enabling seamless integration with recruitment databases or job recommendation systems.

This structured output ensures efficient storage, retrieval, and analysis of candidate data. Overall, the workflow provides a reliable, scalable, and automated approach to resume parsing, minimizing manual recruitment efforts while improving the precision and accuracy of candidate information extraction.

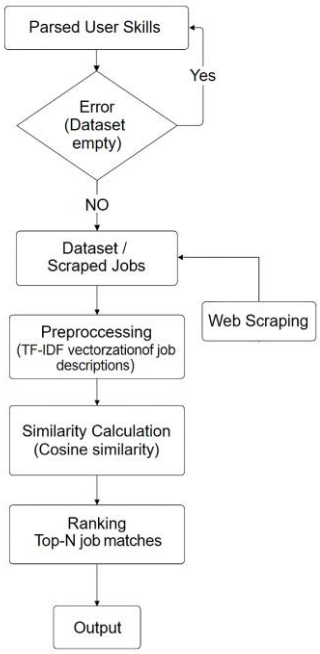

Fig.2. Job Recommendation System Process

• The Job Recommendation System

- Dataset / Scraped Jobs:

The system collects job listings from various online job sites through an automated web-scraping module built using Selenium WebDriver and BeautifulSoup in Python. It fetches real-time job data from platforms like Indeed, LinkedIn, and TimesJobs by launching browser sessions for each job title and location specified in the script. Selenium's scrolling capability is used to load dynamic content, ensuring that both visible and hidden listings are captured. BeautifulSoup then processes the HTML content to extract essential details such as job titles, company names, locations, short descriptions, and application links. The system gathers up to N job postings from each site for every job title and location combination to maintain dataset variety and reduce duplicates. All extracted records are merged and cleaned using Pandas to remove duplicates. The final processed dataset is saved as a UTF-8 encoded CSV file. This updated dataset forms the basis for further stages like preprocessing, TF-IDF vectorization, and similarity matching.

- Preprocessing (TF-IDF Vectorization):

In this stage, the raw job descriptions collected from the web-scraping module are cleaned and converted into structured numerical form to prepare them for similarity analysis. The preprocessing pipeline starts by loading the dataset collected from different job platforms and stored as a CSV file. The text data goes through several natural language processing steps such as tokenization, lowercasing, punctuation removal, and stop-word elimination to ensure clean and consistent text.

After cleaning, the system applies TF-IDF (Term Frequency–Inverse Document Frequency) to convert the text into numerical representations that highlight the importance of significant words. TF-IDF assigns lower weight to common words like "job" or "experience" and higher weight to important technical keywords like "TensorFlow," "React.js," or "SQL." The resulting TF-IDF matrix is then used for cosine similarity calculations, enabling accurate and meaningful skill-to-job comparisons.

-Similarity Calculation (Cosine Similarity):

This stage measures how closely the user's skills match each job description. After converting both the user's extracted skills and the job descriptions into TF-IDF vectors, the system uses cosine similarity to compare them. Cosine similarity is effective for text analysis because it compares the direction of the vectors instead of their size, which avoids issues caused by different document lengths. The similarity score ranges from 0 to 1, where a higher value means a stronger match between the user's skills and the job requirements. The system computes similarity values for all job entries and stores them in a similarity matrix. These scores are then used in the ranking stage to produce personalized job recommendations.

-Ranking (Top-N Job Matches):

In the ranking stage, job recommendations are sorted based on how well they match the user's skills. First, cosine similarity scores between the user's skill vector and each job description are calculated. These scores are sorted in descending order so the best matches appear first. Pandas is used to efficiently organize and rank these scores.

To make the recommendations more personalized, a location filter is applied after ranking, showing only jobs from the user's preferred city or region. Additional optional filters such as company name or job type can also be used. The final ranked list contains the top N job matches—typically the top 10 or 20—based on both skill relevance and location preference. These results are then displayed to the user through an interactive graphical interface.

• The Integration and Development of Graphical User Interface

After developing the individual modules separately, the integration phase combined the Resume Parser and Job Recommendation System into a unified framework. The extracted text and recognized skills from the resume parser were used as inputs for the job recommendation module. This setup enabled the system to generate personalized job recommendations based on the candidate's profile. The recommended jobs were ranked using similarity scores and further filtered by location before being presented as the final output.

Once the integration was completed, a full Graphical User Interface (GUI) was developed to bring all components together. The GUI allows users to upload resumes, extract important information, and view job recommendations in a single interactive platform.

-Hypertext Markup Language:

HTML was used to create the basic structure of the frontend, we used HTML as shown in Figure from 3 to 5.

```
<main class="pt-28 flex flex-col items-center text-center px-5">
  <h2 class="text-5xl font-bold text-gray-900 mb-6 leading-tight">
    Discover Jobs That Match Your <span class="text-blue-600">Skills</span>
  </h2>
  <p class="text-lg text-gray-500 max-w-2xl mb-10">
    Upload your resume and get instantly matched to real job opportunities
    that fit your skills and experience.
  </p>
  <a href="/upload" class="bg-blue-600 text-white px-8 py-4 rounded-xl text-lg
  font-medium hover:bg-blue-700 transition-all">
    Upload Your Resume →
  </a>

  <img src="https://cdn.dribbble.com/users/1162077/screenshots/3848914/programmer.gif"
       alt="resume scan" class="mt-12 w-[500px] rounded-2xl shadow-lg">
</main>
```

Fig. 3. HTML code snippet (Home page)

```
<form id="uploadForm" action="{{ url_for('submit_data') }}" method="POST"
enctype="multipart/form-data" class="space-y-6">
  <input type="file" name="userfile" accept=".pdf,.docx"
         class="w-full border-2 border-dashed border-gray-300 rounded-lg
         p-5 text-gray-500 cursor-pointer hover:border-blue-500 transition"
         required>
  <button type="submit" class="w-full bg-blue-600 text-white py-3 rounded-xl
  font-semibold text-lg hover:bg-blue-700 transition">
    🔍 Analyze My Resume
  </button>
</form>


  <div id="loading" class="hidden mt-8 flex flex-col items-center">
    <img src="https://assets2.lottiefiles.com/private_files/lf30_jmgekfqg.json"
         alt="loading animation" class="w-32 animate-pulse">
    <p class="text-gray-500 mt-3">Analyzing your resume...</p>
  </div>
</div>

<script>
  const form = document.getElementById('uploadForm');
  const loading = document.getElementById('loading');
  form.addEventListener('submit', () => {
    loading.classList.remove('hidden');
  });
</script>
```

Fig. 4. HTML code snippet (Upload page)

```
<form method="GET" action="/filter" class="flex gap-4 mt-6 mb-10
justify-center">
  <select name="location" class="border rounded-lg px-3 py-2">
    <option value="">All Locations</option>
    {% for loc in dropdown_locations %}
      <option value="{{ loc }}">{{ loc }}</option>
    {% endfor %}
  </select>
  <button type="submit" class="bg-blue-600 text-white px-4 py-2
  rounded-lg">Filter</button>
</form>

  <div class="grid md:grid-cols-2 lg:grid-cols-3 gap-6">
  {% for job in job_list %}
  <a href="{{ job.Link }}" target="_blank" class="no-underline">
    <div class="bg-white rounded-xl p-6 shadow-lg hover:shadow-2xl
    transition-all duration-300 border border-gray-100 hover:scale-[1.02]">
      <h3 class="text-xl font-semibold text-gray-900">{{ job.Position }}</h3>
      <p class="text-blue-600 mt-1 font-medium">{{ job.Company }}</p>
      <p class="text-gray-500 text-sm mt-2">{{ job.Location }}</p>
    </div>
  </a>
  {% endfor %}
</div>
```

Fig. 5. HTML code snippet (Result page)

– The Integration of Frontend and Backend:

We used the Flask for integrating our backend along with our frontend.

IV. RESULTS

We created a website using HTML, CSS, and Flask. We gathered the dataset on job listings, which formed the basis for the recommendation system. By using content-based filtering, the system can suggest jobs based on user skills that were extracted from the resume submitted by the user.

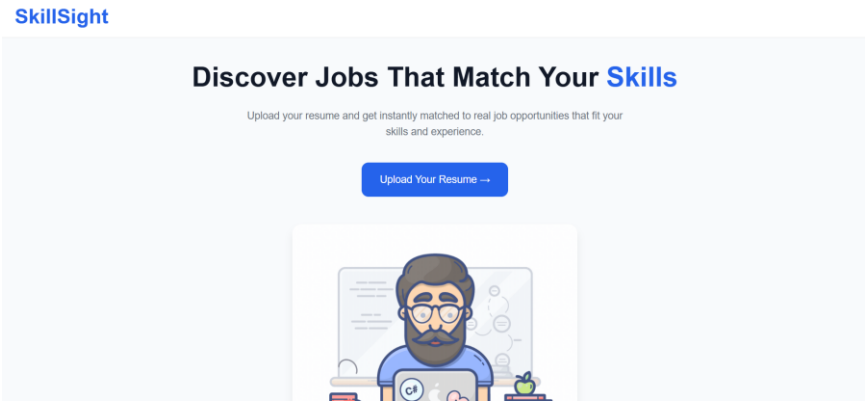

Fig. 6. Home page

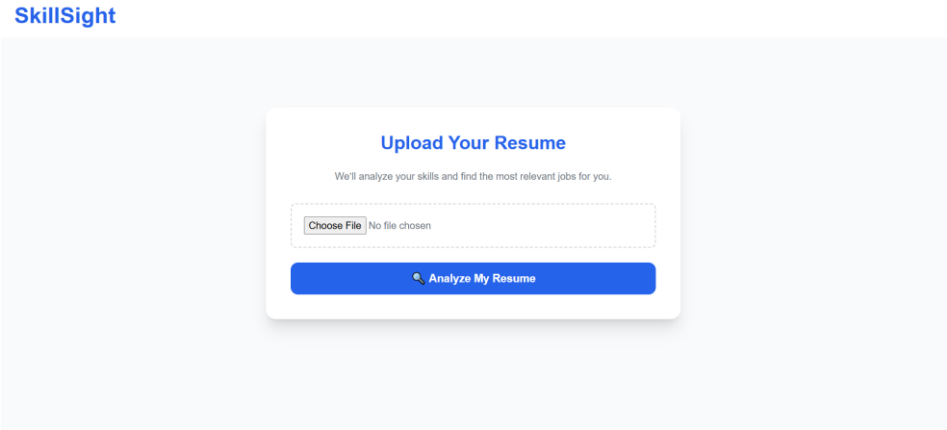

Fig. 7. Upload page

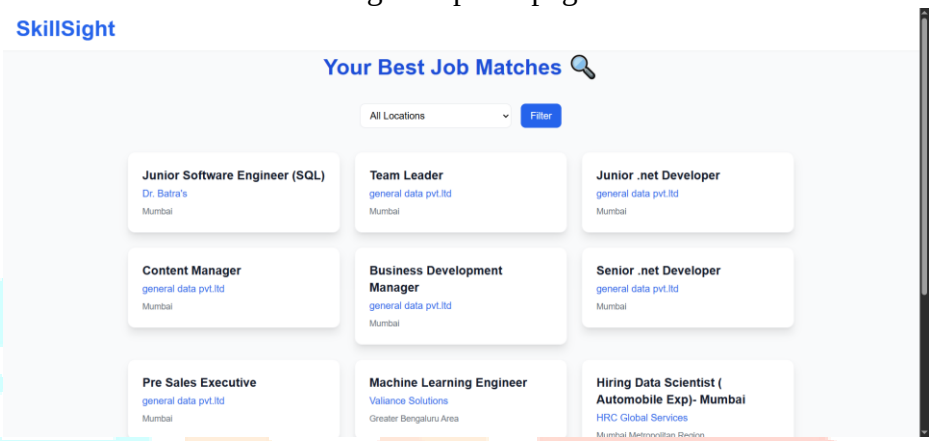

Fig. 8. Result page

In Figure 6, the launch interface of the proposed website is shown. This is where users can start interacting with the system and access the main user interface. This interface combines the features of the resume parser and the job recommendation system, as seen in Figure 7, the user is asked to upload a resume file for analysis. After the file is selected and submitted, the backend, built with the Flask framework, processes the resume. It extracts important details like skills, experience, and qualifications. The system then sends the extracted skills to the job recommendation module. Based on these skills, the system finds and shows a list of recommended job opportunities, as shown in Figure 8. The user can click on any of the listed jobs, which takes them to the respective website where the job is posted. This provides an easy and interactive experience.

V. Conclusion and Future Work

This paper presents an AI-based Resume Parser and Job Recommendation System designed to automate the extraction and analysis of candidate information. By combining Natural Language Processing (NLP) and Machine Learning (ML) techniques, the system identifies and organizes key resume attributes such as skills, education, and experience. The system uses TF-IDF vectorization and cosine similarity for efficient job-candidate matching, while the Flask-based interface provides a simple and interactive user experience. Experimental evaluation shows that the model achieves high precision and reliability, significantly cutting down manual screening efforts, and improving recruitment efficiency.

In the future, this system can be improved by integrating deep learning models like BERT or GPT for better context understanding of skills and semantic analysis. Adding support for multilingual resumes and incorporating Optical Character Recognition (OCR) would allow processing of scanned documents. Additionally, real-time integration with professional networks and job portals, such as LinkedIn or Naukri, could offer dynamic job updates and personalized recommendations. Cloud-based deployment and API integration would improve scalability and make the system more accessible to organizations of different sizes. Overall, this work lays out the groundwork for a smarter, flexible, and data-driven recruitment system.

VI. ACKNOWLEDGMENT

The authors thank T.John Institute of Technology and their guide, Amrita Sinha, Department of Information Science and Engineering, for continuous guidance and support throughout the project.

REFERENCES

[1] N. Jayakumar, A. R. Patil, S. Joshi, P. Narayanan, and S. Saoji, "An approach to parse resume and recommend job using modified K-mers and Firefly Algorithm (FFA) in resume parsing," *SJIS*, vol. 35, 2023.

[2] S. Bhor, V. Gupta, V. Nair, H. Shinde, and M. S. Kulkarni, "Resume parser using natural language processing techniques," *International Journal of Research in Engineering and Science (IJRES)*, vol. 9, pp. 1–6, 2021.

[3] I. Paparrizos, B. B. Cambazoglu, and A. Gionis, "Machine learned job recommendation," in *Proc. 5th ACM Conf. Recommender Systems*, 2011, pp. 325–328.

[4] A. Wahab S. and M. N. Nachappa, "Resume parser with natural language processing," *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, Mar. 2022.

[5] N. Patil, S. Yadav, and V. Biradar, "Resume parser and analyzer using NLP," *International Research Journal of Modernization in Engineering Technology and Science*, 2023.

[6] V. Desai, D. Bahl, S. Vibhandik, and I. Fatma, "Implementation of an automated job recommendation system based on candidate profiles," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, May 2017.

[7] S. K. Mishra and M. Reddy, "A bottom-up approach to job recommendation system," *International ACM*, Sept. 2016.

[8] B. D. Puspasari, L. L. Damayanti, A. Pramono, and A. K. Darmawan, "Implementation of K-means clustering method in job recommendation system," in *7th Int. Conf. Electrical, Electronics and Information Engineering (ICEEIE)*, Oct. 2021.

[9] H. Mhaske, A. N. Kshirsagar, S. Nevase, and A. Pimparkar, "NLP based resume parser analysis," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 10, May 2023.

[10] F. Khan, H. Patel, A. Shaikh, F. Sayed, and A. R. Soorya, "Resume parser and summarizer," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 3, Apr. 2023.

[11] P. Rasal, Y. Balwaik, M. Rayate, R. Shinde, and A. S. Shinde, "Resume parser analysis using machine learning and natural language processing," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 5, Mar. 2023.

[12] P. R. Kulkarni, Y. Vaidya, A. Shelare, N. Attarde, and M. Sali, "Resume parser using ML and NLP," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 5, Nov. 2023.