WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

An Intelligent Resume Parsing and Ranking System for Automated Candidate Shortlisting

Mr. M. Hari Krishna, B. Bharath Kumar, MD Rashad, M Bhanu Prakash, N Shiva Charan

Assistant. Professor III B Tech Student,
Department of CSE (Data Science), ACE Engineering College, Hyderabad, India

ABSTRACT

In the modern recruitment landscape, manual resume screening remains a time-consuming and error-prone task, often leading to inefficiencies in candidate shortlisting. This paper presents the design and implementation of an intelligent resume parsing system that leverages Natural Language Processing (NLP) and machine learning techniques to automate the extraction and evaluation of candidate information from unstructured resume files. The system supports various document formats, including .doc, .docx, and .pdf, and accurately extracts key data fields such as contact information, technical and soft skills, educational qualifications, and professional experience.

Keywords: Resume Parsing, Automated Recruitment, Candidate Shortlisting, Information Extraction, Resume Screening, TF-IDF, Cosine Similarity, Python Automation, Document Processing

1. INTRODUCTION

In today's competitive job market, HR departments face the challenge of screening large volumes of resumes for each job opening. Traditional manual screening is time-consuming, error-prone, and inefficient, especially when dealing with unstructured document formats.
To address this, automated resume parsing systems have emerged as effective solutions for extracting key candidate information such as contact details, skills, and experience. This paper presents a Python-based Resume Parser that supports .doc, .docx, and .pdf files.

2.EXISTING SYSTEM

Current resume screening practices in many organizations depend heavily on manual evaluation by HR personnel. Recruiters typically open each resume individually, review the contents, and extract critical information such as contact details, educational background, professional experience, and skill sets. These details are either recorded manually or entered into internal systems such as Applicant Tracking Systems (ATS). While some ATS platforms offer parsing capabilities, they often suffer from low accuracy, poor format compatibility, and require significant manual correction.

Manual Effort and Time Consumption: The process is highly time-intensive and inefficient, especially during bulk recruitment or campus placement drives.

Inconsistent Data Interpretation: Human evaluators may interpret resume content differently, leading to inconsistent shortlisting decisions.

Limited Parsing Accuracy: Many existing tools rely on keyword matching or fixed templates, making them ineffective for resumes with complex layouts or unconventional formats.

3.PROPOSED SYSTEM

The proposed Resume Parser system addresses the limitations of traditional and semi-automated resume screening methods by introducing an intelligent, scalable, and efficient solution for modern recruitment workflows. Developed using Python, the system automates the extraction and analysis of resume data, providing HR teams with structured candidate information while significantly reducing manual effort and time.

Key features of the proposed system include:

1 **Multi-Format Support:** Compatible with .doc, .docx, and .pdf files, allowing processing of resumes in commonly used formats.

2 **Automated Data Extraction:** Extracts critical fields such as name, contact information, educational background, professional experience, and technical skills.

3. **Batch Processing:** Allows multiple resumes to be processed simultaneously, improving efficiency during bulk hiring.

4 **Structured Output Reports:** Generates Excel sheets summarizing parsed data for easy comparison and analysis.

4.METHODOLOGY

The proposed Resume Parser follows a modular pipeline that includes dataset preparation, model training, resume file processing, information extraction, and candidate ranking. The system architecture is designed to ensure scalability, accuracy, and real-time performance in high-volume recruitment scenarios.

1. Data Collection

- Uses the Kaggle Resume Dataset and custom industry-specific resumes.
- Supports .pdf, .docx, and .doc formats.

2. Text Extraction & Preprocessing

- Extracts text using libraries like PyMuPDF and python-docx.
- Cleans data by removing noise, standardizing format, and normalizing content.

3. Information Extraction

- Uses spaCy and regex to extract key details such as name, contact, skills, education, and experience.
- Categorizes data into structured sections.

4. Candidate Matching

- Compares extracted content with job descriptions using TF-IDF and Cosine Similarity.
- Assigns scores based on relevance.

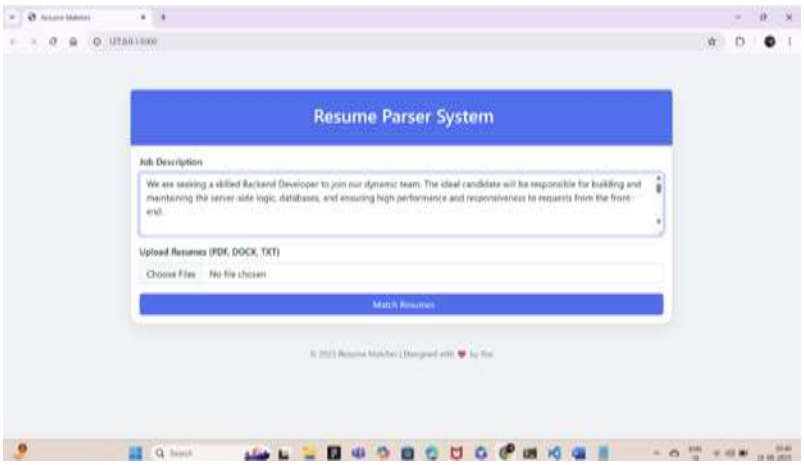
5. Reporting & Output

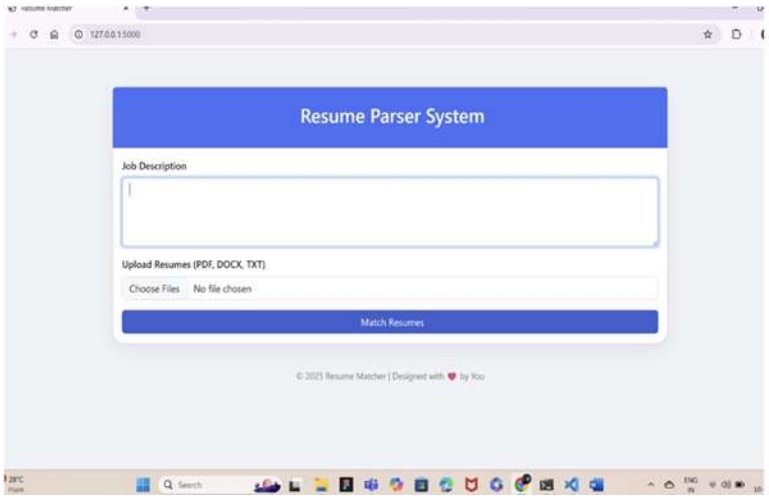
- Generates Excel reports with ranked candidate data.
- Displays results in a user-friendly interface for easy analysis by HR teams.

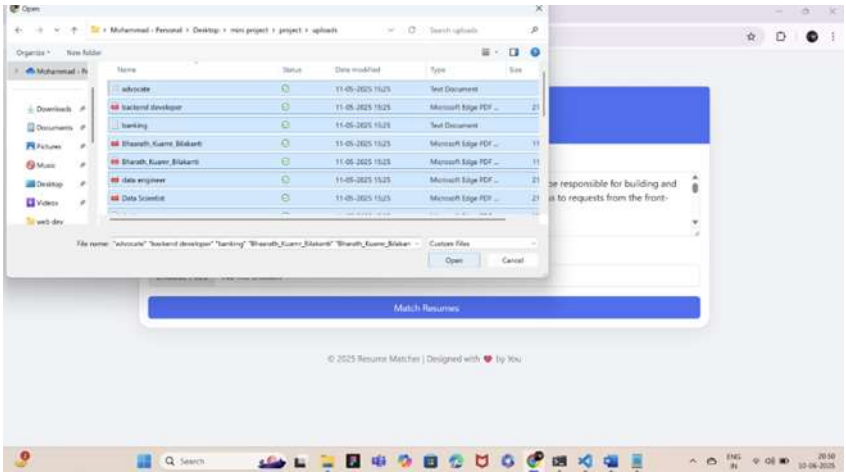
5.SYSTEM ARCHITECTURE

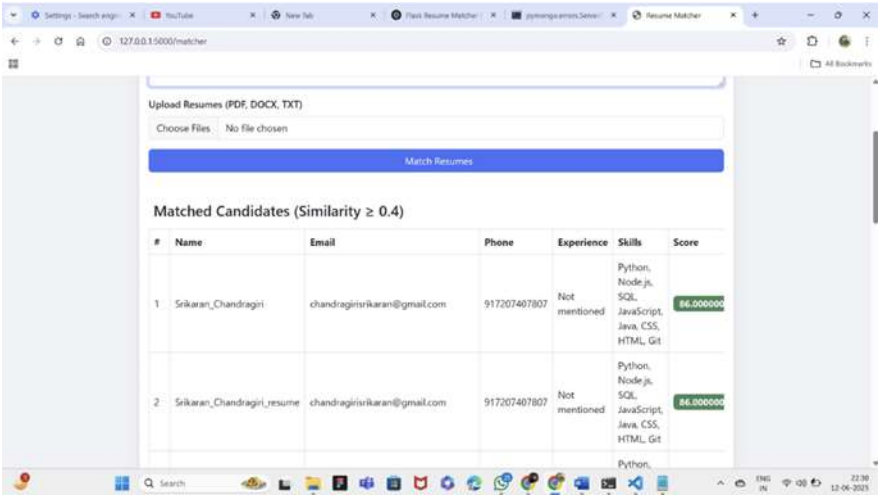
The system starts by collecting a dataset of resumes, which undergoes initial preprocessing to clean and standardize the text. Key information is extracted and converted into a structured format through feature vector generation. A classification model is then applied to analyze and score resumes based on relevance. Text normalization ensures consistency across different resume formats. Finally, the most suitable candidates are shortlisted based on their processed feature representations.

6. RESULTS AND OUTPUT









7. CONCLUSION

In conclusion, the project "Resume Parser for Job Readiness" effectively addresses key recruitment challenges by automating resume screening and ranking candidates based on their relevance to specific job descriptions. Using Python and NLP libraries like spaCy, regex, and pandas, the system extracts essential details—such as name, contact, skills, experience, and education—from various resume formats (.pdf, .docx, .doc). A key feature is its ability to compute relevance scores by comparing resumes with job descriptions, helping HR prioritize top candidates efficiently. This enhances the hiring process through greater speed, consistency, and objectivity, making the system valuable for recruiters, job portals, and placement cells**.**

8.FUTURE SCOPE

While the current system performs well in extracting data and ranking resumes.

Integration with Machine Learning Models: Implementing supervised learning algorithms can improve the accuracy of relevance ranking based on historical hiring data.

Semantic Matching Using Transformer Models: Advanced models like BERT or RoBERTa can be used to perform deeper contextual matching between resumes and job descriptions.

Web-Based Interface or API Deployment: Developing a user-friendly web application or API will allow HR teams to easily upload resumes, job descriptions, and download ranked candidate lists.

9. REFERENCES

[1] Sanket Sarwade, *Resume Parser using Python*, GitHub repository, Available: https://github.com/sanketsarwade/Resume-Parser-using-Python, Accessed: June 26, 2025.

[2] Omkar Pathak, *ResumeParser*, GitHub repository, Available: https://github.com/OmkarPathak/ResumeParser, Accessed: June 26, 2025.

[3] Kushal-022, *Resume Parser using spaCy*, GitHub repository, Available: https://github.com/kushal-022/Resume-Parser-using-Spacy, Accessed: June 26, 2025.

[4] Sayak Paul, *pyresparser – Resume Parser Using NLP Techniques*, GitHub repository, Available: https://github.com/OmkarPathak/pyresparser, Accessed: June 26, 2025.

[5] Yogesh Kumar, *AI-Resume-Parser*, GitHub repository, Available: https://github.com/yogesh-kumar-42/AI-Resume-Parser, Accessed: June 26, 2025.