Advances in Computer Engineering and Science (ACES) (2026)
Volume -1, Issues – 2

Hire Smart: Artificial Intelligence- based Resume Parser

Prabakaran G
Department of ECE
Vivekanandha College of Technology
Tiruchengode, Namakkal, Tamilnadu, India
E-Mail: prabaece345@gmail.com

NivethaSri V
Department of Information Technology
Vivekanandha College of Technology
Tiruchengode, Namakkal, Tamilnadu, India
E-Mail: nivethasri433@gmail.com

Dharshini S
Department of Information Technology
Vivekanandha College of Technology
Tiruchengode, Namakkal, Tamilnadu, India
E-Mail: dharshinisaravanan368@gmail.com

Pavithra L
Department of Information Technology
Vivekanandha College of Technology
Tiruchengode, Namakkal, Tamilnadu, India
E-Mail: pavi48852@gmail.com

Harishini S
Department of Information Technology
Vivekanandha College of Technology
Tiruchengode, Namakkal, Tamilnadu, India
E-Mail: harishiniselvakumar2004@gmail.com

Abstract: The AI-Powered Resume Parser is a web-based application designed to automate the extraction of relevant information from resumes using Natural Language Processing (NLP) and Machine Learning techniques. In traditional recruitment processes, manually screening resumes is time-consuming and inefficient. This system aims to simplify and accelerate the hiring process by automatically analysing resumes and converting unstructured data into structured formats. The application allows users to upload resumes in various formats such as PDF and DOCX. The system then processes the document, extracts key details including candidate name, contact information, skills, education, and work experience, and presents the output in a structured format like JSON or database entries. By leveraging NLP libraries, the parser improves accuracy in identifying important entities and reduces human effort. Additionally, the system can be extended to include features such as skill matching, job recommendation, and candidate ranking, making it highly useful for recruiters and organizations. This project demonstrates the integration of full stack development with AI techniques.

Keywords: Resume parsing, natural language processing, machine learning, named entity recognition, information extraction, job recommendation

I. INTRODUCTION

In today's fast-paced recruitment environment, organizations receive a large number of resumes for every job opening, making the manual screening process time-consuming and inefficient. Recruiters often spend significant time reviewing each resume to extract relevant candidate information such as skills, education, and experience. This creates a need for an automated and intelligent system that can simplify and accelerate the hiring process. The AI-Powered Resume Parser is designed to address this challenge by using Natural Language Processing (NLP) and Machine Learning techniques to automatically extract and organize information from resumes. Resumes are typically unstructured documents, which makes it difficult to process them using traditional methods. By applying advanced text processing techniques, the system converts unstructured resume data into structured and meaningful information[1].

1.1 Background of Resume Parsing

In recent years, the recruitment process has undergone significant transformation due to the rapid growth of digital technologies. Organizations now receive hundreds or even thousands of resumes for a single job opening, making manual screening a time-consuming and inefficient task. Recruiters are required to carefully review each resume to extract important details such as candidate qualifications, details, skills, and work experience, which increases workload and slows down the hiring process[2].

1.2 Limitations of Traditional Methods

Traditional resume screening methods, which mainly rely on manual review and basic keyword-based systems, have several limitations that reduce efficiency and accuracy in the recruitment process. One of the major limitations is the time-consuming nature of manual screening. Recruiters must go through a large number of resumes individually, which significantly increases the time required to shortlist candidates, especially when applications are high in volume.

1.3 Role of Machine Learning in Resume Parsing

Machine Learning (ML) plays a crucial role in enhancing the efficiency and accuracy of resume parsing systems. Unlike traditional rule-based approaches, ML enables the system to learn from data and improve its performance over time. It helps in understanding the structure and content of resumes, even when they vary in format and style. One of the key contributions of machine learning in resume parsing is automatic information extraction[3]. ML models can identify and extract important entities such as

candidate name, email, phone number, skills, education, and work experience with higher accuracy.

1.4 Challenge of Resume Data Variability

Despite the advancements in machine learning, resume parsing systems face significant challenges due to the variability in resume formats and structures. Resumes come in different layouts, styles, and file formats such as PDF, DOCX, and images, making it difficult to extract consistent information. Additionally, candidates use diverse terminologies and writing styles, which leads to ambiguity in identifying key details like skills, experience, and education. This variability often results in inaccurate data extraction and reduced system performance, especially when handling unstructured or creatively designed resumes.

1.5 Proposed Flask Based Framework

A Flask-based resume parsing system can efficiently process and analyze resumes using machine learning and natural language processing techniques. Users can upload resumes through a web interface, and the system processes the documents using trained models. The extracted information—such as candidate name, skills, experience, and qualifications—is then structured and returned in JSON format. This framework enables real-time processing, easy integration with recruitment platforms, and scalability for handling large volumes of resumes in hiring systems.

1.6 Objectives and Contributions

This study aims to design and develop an intelligent resume parsing system that automates the extraction of relevant information from resumes using machine learning and natural language processing techniques. The system is intended to simplify and accelerate the recruitment process by converting unstructured resume data into structured and meaningful insights. The key contributions of this work include the Designed and implemented a Flask-based web application for resume parsing and Integrated machine learning and NLP models for intelligent data extraction and developed a structured output format (JSON) for easy integration with recruitment systems Improved parsing efficiency compared to traditional keyword-based methods. Created a user-friendly interface for uploading and analysing resumes. Provided a scalable solution suitable for real-world hiring applications

II. LITERATURE REVIEW

Recent advancements in recruitment automation have led to the development of intelligent resume parsing systems using Natural Language Processing (NLP) and Machine Learning (ML) techniques. Several studies have explored methods to extract structured information from unstructured resumes efficiently. Early research in resume parsing focused on rule-based approaches combined with basic Natural Language Processing (NLP) techniques. Natural Language Processing (NLP) techniques were introduced to improve the extraction of structured data from unstructured resumes. NER-based systems enabled the identification of entities like names, organizations, and locations. This approach significantly improved the accuracy of extracting important candidate details

Machine learning models such as Support Vector Machines (SVM), Naive Bayes, and Decision Trees have been used to classify resumes into job categories, improving candidate filtering and selection [4]. Deep learning techniques, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models, laboratory environments[5]. Transformer-Based Models for Text Understanding Recent studies have utilized transformer-based models like BERT for better contextual understanding of resume content, leading to highly accurate information extraction [6]. Resume Matching with Job Descriptions Some systems focus on matching resumes with job descriptions using similarity measures and machine learning, helping recruiters identify the best-fit candidates efficiently [7]. Modern approaches consider document layout and structure, enabling better parsing of resumes with complex designs, tables, and multi-column formats [8]. Research has focused on handling multiple file formats such as PDF, DOCX, and images, improving system robustness in real-world applications [9]. AI-powered recruitment systems integrate resume parsing with candidate ranking and recommendation, reducing manual effort and improving hiring efficiency [10]. The literature shows a clear evolution from rulebased methods to advanced AI-driven systems. While significant improvements have been made using NLP, machine learning, and deep learning, challenges such as format variability, semantic understanding, and scalability still remain. This highlights the need for an efficient and intelligent resume parsing system.

III. METHODOLOGY

The proposed system follows a multi-stage pipeline to efficiently process and extract structured information from resumes. The methodology integrates Natural Language Processing (NLP), Machine Learning (ML), and a Flaskbased web framework to ensure accurate and scalable resume parsing [11]. The system begins with the collection and upload of resumes in various formats such as PDF, DOCX, and TXT through a web-based interface. These documents are then processed using text extraction techniques to convert them into machine-readable format and lemmatization to ensure clean and consistent data. Following this, the system performs section segmentation to identify different parts of the resume such as personal details, education, skills, and work experience.

Next, advanced NLP techniques such as Named Entity Recognition (NER) and pattern matching are applied to extract relevant entities like names, contact information, skills, and qualifications. Machine learning models are further integrated to improve the accuracy of classification and extraction by understanding contextual relationships within the text.

3.1 Data Collection

The data collection phase plays a crucial role in developing an efficient resume parsing system. This dataset typically includes:

- Resumes in multiple formats (PDF, DOCX, TXT)
- Candidate personal information (name, email, phone number)
- Educational details (degrees, institutions, year of passing)
- Work experience (job roles, companies, duration)
- Skills and technical expertise
- Projects and certifications
- Different resume layouts and writing styles
- The dataset contains two categories such as structured resumes (well-formatted and clearly organized) and unstructured resumes (varying formats, layouts, and styles).

3.2 Data Preprocessing

The data preprocessing phase is essential for cleaning and preparing resume data for accurate information extraction.

- Converting resumes from PDF and DOCX formats into plain text
- Removing unnecessary symbols, special characters, and formatting
- Lowercasing and normalization of text
- Tokenization (splitting text into words and sentences)
- Stop-word removal (eliminating common irrelevant words)
- Lemmatization/Stemming (reducing words to their root form)
- Removing duplicate or irrelevant information.

3.3 Cross-compilers to other languages

There are several compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language:

- TypeScript → JavaScript: TypeScript code is compiled into JavaScript to run in browsers.
- Java → Bytecode (JVM) : Java programs are compiled into bytecode, which runs on any system with a Java Virtual Machine (JVM).
- C/C++ → Machine Code for Different OS Code written in C/C++ can be compiled for Windows, Linux, or embedded systems.
- Python → C (via tools like Cython) Python code can be converted into C for performance improvement.
- Kotlin → JavaScript / Native: Kotlin supports cross-compilation to multiple platforms.

3.4 Feature Selection

The feature selection phase plays a crucial role in identifying the most relevant and meaningful information from resumes to enhance the performance of the resume parsing system. In this stage, important features such as candidate name, email address, phone number, skills, educational qualifications, and work experience are selected for further processing. Natural Language Processing (NLP) techniques are used to identify and extract these key entities from unstructured text. Additionally, domain-specific keywords and technical skills are recognized to improve the accuracy of candidate profiling. Irrelevant or redundant information is removed to reduce noise and improve system efficiency. By focusing only on significant features, the system achieves better accuracy, reduces computational complexity, and enables effective analysis and filtering of candidates for recruitment purposes.

3.5 Model Environment

The model environment defines the software and tools used for developing and executing the resume parsing system. The proposed system is implemented using Python due to its simplicity and strong support for Natural Language Processing and Machine Learning libraries. Libraries such as NLTK and spaCy are used for text processing and entity extraction, while additional libraries support data handling and preprocessing [12]. The system is designed to run on multiple platforms, ensuring flexibility and ease of deployment. This environment provides efficient processing, scalability, and seamless integration of various components required for resume parsing.

3.6 Database design

The database design is an essential component of the resume parsing system, as it is responsible for storing and managing the extracted information in a structured manner. In this project, the parsed data from resumes is organized into a structured format such as JSON and can be stored in a database for efficient retrieval and analysis. The database is designed to store key details including candidate name, contact information, skills, educational qualifications, and work experience. A simple and scalable database structure is maintained to ensure easy access and management of data. Each resume is treated as a separate record, with fields corresponding to different attributes of the candidate. Figures 1 and 2 shows the architecture diagram and overview of research work, respectively.

3.7 Performance Evaluation

To evaluate the effectiveness of the model, several metrics are used:

- Accuracy: Measures correctness of extracted data
- Precision: Measures relevance of extracted data.
- Recall (Sensitivity): Measures completeness of extraction
- F1-Score: Balances precision and recall
- Detection Speed: Time taken to identify and respond to threats
- Processing Time: Measures time taken to parse resumes.
- Format Handling Efficiency and Checks performance across different formats.

3.8 System Implementation

The system implementation of the resume parser is carried out using Python along with various supporting libraries and tools. The process begins with the user uploading a resume through the Streamlit-based web interface. The uploaded file is then processed using libraries such as PyPDF2 for PDF files and python-docx for DOCX files to extract the textual content. The extracted text

undergoes preprocessing to remove unnecessary characters and standardize the data.

Next, Natural Language Processing techniques are applied using spaCy to identify and extract important entities such as names, skills, education, and work experience. Additionally, regular expressions (Regex) are used to extract specific patterns like email addresses and phone numbers. The extracted information is then organized into a structured format such as JSON for easy storage and analysis. Finally, the processed results are displayed to the user through the Streamlit interface, providing a simple and interactive way to view the parsed resume data.

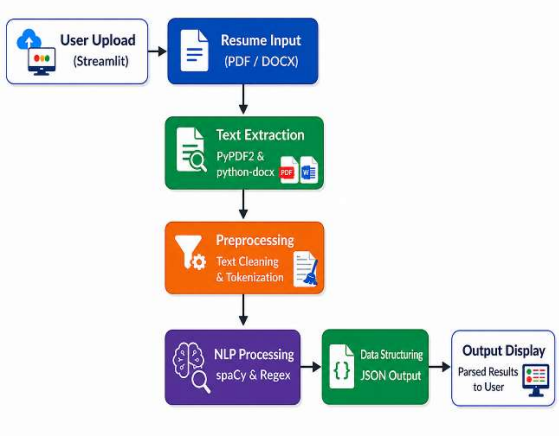

Figure 1 : Architecture diagram

NATURAL LANGUAGE PROCESSING

In this study, Natural Language Processing techniques and rule-based methods are implemented to extract and structure information from resumes. These techniques are chosen based on their efficiency, accuracy, and ability to handle unstructured textual data from various resume formats.

4.1 NLP-Based System

Natural Language Processing (NLP) plays a vital role in the resume parsing system by enabling the extraction and understanding of unstructured textual data. It helps in identifying important information such as candidate name, skills, education, and work experience from resumes with different formats and styles.

The system uses pre-trained NLP models to recognize entities and patterns within the text, improving the accuracy of information extraction. By analyzing the context and structure of the resume, NLP techniques convert raw data into structured and meaningful information. This approach enhances the efficiency and reliability of the system, making it suitable for real-world recruitment applications where large volumes of resumes need to be processed quickly and accurately.

4.2 Techniques Used in NLP

The resume parsing system utilizes various Natural Language Processing techniques to extract meaningful information from unstructured resume data. Tokenization is used to break the text into smaller units such as words and sentences for easier processing. Named Entity Recognition (NER) is applied to identify important entities like candidate names, skills, organizations, and locations. Text preprocessing techniques such as cleaning, normalization, and stop-word removal are used to improve data quality and consistency.

Additionally, regular expressions (Regex) are used to extract specific patterns such as email addresses, phone numbers, and dates. These techniques work together to enhance the accuracy and efficiency of the resume parsing system [13].

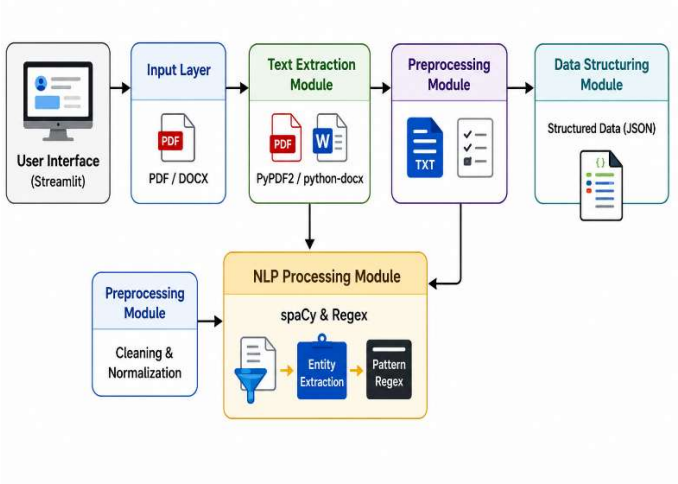

Figure 2 : Overview of proposed research work

4.3 Why NLP is Used

Natural Language Processing is used in this system because resumes contain unstructured textual data that cannot be easily processed using traditional methods. NLP enables the system to understand, analyse, and extract relevant information such as skills, education, and work experience from resumes with different formats and writing styles. It improves the accuracy of data extraction and reduces the need for manual screening. Additionally, NLP techniques provide scalability and flexibility, allowing the system to handle large volumes of resumes efficiently. This makes NLP an essential component for building an effective and intelligent resume parsing system

4.4 ROLE OF NLP

- Extracts important information from unstructured resume data
- Identifies key entities like name, skills, and experience
- Understands context and meaning of text
- Handles different resume formats and writing styles
- Reduces manual effort in resume screening
- Converts raw data into structured format
- Improves accuracy and efficiency of the system

IV. IMPLEMENTATION

The resume parsing system is implemented using Python along with various supporting libraries for efficient processing and extraction of information. The system provides a user-friendly interface through Streamlit, allowing users to upload resumes in formats such as PDF and DOCX. Once the resume is uploaded, text extraction is performed using PyPDF2 for PDF files and python-docx for DOCX files. The extracted text is then passed through a preprocessing stage where unnecessary characters, symbols, and formatting are removed to ensure clean and consistent data.

Natural Language Processing techniques are applied using spaCy to identify and extract important entities such as candidate name, skills, education, and work experience. Additionally, regular expressions (Regex) are used to extract specific patterns like email addresses and phone numbers. The extracted information is then structured into a JSON format, making it easy to store and analyze. Finally, the processed data is displayed to the user through the Streamlit interface, providing a simple and interactive way to view the parsed resume details.

The proposed resume parsing system is implemented using Python, leveraging its extensive support for Natural Language Processing and text processing libraries. The system is designed to provide an efficient and user-friendly interface for extracting structured information from unstructured resume data. The implementation begins with the development of a webbased interface using Streamlit, which allows users to upload resumes in commonly used formats such as PDF and DOCX. The system ensures proper file handling and validation before processing the uploaded documents.

Once the resume is uploaded, text extraction is performed using PyPDF2 for PDF files and python-docx for DOCX files. These libraries convert the document content into plain text, which serves as the input for further processing. The extracted text is then passed through a preprocessing stage, where it undergoes cleaning, normalization, tokenization, and removal of unnecessary symbols and stop words to improve data quality. The system is designed to handle multiple resumes efficiently and can be extended to support additional features such as candidate ranking and job matching.

V. RESULTS AND ANALYSIS

The resume parsing system effectively extracts important information such as name, contact details, skills, education, and work experience from resumes. It supports multiple file formats including PDF and DOCX. The system converts unstructured resume data into a structured format for easy analysis. NLP techniques improve the accuracy of information extraction. The results are displayed through a user-friendly interface. The system reduces manual effort in resume screening. Overall, it provides fast and reliable performance for recruitment applications. Figures 3 and 4 shows the results of proposed method.

Figure 3 : Summary of parsed resume

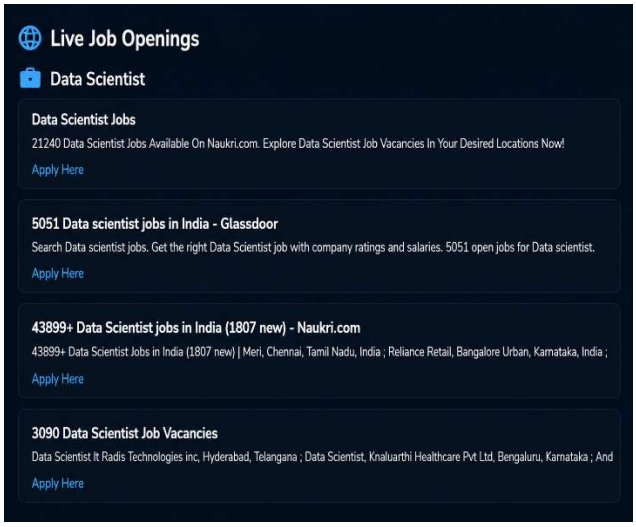

Figure 4 : Summary of final output

VI. CONCLUSION

The proposed AI-based resume parsing system successfully demonstrates an efficient approach to extracting structured information from unstructured resume data. By utilizing Natural Language Processing techniques along with rulebased methods, the system is able to identify and extract key details such as candidate name, contact information, skills, education, and work experience with good accuracy. The system supports multiple file formats including PDF and DOCX, making it flexible and suitable for real-world applications. The integration of tools such as spaCy, Regex, PyPDF2, and python-docx ensures effective text extraction and processing. Additionally, the use of a Streamlit-based interface provides a user-friendly platform for uploading resumes and viewing results. The implementation reduces manual effort

involved in resume screening and significantly improves the speed and efficiency of the recruitment process. It also ensures consistency in data extraction, which is essential for largescale hiring systems.

VII. FUTURE WORK

The proposed resume parsing system successfully extracts and processes important information such as candidate name, contact details, skills, educational qualifications, and work experience from resumes. The system supports multiple file formats including PDF and DOCX, ensuring flexibility and usability in real-world scenarios. It efficiently converts unstructured resume data into a structured format, making it easier for storage, analysis, and integration with recruitment systems. The implementation of Natural Language Processing techniques significantly improves the accuracy of information extraction by identifying relevant entities and understanding the context of the text.

Additionally, the use of regular expressions enhances the extraction of specific patterns such as email addresses, phone numbers, and dates. The system performs well across different resume formats and writing styles, demonstrating its robustness and adaptability. The Streamlit-based interface provides a simple and interactive platform for users to upload resumes and view the extracted results instantly. Furthermore, the system reduces manual effort involved in resume screening and increases the speed of the recruitment process. It is capable of handling multiple resumes efficiently, making it suitable for bulk processing. The overall performance of the system is reliable, with good accuracy and fast processing time, making it effective for real-world recruitment applications.

REFERENCES

[1] D. Abisha, S. Keerthana, R. Navedha Evanjalin, K. Kavitha, S. Jothi Mary, and R. Ramya, "Resspar: AI-driven resume parsing and recruitment system using NLP and generative AI," in *2024 Second international conference on intelligent cyber physical systems and Internet of Things (ICoICI)*, 2024: IEEE, pp. 1-6.

[2] S. Dhanalakshmi and S. Sathiyabama, "Reviews on swarm intelligence algorithms for text document clustering," *vectors,* vol. 1, p. 2, 2022.

[3] S. Dhanalakshmi, S. Sathiyabama, and D. Ayyamuthukumar, "An Efficient Document Clustering Method Using Improved Bacterial Colony Optimization," *International Journal of Advanced Networking and Applications,* vol. 17, no. 4, pp. 7062-7075, 2026.

[4] D. Shareef, A. S. Sulthana, J. Rajeswari, G. Vasavi, and G. Vasavi, "A Unified AI-Powered Resume Generation and ML-Based Candidate Filtering Framework for Smart Campus Recruitment," in *2026 7th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, 2026: IEEE, pp. 1085-1092.

[5] P. Khurana and J. Singla, "Artificial Intelligence in Recruitment: Frameworks and Innovations," in *2026 4th International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, 2026: IEEE, pp. 1598-1603.

[6] A. P. Bhopale and A. Tiwari, "Transformer based contextual text representation framework for intelligent information retrieval," *Expert Systems with Applications,* vol. 238, p. 121629, 2024.

[7] S. Guo, F. Alamudun, and T. Hammond, "RésuMatcher: A personalized résumé-job matching system," *Expert Systems with Applications,* vol. 60, pp. 169-182, 2016.

[8] F. Han *et al.*, "Latex2layout: High-fidelity, scalable document layout annotation pipeline for layout detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026, vol. 40, no. 37, pp. 30907-30915.

[9] R. Pawar, S. Karpe, K. Jain, A. Shelke, and P. Tank, "Implementation of an Automated Semantic Job-Matching and Resume Screening Platform," *Available at SSRN 6665899,* 2026.

[10] K. Punnya and T. Ganga, "AI BASED RESUME SCREENING AND CANDIDATE RANKING SYSTEM," in *International Conference on Sustainable Computing & Intelligent Systems*, 2026: Shanlax Publications, p. 96.

[11] A. K. Sinha, M. Amir Khusru Akhtar, and A. Kumar, "Resume screening using natural language processing and machine learning: A systematic review," *Machine Learning and Information Processing: Proceedings of ICMLIP 2020,* pp. 207-214, 2021.

[12] A. Sinha, M. Akhtar, and M. Kumar, "Automated resume parsing and job domain prediction using machine learning," *Indian Journal of Science and Technology,* vol. 16, no. 26, pp. 1967-1974, 2023.

[13] K. W. Chong, K. W. Ng, and Y. H. Fu, "Resume data extract and job recruitment Chatbot features for AI-based resume screening & analytics," *MethodsX,* vol. 16, p. 103775, 2026.