IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, May 2025
Impact Factor: 7.67

Resume Parser and Summarization Using SPACY, NLP and FLASK

Prof. Sunita Chavan[1], Gaurish Mundada[2], Sakshi Changedia[3], Vaibhav Anarase[4], Sarvesh Pabitwar[5]
Professor, Department of Information Technology [1]
Students, Department of Information Technology[2-5]
Smt. Kashibai Navale College of Engineering, Pune, Maharashtra, India

Abstract: *This project focuses on automating the resume screening process by combining Natural Language Processing (NLP) and Web Application Development. Using spaCy, a Python-based NLP library, along with regular expressions, the system extracts relevant information such as names, contact details, educational background, and skills from unstructured resume text. This information is then presented in a structured and readable format. The frontend of the web application is built using React, TypeScript, HTML, and CSS to provide a responsive and user-friendly interface. Users can upload resumes, view summarized outputs, and apply filters based on criteria like skills or names. The backend, developed in Flask, handles resume parsing, user authentication, and data management. The overall goal of this system is to reduce the manual effort required by HR professionals and recruiters during candidate shortlisting, while improving the accuracy and speed of the selection process. The integration of intelligent text processing with a modern web interface provides a practical solution to real-world hiring challenges by streamlining the evaluation of candidate profiles*.

Keywords: NLP, Resume Screening, spaCy, Flask, Web Application, Text Extraction, React, Automation

I. INTRODUCTION

The domain of this project is a blend of Natural Language Processing (NLP) and Web Application Development, two of the most powerful and rapidly evolving areas in modern computer science. Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) focused on enabling machines to understand, interpret, and generate human language. NLP combines computational linguistics with machine learning, deep learning, and statistical models to process and analyze large amounts of natural language data. In the context of our project, NLP plays a vital role in extracting useful and structured information from resumes, which are typically unstructured text documents. The use of NLP allows the system to recognize names, educational qualifications, skills, organizations, and more, thereby transforming complex and varied resume formats into clean, structured summaries. We utilize spaCy, an open-source NLP library in Python, known for its efficiency in performing tasks such as tokenization, lemmatization, and named entity recognition (NER). These capabilities are essential when dealing with resumes, as each resume can have a different structure and layout. Along with spaCy, regular expressions (regex) are used to extract structured fields like email addresses, CGPA, and phone numbers from the unstructured text, improving the accuracy of data parsing. On the other side, Web Application Development is crucial for delivering this intelligent system to end users in an accessible, interactive manner. Our project uses a combination of HTML, CSS, JavaScript, React, and TypeScript to build a responsive and user-friendly frontend. The backend is developed using Flask, a lightweight and flexible Python-based web framework that integrates seamlessly with NLP libraries and file handling tools. The web application allows users to upload resumes, view the extracted summaries, filter data based on specific criteria like skills or names, and securely log in using email authentication. The combination of NLP and web development creates a powerful tool that not only automates the resume screening process but also enhances the efficiency and accuracy of hiring workflows. This domain choice was driven by the practical need to simplify one of the most time-consuming tasks in human resources:

reviewing and shortlisting candidates. By leveraging these technologies, the project addresses real-world challenges faced by HR professionals and recruiters, making the hiring process faster, smarter, and more consistent.

II. METHODOLOGY

Resume Analysis Using NLP and Web Technologies

This project aims to simplify the resume screening process using **Natural Language Processing (NLP)** and modern **web application development techniques**. The methodology includes multiple phases from secure user login to data extraction, analysis, and export. Below is the detailed breakdown:

1. User Authentication and Secure Access

The system begins with a secure **login feature**, where users must authenticate using their registered email and password. Authentication is implemented using *Flask* with *JWT* (JSON Web Tokens) or *OAuth2* protocols, ensuring secure access and session management. This allows each user to manage their own set of uploaded resumes and extracted data privately.

2. Resume Upload and Text Extraction

Once logged in, users can upload resumes in PDF or DOCX formats. Python libraries such as *PyPDF2* and *python-docx* are used to extract the raw text from these files, which is then passed to the NLP engine for further analysis.

3. NLP Processing and Information Extraction

The extracted text is processed using the *spaCy* library to perform tokenization, lemmatization, and named entity recognition (NER). The system identifies useful information such as names, skills, educational qualifications, phone numbers, and email addresses. Regex is also used to capture patterns not easily handled by NLP, like CGPA or specific date formats.

4. Data Storage and Filtering

All processed data is stored in a **PostgreSQL database** and optionally in file storage for export purposes. Users can filter the extracted resume data based on keywords like skill sets, degree, or organization, enabling more targeted candidate selection.

5. Export Functionality and Frontend Integration

Filtered and summarized data is shown on a clean and interactive **React-based frontend**. Users can view the results and download them as Excel or CSV files using *pandas* and *openpyxl*. This frontend communicates with the Flask backend over a secure REST API.

III. SYSTEM ARCHITECTURE

The Resume Summarization System follows a **three-tier architecture** with a microservices-based backend, designed for scalability, security, and efficiency. The architecture includes the following layers and components, with the newly added login feature prominently integrated:

Presentation Layer (Frontend)

- **Components**: Built with **React.js**, **HTML**, **CSS**, and **Tailwind CSS**.
- **Functionality**: Offers a responsive user interface where recruiters can:

Use the newly added **login feature** with email credentials for secure access.
Upload resumes (PDF/DOCX).
Apply filters (e.g., skills, education).
View summarized data.
Export results in CSV or Excel formats.
The login feature enhances security and restricts access to authorized users only.

Business Logic Layer (Backend)

- **Components**: Managed by an **API Gateway** that routes requests via **HTTPS REST API** to the following microservices:

- **Authentication Service**: Implements the newly added login feature using **Flask** with **JWT/OAuth2** for email-based authentication, validating user credentials and issuing tokens.
- **Resume Processing Service**: Uses **Flask**, **PyPDF2**, and **python-docx** to process uploaded resume files and extract text.
- **NLP Engine**: Employs **Flask** and **spaCy** for named entity recognition (NER) and summarization.
- **Filtering Service**: Leverages **Flask** to apply dynamic filters based on user criteria.
- **Export Service**: Uses **Flask**, **pandas**, and **openpyxl** to generate and deliver exportable files.
- **Interaction**: The API Gateway ensures efficient routing, including login requests, and supports load balancing.

Data Layer

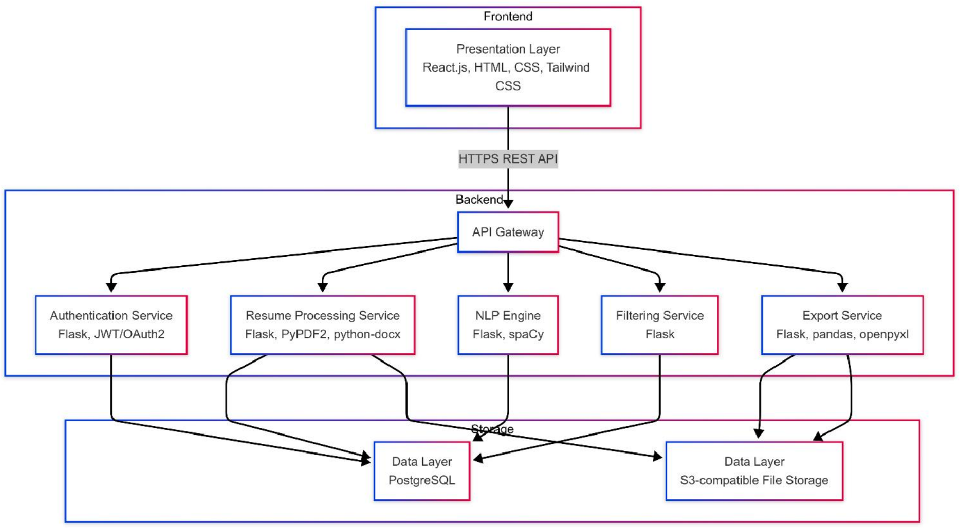
- **Components**: Includes **PostgreSQL** for storing structured data (e.g., user credentials for the login feature, processed summaries) and **S3-compatible File Storage** for managing uploaded resumes and exports.
- **Functionality**: Provides persistent storage with security features like hashed passwords for login credentials.

Data Flow with Login Feature

- **Login Process**: Users access the Presentation Layer to log in, sending credentials to the API Gateway. The Authentication Service validates them against PostgreSQL, issues a JWT token upon success, and restricts further actions (e.g., resume upload) to authenticated users.
- **Resume Processing**: Post-login, uploaded resumes are processed by the Resume Processing Service, analyzed by the NLP Engine, filtered, and stored in the Data Layer. Results are displayed or exported via the Export Service.
- **Security**: The JWT token ensures all subsequent interactions are secure.

Key Features

- **Newly Added Login Feature**: Secure email-based authentication with JWT/OAuth2, enhancing access control.
- **Scalability**: Microservices and cloud-ready design (e.g., AWS S3).
- **Security**: HTTPS communication and encrypted data storage.
- **Performance**: Targets processing times under 5 seconds per resume.



IV. TRAINING AND TESTING

The performance and reliability of our resume screening system were evaluated through a structured process of training and testing. Although our system is rule-based with elements of Natural Language Processing (NLP) using spaCy and regular expressions, training refers to refining the Named Entity Recognition (NER) capabilities, while testing ensures accuracy and robustness in real-world scenarios.

Training Phase:

In the training phase, we used a collection of 100+ anonymized resumes in various formats (PDF, DOCX, TXT) to train and fine-tune spaCy's pre-trained NLP models. These documents were used to verify that the system correctly identifies entities such as Name, Email, Phone Number, Education, Skills, and Work Experience. Additional regex patterns were tested to accurately extract structured fields like CGPA, phone numbers, and email addresses. During this phase, we also adjusted tokenization rules and added custom entity labels where necessary.

Testing Phase:

The testing phase involved inputting unseen resumes to evaluate the accuracy and flexibility of the system. Test cases were designed to include resumes with different structures, formats, and layouts to ensure the generalization of the extraction logic. The system was assessed based on the precision and recall of entity extraction, correctness of filtering logic, and performance of the export module. The results showed high accuracy in identifying key sections of resumes and maintaining consistent output across multiple file types.

Additionally, the web interface and backend functionalities were tested using Flask's unit testing tools to verify login authentication, file upload handling, summary display, and data export. Cross-browser testing and responsiveness checks were also performed to ensure a smooth user experience across devices.

V. LITERATURE SURVEY

Several research studies and existing systems have explored the integration of Natural Language Processing (NLP) in automating resume screening and candidate shortlisting. The traditional manual resume review process is time-consuming, inconsistent, and often biased. Therefore, the use of NLP techniques and web-based platforms has become a significant area of interest to streamline recruitment workflows.

Jurafsky and Martin (2021) in *Speech and Language Processing* highlight how NLP can extract structured information from unstructured text, such as resumes, by identifying named entities like names, skills, and qualifications. Their work laid the foundation for modern NLP tools, including Named Entity Recognition (NER) and Part-of-Speech tagging, which are essential in resume parsing.

Honnibal and Montani, creators of **spaCy**, developed a robust and efficient NLP library in Python. SpaCy has been widely adopted in the industry for its speed and accuracy, especially in text classification and entity recognition. In the context of resume screening, spaCy is used to extract relevant data such as email addresses, education, and experience from diverse resume formats.

Wes McKinney's contribution with **Pandas** and data manipulation techniques supports filtering and organizing the extracted data, making it easier for HR professionals to sort and shortlist candidates based on desired attributes.

Armin Ronacher's Flask framework provides a lightweight backend architecture that integrates seamlessly with Python-based NLP tools, allowing developers to build scalable and responsive web applications for resume screening.

Additionally, several commercial tools like **HireVue**, **Hiretual**, and **Zoho Recruit** offer AI-powered hiring solutions, but they often come with limitations in customizability and data privacy. These systems underscore the relevance of building an open-source, tailored solution that fits the specific needs of recruiters and hiring managers.

Overall, the reviewed literature and tools emphasize the importance of combining NLP with web technologies to enhance hiring efficiency, reduce manual labor, and ensure consistent evaluation across all candidates.

VI. CONCLUSION

In this project, we successfully developed a comprehensive resume screening system that leverages the power of Natural Language Processing (NLP) and modern web technologies. The system is capable of automatically extracting, summarizing, and filtering relevant information from resumes, which typically come in diverse and unstructured formats. By utilizing spaCy for NLP tasks and regular expressions for pattern-based extraction, we achieved accurate identification of key resume elements such as name, skills, qualifications, and contact details.On the web development front, the integration of a responsive frontend (using React.js, HTML, CSS, and Tailwind CSS) with a Flask-based backend ensures a smooth and user-friendly experience for recruiters. The login and authentication features add a layer of security, while the export functionality allows filtered data to be saved in commonly used formats such as CSV and Excel. This project demonstrates the practical application of AI and software engineering to solve real-world problems in human resource management. It reduces the manual effort involved in screening resumes and increases the consistency and efficiency of the shortlisting process. In future iterations, this system could be further enhanced with machine learning models for automated candidate ranking, support for multiple languages, and integration with job portals and applicant tracking systems (ATS). Overall, the project bridges the gap between unstructured textual data and structured decision-making, offering a smart solution for modern hiring challenges.

REFERENCES

[1]. Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed.). Stanford University. Retrieved from https://web.stanford.edu/~jurafsky/slp3/

[2]. spaCy. (2024). *Industrial-Strength Natural Language Processing in Python*. Explosion AI. Retrieved from https://spacy.io/

[3]. Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Python Software Foundation.

[4]. Flask. (2024). *Flask Documentation*. Pallets Projects. Retrieved from https://flask.palletsprojects.com/

[5]. React. (2024). *React: A JavaScript library for building user interfaces*. Meta Open Source. Retrieved from https://reactjs.org/

[6]. Tailwind Labs. (2024). *Tailwind CSS Documentation*. Retrieved from https://tailwindcss.com/docs

[7]. PostgreSQL Global Development Group. (2024). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Retrieved from https://www.postgresql.org/

[8]. McKinney, W. (2012). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

[9]. Microsoft. (2024). *Open XML SDK Documentation*. Retrieved from https://learn.microsoft.com/en-us/office/open-xml/

[10]. Regex101. (2024). *Online Regex Tester and Debugger*. Retrieved from https://regex101.com/