AI-RESUME-ANALYZER

1. Executive Summary

In modern recruitment, manually screening hundreds of resumes for a single job opening is inefficient, time-consuming, and prone to human bias. This process creates a significant bottleneck for hiring managers and human resources (HR) departments. The AI-Powered Resume Ranker project directly addresses this challenge by providing an intelligent web application to automate and optimize the initial screening phase.

This report outlines the design and implementation of this system. By leveraging Natural Language Processing (NLP) techniques, the application analyzes, scores, and ranks a batch of candidate resumes against a specific job description. The system quantifies the relevance of each resume, enabling HR professionals to instantly identify the most qualified candidates. The complete pipeline, from PDF text extraction to deployment as an interactive Flask web application, was successfully implemented.

2. Project Objective

The primary objective of this project was to build an end-to-end web tool to automatically rank job candidates based on their resumes. The specific goals were to:

1. Extract plain text from multiple uploaded PDF resumes.
2. Preprocess the text from both the job description and resumes using SpaCy to clean and normalize the data.
3. Vectorize the cleaned text using the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm.
4. Implement a scoring algorithm (Cosine Similarity) to quantify the match between each resume and the job description.
5. Develop a Flask web application with a simple UI to upload resumes and display the final ranked list.
6. Add a feature to download the ranked report as a CSV file for HR records.

3. Methodology & Technology Stack

The project was built on a robust NLP pipeline, supported by a stack of powerful and standard Python libraries. The tools below were selected for their specific roles in the project's workflow.

Core Technologies:

- **Python 3:** The core programming language for the entire project.
- **Flask:** A lightweight web framework used to build the web UI and serve the model.
- **SpaCy:** Used for advanced NLP preprocessing (lemmatization, stop-word removal).
- **Scikit-learn (Sklearn):** For TF-IDF vectorization and calculating Cosine Similarity scores.
- **pdfplumber:** A dedicated library to accurately extract text from PDF files.
- **Pandas:** To structure the final ranked data and generate the CSV report.
- **HTML/CSS:** Used to create the simple, functional frontend for the web application.

AI-RESUME-ANALYZER

Step-by-Step Workflow:

1. PDF Text Extraction

A function using pdfplumber opens each uploaded PDF file, iterates through its pages, and extracts all text content into a single string for each resume.

2. NLP Preprocessing (SpaCy)

Raw text is "noisy" and unsuitable for direct analysis. Both the job description and all resume texts are passed through a preprocessing pipeline using SpaCy: Lowercasing, Tokenization, Stop Word & Punctuation Removal, and Lemmatization (e.g., "running" -> "run").

3. Vectorization (TF-IDF)

The cleaned text strings are converted into a numerical format using TfidfVectorizer from Scikit-learn. This model gives higher weight to keywords that are important to the job description but rare across all other documents.

4. Scoring Algorithm (Cosine Similarity)

To "rank candidates," the TF-IDF vectors for the job description and each resume are compared using Cosine Similarity. This produces a match score between 0.0 (no match) and 1.0 (perfect match).

4. System Architecture & Deployment

The entire NLP pipeline was wrapped in a user-friendly web application, making the tool accessible to non-technical users.

- **Backend (Flask):** The Flask application (app.py) serves as the system's core. It handles web routes, file uploads, orchestrates the NLP workflow, and generates the final report.
- **User Interface (UI):** A single index.html template file provides a clean form with a <textarea> for the job description and a file input for multiple PDFs. Results are displayed in a clean HTML table.

5. Deliverables

The project successfully delivered all planned components:

- **Flask App:** A functional app.py file containing all backend logic and routes.
- **Scoring Algorithm:** Python functions for extract_text_from_pdf, preprocess_text, and rank_resumes.
- **UI:** A clean, single-page index.html template for the full user experience.
- **Sample Outputs:** The application generates a resume_ranking.csv file and an on-screen results table.

6. Conclusion

This project successfully demonstrated the power of NLP for automating a critical HR function. The AI-Powered Resume Ranker provides an objective, data-driven, and incredibly fast alternative to manual resume screening, empowering recruiters to focus their time and effort on the most promising candidates. It serves as a robust foundation for future enhancements, such as entity recognition or integration with an applicant tracking system (ATS).