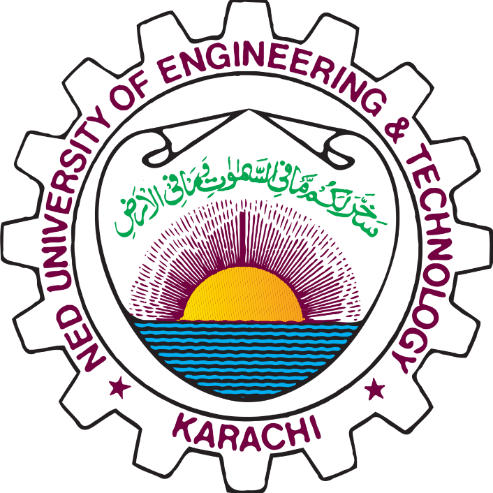
**NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY**





**Artificial Intelligence (CT-361)**

**AI Resume Parser**

**Group ID: AI-14**

**Submit To: Miss Navera Sami**

**Group Members:**

|  |  |
| --- | --- |
| **Zainab Iman Khan** | **SE-21061** |
| **Mohammad Talal Zafar** | **SE-21076** |
| **Muhammad Anas Khan** | **SE-21079** |

# **Resume Parser**

## **Abstract**

## The main focus of this project is to build an intelligent resume parsing and classification model for recruitment that streamlines the initial screening process. The system uses the SpaCy NLP model "en\_core\_web\_sm", regular expressions and keyword searching to extract key entities like names, skills, experience, and education from resumes. By leveraging SpaCy's Named Entity Recognition (NER) capabilities, regex, and keyword search, the system identifies and organizes essential resume information. The TF-IDF vectorization technique and K-Means clustering are then used to classify resumes into job roles such as frontend developer, backend developer, and Python developer. This automation aids hiring managers by reducing manual workload in the initial stages of recruitment. The project is based on the assumption that automation can increase the accuracy and speed of screening, offering a significant efficiency boost. The model assigns a percentage match score for each resume based on job compatibility, helping recruiters prioritize candidates.

## **Platform**

**Platform: Kaggle**

For this project, Kaggle was chosen as the platform because of the solid backing it gives to frameworks used in AI as well as machine learning that require sizable computational power. As of now, Kaggle offers pretrained large-scale models which don’t come with disk access but with GPU, which is necessary to train and run models like LLMs. Cloud based architecture of the platform enables running of the model in real time and frequent testing is always beneficial especially for fine tuning the resume parsing.

**Technical Justification**

* **SpaCy Model:** The en\_core\_web\_sm model is used to perform NER on resumes, extracting structured data such as names, skills, and experience sections
* **Python Environment**: Utilized Kaggle’s integrated environment for programming in Python for data cleaning, models deployment and assessment.
* **Backend and Frontend Stack:** Flask is used to develop the backend API, while React powers the frontend, allowing for seamless interaction with the resume parsing and classification functionalities.

**Why Kaggle?**

Kaggle was chosen because of many computational resources available for use in creating and deploying models that use NLP which requires large GPU usage. This will ensure that the system is able to parse several resumes within a short time as and when they come.

## **Data**

### **1. Resume Dataset**

### **Format:** The dataset includes resumes in the text and PDF forms.

### **Content:** Common sections found on the resumes are personal details, skills, qualification, employment history, achievements, education, training and certifications.

### **Training Data:** Used 1000 resumes collected from various fields for covering the diversification factor for the model.

### **Test Data:** 20 resumes were used to determine the effectiveness of the system.

### **2. Job Descriptions Dataset**

**Categories**: Information about job descriptions were collected for 20 positions, such as frontend developer, backend developer, data scientist.

**Purpose:** They are used by the classification model in order to determine how well the candidate on a resume matches to the description of the desirable skills and references for each position.

### **3. Data Preprocessing**

**Steps**: Text extraction from PDFs, data cleaning, normalization, and feature extraction.

**Tools Used**: Libraries such as pdfminer for text extraction and Scikit-learn for feature processing.

**Data Handling**:

**Positive Example**: A resume with clearly defined sections like "Skills" and "Experience" aligning well with job descriptions.

**Negative Example**: A resume with inconsistent formatting or missing critical sections, making it challenging to extract relevant information.

## **Implementation**

### **1. Resume Parsing with SpaCy and Regex**

**Model Used**: SpaCy en\_core\_web\_sm model, along with regex, is employed to extract structured data from resumes (e.g., names, skills, experience)..

**Preprocessing**: Resumes are converted from PDF to text and processed for effective information retrieval.

### **2. Classification Model for Job Matching**

**Objective**: To classify the parsed resume data based on the job type entered by the user.

**Methodology**:

* **Clustering:** K-Means clustering classifies resumes into job roles by grouping similar resumes based on feature vectors.
* **Text Matching:** Cosine similarity is applied to generate a match percentage, reflecting the resume’s alignment with job requirements.
* **Vectorization:** TF-IDF is used to transform resume data into feature vectors

**Output**: The system provides a score indicating the closeness of each resume to the job description, helping recruiters prioritize candidates.

## **Testing**

### **Importance of Testing**

Before development, defining the testing criteria was crucial to ensure reliable project outcomes. Testing was essential to validate that the resume parser could accurately extract information and the classification model could correctly match candidates to job descriptions.

### **1. Test Data and Scenarios**

* **Data Used**: 10 diverse resumes were used for testing, covering various formats and content types.
* **Testing Conditions**:

**Positive Test Case**: Resumes that were well-structured and clearly matched the job descriptions.

**Negative Test Case**: Resumes with unrelated skills, inconsistent formatting, or missing sections like "Education" or "Skills."

* **Evaluation Metrics**: Accuracy in parsing information, precision in job role classification, and the percentage match score.

### **2. Testing Approach**

The testing phase included evaluating the model's performance on both well-structured and poorly structured resumes.

Emphasis was placed on ensuring the system could handle real-world variations in resume formats, providing reliable results even with challenging data inputs.

## **Future Use and Enhancements**

### **1. Scalability**

**Integration with ATS**: The system can be integrated with existing Applicant Tracking Systems (ATS) for automated resume screening, providing a seamless experience for recruiters.

**API Development**: Building an API to allow external systems to leverage the resume parsing and classification functionality.

### **2. Continuous Model Improvement**

**Retraining Capability**: The system includes an option for retraining with new resume data to adapt to changing job market requirements.

**Multilingual Support**: Future versions will support resumes in multiple languages, expanding the system’s applicability across global markets.

### **3. Advanced Feature Expansion**

**Skill Gap Analysis**: Implementing a module to analyze skill gaps between candidates and job descriptions, offering actionable insights for upskilling.

**Diversity and Inclusion Analysis**: Adding metrics to assess diversity aspects in the hiring pipeline, promoting inclusive recruitment practices.