# Resume Analyzer Using NLP and Machine Learning (CS683 NLP)

## Automating Resume Evaluation for Efficient Recruitment


Team Members:

- Mayank Gupta (2101117)
- Manish Kumar (2101114)
- Harsh Rajput (2101081)

# Problem Statement

- **Challenges in Recruitment**:
  - Manual resume screening is time-consuming and inefficient.
  - Difficulty in objectively assessing the relevance of resumes to specific job descriptions.
  - High volume of applications can overwhelm HR teams.
  - The challenge of HR professionals lacking technical expertise.
  - The issue of job descriptions not listing all domain-specific skills.
  - The consequence of critical applicant skills going unnoticed and qualified candidates being overlooked.

- **Objective**:
  - Develop an automated system to analyze and rank resumes based on their relevance to a job description, highlighting crucial skills present in the applicant’s resume.
  - Skill Identification: Ensures that essential, domain-specific skills are recognized and displayed, even if not explicitly listed in the job description.

# Solution Overview

- Automated Resume Analysis:

  Streamlines the resume screening process by automatically extracting and evaluating resume content.

- Comprehensive Skill Identification:

  Detects and displays essential and domain-specific skills, ensuring critical competencies are recognized even if not explicitly mentioned in job descriptions.

- Objective and Efficient Ranking:

  Generates relevance scores for each resume, enabling HR teams to rank candidates effectively without requiring deep technical expertise.

- Enhanced Recruitment Efficiency:

  Reduces manual effort and minimizes the risk of overlooking qualified candidates, improving overall hiring outcomes.

# Tech Stack Used

- Programming Language
  - Python 3.x
    - Widely used for data analysis and ML.
    - Rich ecosystem for NLP and PDF processing.
    - Strong community support.

- Natural Language Processing (NLP)
  - spaCy
    - Efficient text processing and entity recognition.
    - Pre-trained models, scalable, easy integration.
  - Sentence Transformer
    - Semantic similarity analysis.
    - Generates sentence embeddings, versatile models like all-MiniLM-L6-v2.

- PDF Text Extraction
  - PyMuPDF (fitz)
    - High accuracy with complex layouts.
    - Extracts text, images, metadata.
    - Lightweight for bulk processing.
    - Fuzzy String Matching

# Tech Stack Used

- Fuzzy String Matching
  - RapidFuzz
    - Handles variations and typos.
    - High-performance, flexible partial matching.
    - Enhances skill identification accuracy.
- Data Handling and Reporting
  - pandas
    - Data manipulation and analysis.
    - Powerful DataFrames, easy CSV export.
- Pre-trained Models
  - spaCy's en_core_web_sm & SentenceTransformer's all-MiniLM-L6-v2
    - Foundation for NLP tasks and semantic similarity.
- Skill Keywords Dictionary
  - Custom Dictionary
    - Categorizes skills by 50+ job titles.
    - Facilitates accurate and extendable skill matching.

# Explanation and Methodology

- Text Extraction and Preprocessing
  - PDF Text Extraction
    - Tool Used: PyMuPDF (fitz)
    - Method: extract_text_from_pdf(pdf_path)
      - Extracts text content from PDF resumes.
      - Handles various PDF formats and layouts efficiently.
  - Text Preprocessing
    - Purpose: Clean and standardize the extracted text.
    - Method: preprocess_text(text)
      - Removes extra whitespace and non-ASCII characters.
      - Prepares text for NLP processing to enhance analysis accuracy.

# Explanation and Methodology

- Skill Extraction
  - Extracting Job Titles and Primary Skills
    - Tool Used: spaCy
    - Method: extract_job_title_and_primary_skills(job_description)
      - Parses the job description to identify relevant job titles.
      - Extracts primary skills mentioned explicitly in the job description.
  - Extracting All Skills from Resumes
    - Method: extract_all_skills(resume_text)
      - Identifies all possible skills present in the resume text.
      - Compares tokens and entities with our Skill Keywords Dictionary.
  - Matching Primary Skills
    - Method: match_primary_skills(resume_skills, job_description_skills)
      - Finds the intersection between skills from the resume and the job description.
      - Identifies primary skills that directly match the job requirements.
  - Extracting Secondary Skills
    - Method: extract_secondary_skills(resume_skills, primary_skills, job_titles)
      - Identifies additional relevant skills based on the extracted job titles.
      - Utilizes the Skill Keywords Dictionary to find related skills not explicitly mentioned in the job description.

# Explanation and Methodology

- Project Analysis
  - Extracting Project Section
    - Method: extract_project_section(resume_text)
      - Searches for common project-related headings in the resume.
      - Isolates the project section for focused analysis.
  - Extracting Project Skills
    - Method: extract_project_skills(project_section)
      - Uses spaCy to extract skills and technologies mentioned in projects.
  - Extracting Advanced Terms from Job Description
    - Method: extract_advanced_terms(job_description)
      - Identifies complex terms and phrases that signify advanced competencies.
  - Assessing Project Relevance
    - Method: assess_project_relevance(project_skills, required_skills, advanced_terms)
      - Fuzzy Matching: Uses RapidFuzz to handle variations and typos in skill names.
      - Relevance Scoring:
        - Skill Relevance: Proportion of required skills matched in projects.
        - Complexity Score: Presence of advanced terms from the job description in projects.
      - Overall Relevance Score: Weighted combination of skill relevance (80%) and complexity score (20%).

# Explanation and Methodology

- Experience Analysis
  - Extracting Experience Section
    - Method: extract_experience_section(resume_text)
      - Identifies and extracts the professional experience section from the resume.
  - Assessing Experience Relevance
    - Methods:
      - assess_experience_relevance(experience_description, required_skills, job_titles)
        - Evaluates relevance based on matched skills and job titles.
      - assess_experience_relevance_semantic(experience_description, job_description)
        - Calculates semantic similarity using SentenceTransformer.
      - calculate_experience_relevance(...)
        - Combines keyword relevance (60%) and semantic similarity (40%) for overall scoring.
  - Extracting Experience Duration
    - Method: extract_experience_duration(experience_description)
      - Uses regular expressions and dateutil parser to extract dates.
      - Calculates the duration of each experience in years.
  - Analyzing Total Experience
    - Method: analyze_total_experience(resume_text)
      - Searches for mentions of total years of experience in the resume.
      - Provides an overall measure of the candidate's professional background.

# Explanation and Methodology

- Scoring and Ranking
  - Calculating the Overall Score
    - Method: rank_resume(...)
      - Weighted Factors:
        - Primary Skills: Number of matched skills × 5 (high importance).
        - Secondary Skills: Number of additional relevant skills × 3.
        - Total Experience: Years of experience × 1.5.
        - Relevant Experience Duration: Duration × Average Relevance × 5.
        - Relevant Projects: Count × Average Relevance × 2.
  - Analyzing Resumes
    - Method: analyze_resume(resume_file, job_description)
      - Executes all analysis steps for a single resume, Compiles results into a comprehensive evaluation.
  - Batch Analysis and Reporting
    - Method: analyze_resumes(resume_files, job_description)
      - Processes multiple resumes and sorts them based on scores.
    - Reporting: generate_ranking_report(analysis_results)
      - Generates a CSV report of all analyzed resumes, Facilitates easy comparison and decision-making for HR teams.

# Explanation and Methodology

- Methodology Highlights
  - Combination of Keyword Matching and Semantic Analysis:
    - Ensures both explicit and implicit skills are recognized.
    - Balances literal matches with contextual understanding.
  - Weighted Scoring System:
    - Prioritizes critical factors such as primary skills and relevant experience.
    - Provides a quantitative basis for ranking candidates objectively.
  - Modular Design:
    - Functions are organized for clarity and reusability.
    - Facilitates easy maintenance and scalability.
  - Customization and Extensibility:
    - Skill keywords dictionary can be updated to reflect evolving industry needs.
    - System can adapt to different job roles and requirements.

# Conclusion

- Automated Resume Screening
  - Developed a Python tool using NLP and ML to automate resume evaluation, reducing manual effort and increasing recruitment efficiency.
- Effective Skill Identification
  - Accurately identifies explicit and implicit skills relevant to job descriptions, bridging the gap between HR and technical requirements, ensuring crucial skills aren't overlooked.
- Objective Candidate Ranking
  - Provides a quantitative relevance score for each resume, facilitating unbiased evaluation and aiding better hiring decisions.

# Next Steps

- Expand Skill Dictionary
  - Action: Add more job titles and emerging skills.
  - Benefit: Stay updated with industry trends and enhance matching accuracy.
- Integrate with ATS
  - Action: Connect the tool with existing HR workflows.
  - Benefit: Automate candidate sourcing and screening seamlessly.
- Enhance NLP Capabilities
  - Action: Explore advanced models like GPT-4.
  - Benefit: Improve context comprehension and relevance assessment.

# Github and Live App Link

- Github Link
  - https://github.com/manish92596/Smart-Resume-Analyzer/tree/main
- Live App Link (kindly access through college network only)
  - http://172.16.2.17:8501