📘 Major Project Assignment

ResumeParser & JobMatch Scorer (NLP)

🔍 Project Overview

In today's competitive hiring landscape, organizations receive thousands of resumes for a single job opening, making manual screening both time-consuming and error-prone. This project aims to develop an **AI-powered Resume Parsing and Job Matching System** that automates the process of resume evaluation and candidate ranking.

Using advanced **Natural Language Processing (NLP)** techniques such as **Named Entity Recognition (NER)** and **semantic similarity modeling**, the system will be capable of extracting essential details from resumes, interpreting job requirements, and calculating a **match score** that quantifies the alignment between a candidate's profile and a job description.

The resulting solution will simulate the core functionality of real-world **Applicant Tracking Systems (ATS)** used by major organizations, thereby improving hiring efficiency and accuracy while showcasing practical AI skills relevant to modern recruitment technology.

🎯 Objectives

The primary goals of this project are:

1. Automate the extraction of critical resume components (skills, education, experience) using NLP.

2. Analyze and interpret job descriptions to identify key requirements.

3. Design a robust algorithm to calculate a **match score** between resumes and job postings.

4. Rank candidates based on match scores to assist recruiters in identifying the most suitable profiles.

5. (Optional) Provide resume improvement feedback and visualization dashboards for recruiters.

📚 Suggested Dataset

Use the following publicly available datasets from Kaggle for model development and evaluation:

- 📄 **Resume Dataset (NLP):**
  https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset
  (Contains labeled resumes with skills, experience, and education details.)

- 📄 **Job Descriptions Dataset:**
  https://www.kaggle.com/datasets/sanjanchaudhari/job-descriptions-dataset
  (Collection of job postings suitable for semantic matching and requirement extraction.)

🛠️ Technical Requirements

- Python 3.x
- NLP Libraries: **spaCy**, **NLTK**, **transformers**, **sentence-transformers**
- Machine Learning Tools: **scikit-learn**, **pandas**, **numpy**
- Visualization: **matplotlib**, **seaborn**, **plotly** (optional)
- Deployment (Optional): **Streamlit** / **Flask**

🧭 Project Workflow & Deliverables

The project is divided into structured phases. Each phase contains actionable tasks designed to build the system incrementally.

Phase 1: Data Understanding & Preprocessing

Task 1: Import the resume and job description datasets. Perform exploratory data analysis (EDA) to understand the data distribution, structure, and quality.

Task 2: Preprocess text data by:

- Converting resumes and job descriptions into plain text.
- Removing stopwords, punctuation, and special characters.
- Tokenizing and lemmatizing text.

Task 3: Ensure both resumes and job descriptions are normalized and structured for downstream NLP tasks.

Phase 2: Resume Parsing & Information Extraction (NER)

Task 4: Develop a Named Entity Recognition (NER) pipeline to extract key information from resumes:

- Candidate name and contact information
- Skills (technical and soft)
- Education details (degree, institution, graduation year)
- Work experience (companies, roles, duration)

Task 5: Fine-tune or customize the NER model to improve accuracy. Evaluate extraction performance using precision, recall, and F1-score.

Phase 3: Job Description Analysis

Task 6: Build a module to parse job descriptions and extract:

- Required skills and technologies
- Minimum experience level

- Desired educational qualifications

Task 7: Represent extracted information in a structured format (e.g., dictionary or JSON) for direct comparison with parsed resume data.

Phase 4: Semantic Matching & Score Calculation

Task 8: Implement a **semantic similarity engine** using sentence embeddings (e.g., BERT, SBERT) to measure how closely a candidate's resume aligns with a job description.

Task 9: Design a **match scoring algorithm** by combining:

- Skill similarity score
- Experience alignment score
- Education compatibility score

The final score should range between **0–100**.

Phase 5: Candidate Ranking & Insights

Task 10: Develop a ranking system to sort resumes based on their match scores for a specific job description.

Bonus (Optional):

- Visualize the top matches with bar charts or dashboards.
- Generate feedback reports suggesting how candidates can improve their resumes.

📤 Submission Guidelines

Students must submit the following deliverables:

1. **Code Implementation:**
    - A Jupyter Notebook containing all preprocessing, model building, and evaluation steps.
2. **Project Report (2–4 pages):**
    - Problem statement, methodology, results, and key insights.
3. **Sample Output:**
    - Match scores for at least 10 resumes against a selected job description.
4. **Optional Demo:**
    - Streamlit or Flask app for interactive resume-job matching.

🏁 Project Outcomes

By the end of this project, students will have built a functional **AI-powered recruitment system** capable of automating resume screening and matching candidates to job descriptions with a high degree of accuracy. This project demonstrates proficiency in **NLP, semantic modeling, information extraction, and applied AI engineering** — essential skills for modern data science and machine learning roles.