INNOMATICS RESEARCH LABS

Theme 2 - Automated Resume Relevance Check System

Problem Statement

At **Innomatics Research Labs**, resume evaluation is currently **manual, inconsistent, and time-consuming**. Every week, the placement team across Hyderabad, Bangalore, Pune, and Delhi NCR receives **18–20 job requirements**, with each posting attracting **thousands of applications**.

Currently, recruiters and mentors manually review resumes, matching them against job descriptions (JD). This leads to:

- **Delays** in shortlisting candidates.
- **Inconsistent judgments**, as evaluators may interpret role requirements differently.
- **High workload** for placement staff, reducing their ability to focus on interview prep and student guidance.

With hiring companies expecting **fast and high-quality shortlists**, there is a pressing need for an **automated system** that can scale, be consistent, and provide actionable feedback to students.

Objective

The **Automated Resume Relevance Check System** will:

1. **Automate resume evaluation** against job requirements at scale.
2. **Generate a Relevance Score (0–100)** for each resume per job role.
3. **Highlight gaps** such as missing skills, certifications, or projects.
4. Provide a **fit verdict (High / Medium / Low suitability)** to recruiters.
5. Offer **personalized improvement feedback** to students.
6. Store evaluations in a web-based dashboard accessible to the placement team.

This system should be robust, scalable, and flexible enough to handle **thousands of resumes weekly**.

Sample Data

Click here to download sample data.

Proposed Solution

We propose building an **AI-powered resume evaluation engine** that combines **rule-based checks** with **LLM-based semantic understanding**.

The system will:

- Accept **resumes (PDF/DOCX)** uploaded by students.
- Accept **job descriptions** uploaded by the placement team.
- Use **text extraction + embeddings** to compare resume content with job descriptions.
- Run **hybrid scoring**:
  - **Hard match (keywords, skills, education)**
  - **Soft match (semantic fit via embeddings + LLM reasoning)**
- Output a **Relevance Score, Missing Elements, and Verdict**.
- Store results for the placement team in a searchable **web application dashboard**.

This approach ensures both **speed (hard checks)** and **contextual understanding (LLM-powered checks)**.

Workflow

1. **Job Requirement Upload** - Placement team uploads job description (JD).
2. **Resume Upload** - Students upload resumes while applying.
3. **Resume Parsing**
   - Extract raw text from PDF/DOCX.
   - Standardize formats (remove headers/footers, normalize sections).
4. **JD Parsing**
   - Extract role title, must-have skills, good-to-have skills, qualifications.
5. **Relevance Analysis**
   - Step 1: Hard Match – keyword & skill check (exact and fuzzy matches).
   - Step 2: Semantic Match – embedding similarity between resume and JD using LLMs.
   - Step 3: Scoring & Verdict – Weighted scoring formula for final score.
6. **Output Generation**
   - Relevance Score (0–100).
   - Missing Skills/Projects/Certifications.
   - Verdict (High / Medium / Low suitability).
   - Suggestions for student improvement.
7. **Storage & Access**
   - Results stored in the database.
   - The placement team can search/filter resumes by job role, score, and location.
8. **Web Application**
   - Placement team dashboard: upload JD, see shortlisted resumes.

Tech Stack (Core Resume Parsing, AI Framework and Scoring Mechanism)

- **Python** – primary programming language.
- **PyMuPDF / pdfplumber** – extract text from PDFs.

- **python-docx / docx2txt** – extract text from DOCX.
- **spaCy / NLTK** – entity extraction, text normalization.
- **LangChain** – orchestration of LLM workflows.
- **LangGraph** – structured stateful pipelines for resume–JD analysis.
- **LangSmith** – observability, testing, and debugging of LLM chains.
- **Vector Store (Chroma / FAISS / Pinecone)** – for embeddings and semantic search.
- **LLM Models** – OpenAI GPT / Gemini / Claude / HuggingFace models for semantic matching & feedback generation.
- **Keyword Matching** – TF-IDF, BM25, fuzzy matching.
- **Semantic Matching** – embeddings + cosine similarity.
- **Weighted Score** – combine hard and soft matches into a final score.

Tech Stack (Web Application) - You can choose other tech stack for FE

- **Flask / FastAPI** – Backend APIs to process uploads, run OMR evaluation, and serve results.
- **Streamlit (MVP)** – Frontend for evaluators (upload, dashboard, review).
- **SQLite / PostgreSQL** – Database for storing results, metadata, and audit logs.

INNOMATICS RESEARCH LABS

How to win a Pizza Party?

Pizza Parties and a cash pool of 25,000 INR will be announced soon.

Check the last page on the main document 👉 Click here