RAJALAKSHMI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

ACADEMIC YEAR 2025 - 2026

SEMESTER III

ARTIFICIAL INTELLIGENCE LABORATORY

MINI PROJECT REPORT

REGISTER NUMBER	
NAME	
PROJECT TITLE	
DATE OF SUBMISSION	
FACULTY IN-CHARGE	**Mrs. M. Divya**

Signature of Faculty In-charge

INTRODUCTION:

Artificial Intelligence (AI) has transformed how we process information and make decisions. One of its practical applications is in automating recruitment processes, where AI can efficiently evaluate resumes and match them to job descriptions. Manually reviewing resumes is time-consuming, prone to human error, and often inconsistent. Automating this process ensures faster, objective, and more accurate assessments.

This project implements an Automated Resume Relevance Checker that evaluates candidate resumes against a given Job Description (JD). It combines hard matching (keyword-based comparison) with soft matching (semantic similarity using Sentence Transformers) to calculate a hybrid relevance score. Additionally, it identifies missing skills in resumes and generates personalized improvement suggestions for candidates.

The main goal of this project is to streamline recruitment by providing quantitative and qualitative insights about candidates' fit for a role, helping recruiters quickly shortlist the most suitable applicants and helping candidates understand areas for improvement.

PROBLEM STATEMENT:

Recruiters and hiring managers often face the challenge of **manually reviewing large volumes of resumes** for each job opening. This process is **time-consuming, prone to human error, and inconsistent**, especially when matching resumes against complex job descriptions. Candidates may also struggle to identify the **skills they are missing** to meet job requirements, leading to a lack of clarity in their career development.

The problem addressed in this project is the need for an **automated, accurate, and efficient system** that can:

1. Evaluate resumes against a job description.
2. Quantify relevance using both **keyword-based (hard)** and **semantic (soft)** similarity.
3. Highlight missing skills and provide actionable suggestions for improvement.

The goal is to **streamline the recruitment process** for organizations while simultaneously giving candidates **insights to enhance their profiles**, making the hiring process faster, more objective, and data-driven.

GOAL:

The goal of this project is to **develop an automated system** that evaluates resumes against a job description and provides meaningful insights for both recruiters and candidates. Specifically, the system aims to:

1. **Quantify Resume Relevance**: Compute a hybrid score using keyword-based (hard) and semantic (soft) matching to rank candidates objectively.
2. **Identify Skill Gaps**: Detect missing skills in resumes compared to the job requirements.

3. **Provide Improvement Suggestions**: Offer actionable recommendations to help candidates enhance their profiles.
4. **Visualize Results**: Present scores, top candidates, and overall analysis in an intuitive table and graphical format.

Expected Result: Recruiters can quickly shortlist the most suitable candidates, while candidates gain clarity on areas to improve.

Possibilities:

- Extendable to multiple domains or industries.
- Can incorporate advanced AI models for more precise semantic matching.
- Enables data-driven recruitment and reduces human bias.

THEORETICAL BACKGROUND:

1. Background of the Problem

Recruitment is a critical process for organizations, but manually evaluating resumes against job descriptions is often **time-consuming, inconsistent, and prone to bias**. Candidates may possess varying levels of skills, and recruiters need a systematic method to **objectively assess relevance**. Automating this process using Artificial Intelligence (AI) can significantly reduce human effort, improve accuracy, and provide actionable insights.

2. Algorithms and Techniques Used

1. **Keyword-based Matching (Hard Scoring)**
    - Uses **TF-IDF (Term Frequency–Inverse Document Frequency)** to represent the JD and resume text as numerical vectors.
    - **Cosine Similarity** calculates the similarity between the two vectors, providing a measure of **keyword overlap**.
    - **Strengths**: Simple, interpretable, effective for detecting exact skill matches.
    - **Limitations**: Fails to capture semantic meaning (e.g., “Python programming” vs. “Python coding”).
2. **Semantic Matching (Soft Scoring)**
    - Uses **Sentence Transformers** (e.g., all-MiniLM-L6-v2) to generate **contextual embeddings** of the JD and resume text.
    - Cosine similarity between embeddings captures **semantic similarity**, even if the exact keywords differ.
    - **Strengths**: Captures meaning beyond keywords, useful for flexible phrasing.
    - **Limitations**: Slightly more computationally intensive than keyword matching.
3. **Hybrid Scoring**
    - Weighted combination of **hard and soft scores** (e.g., 0.6 hard + 0.4 soft).
    - Provides a **balanced evaluation**, combining precise keyword detection with semantic understanding.

---

3. Literature Survey

- **Traditional Approaches**: Earlier systems used only keyword matching or boolean search. While fast, they often miss semantically similar skills expressed differently.
- **Machine Learning Approaches**: TF-IDF combined with classical ML models (e.g., logistic regression or SVM) can classify resume relevance.
- **Deep Learning Approaches**: Modern approaches leverage **transformer-based embeddings** (BERT, Sentence-BERT) for semantic similarity. Studies show these embeddings outperform classical methods in understanding context and meaning.
- **Hybrid Methods**: Recent research emphasizes combining **keyword-based and semantic-based scoring** for better accuracy and interpretability.

---

4. Justification for Choosing the Algorithm

The hybrid approach was chosen because it **leverages the strengths of both keyword and semantic matching**:

- **Keyword Matching (Hard Score)** ensures that **specific required skills** are present in the resume.
- **Semantic Matching (Soft Score)** ensures that **similar or equivalent skills phrased differently** are recognized.
- The **weighted hybrid score** provides an **objective and interpretable measure** of relevance, making it practical for recruitment.

This approach is **efficient, accurate, and adaptable**, making it suitable for both recruiters and candidates to evaluate resumes effectively.

ALGORITHM EXPLANATION WITH EXAMPLE:

Algorithm: Hybrid Resume Relevance Scoring

The project uses a **hybrid scoring algorithm** that combines **hard keyword matching** and **soft semantic matching** to evaluate the relevance of resumes against a job description (JD).

Step-by-Step Explanation:

1. **Input**:
    - Job Description (JD) in PDF/DOCX format.
    - One or more candidate resumes in PDF/DOCX format.
2. **Text Extraction**:
    - Extract text from JD and resumes using pdfplumber for PDFs and python-docx for DOCX files.
3. **Role and Skill Extraction**:
    - Detect role title from JD text (e.g., “Data Scientist Intern”).

    - Identify key skills mentioned in the JD using a predefined skill list (e.g., Python, Machine Learning, SQL).
4. **Hard Score (Keyword Matching)**:
    - Convert JD and resume text into **TF-IDF vectors**.
    - Compute **cosine similarity** between JD and resume vectors.
    - Score ranges from 0–100.
    - **Purpose**: Ensures the resume contains the **exact required skills**.
5. **Soft Score (Semantic Matching)**:
    - Encode JD and resume text into **sentence embeddings** using **Sentence Transformers**.
    - Compute **cosine similarity** between JD and resume embeddings.
    - Score ranges from 0–100.
    - **Purpose**: Captures **semantic meaning**, recognizing equivalent skills or phrasing.
6. **Hybrid Score Calculation**:
    - Combine scores using weights:

$$\text{Hybrid Score} = (0.6 \times \text{Hard Score}) + (0.4 \times \text{Soft Score})$$

7. **Verdict Assignment**:
    - High (≥70), Medium (40–69), Low (<40).
8. **Missing Skills and Suggestions**:
    - Identify skills in JD not present in resume.
    - Generate improvement suggestions using predefined templates.
9. **Output**:
    - A table with **Hybrid Score, Verdict, Missing Skills, Suggestions**.
    - Top 3 candidates highlighted with stars.
    - Optional bar chart visualization of all scores.

IMPLEMENTATION AND CODE

The project is implemented in **Python** using **Streamlit** for the user interface and several libraries for AI and text processing:

- **Libraries Used**:
    - streamlit – web app interface
    - pandas – data manipulation
    - plotly.express – visualization
    - pdfplumber & python-docx – text extraction from PDFs and Word files
    - scikit-learn – TF-IDF vectorization, cosine similarity
    - sentence-transformers – semantic similarity with embeddings

CODE:

```
def extract_text(file):
    if file.name.endswith(".pdf"):
        return extract_text_from_pdf(file)
    elif file.name.endswith(".docx"):
        return extract_text_from_docx(file)
    return ""
def extract_skills_from_jd(jd_text, skill_list=PREDEFINED_SKILLS):
    return [skill for skill in skill_list if skill.lower() in jd_text.lower()]
def generate_improvement_suggestion(missing_skills):
    if not missing_skills: return "All critical skills present!"
    return "\n".join([f"Practice {skill}" for skill in missing_skills])
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer, util
def compute_hard_score(jd_text, resume_text):
    tfidf = TfidfVectorizer(stop_words="english").fit_transform([jd_text, resume_text])
    return cosine_similarity(tfidf[0:1], tfidf[1:2])[0][0]*100
def compute_soft_score(jd_text, resume_text):
    model = SentenceTransformer('all-MiniLM-L6-v2')
    jd_emb = model.encode(jd_text, convert_to_tensor=True)
    resume_emb = model.encode(resume_text, convert_to_tensor=True)
    return util.cos_sim(jd_emb, resume_emb).item()*100
def compute_hybrid_score(hard, soft, hard_weight=0.6, soft_weight=0.4):
```

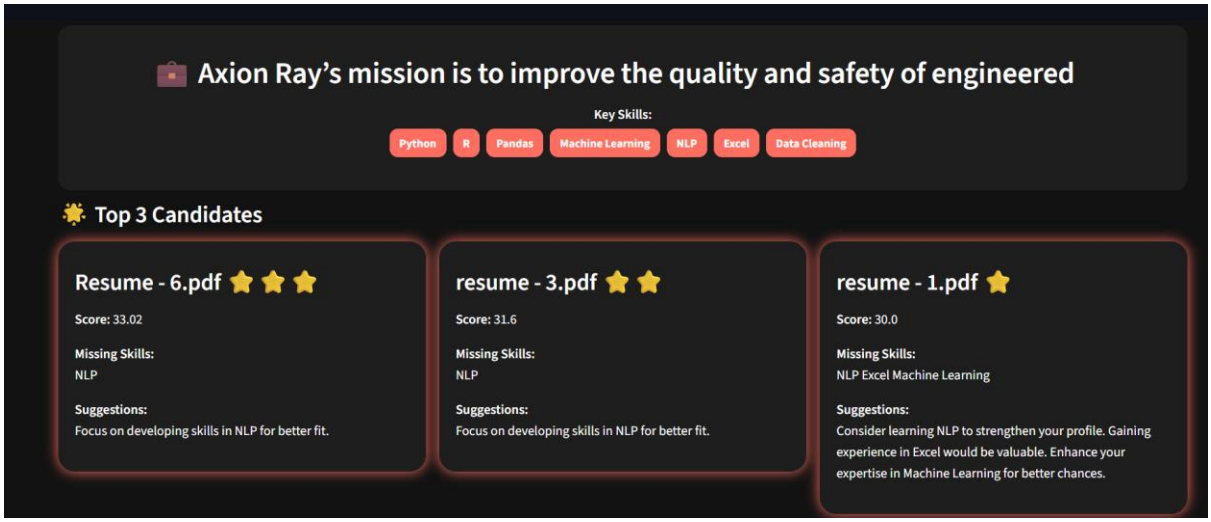
```
    return round(hard*hard_weight + soft*soft_weight, 2)

import streamlit as st


st.title("📄 Automated Resume Relevance Checker")

jd_file = st.file_uploader("Upload Job Description", type=["pdf","docx"])

resume_files = st.file_uploader("Upload Resumes", type=["pdf","docx"],
accept_multiple_files=True)

if jd_file and resume_files:

    st.button("Evaluate", on_click=evaluate_resumes, args=(jd_file, resume_files))
```

OUTPUT:

The application produces the following outputs:

1. **Top Candidates Display**
    - Shows **top 3 candidates** with hybrid score, missing skills, and suggestions.
2. **Full Results Table**
    - Displays **all resumes**, hybrid score, verdict (High/Medium/Low), missing skills, and improvement suggestions.
3. **Graphical Visualization**
    - Horizontal bar chart showing hybrid scores of all resumes.
    - Top candidates easily identified.

SAMPLE OUTPUT:



Future Enhancements

1. **Multi-language Support** – Evaluate resumes in different languages.
2. **Experience & Certification Scoring** – Include candidate experience and certifications for scoring.
3. **Integration with ATS** – Connect directly with Applicant Tracking Systems for large-scale recruitment.
4. **Advanced Semantic Models** – Use larger transformer models (e.g., BERT, RoBERTa) for improved accuracy.
5. **Skill Weighting** – Allow recruiters to assign **different weights** to critical vs optional skills.
6. **Dashboard Analytics** – Add interactive analytics for recruiters to track trends across multiple hires.

Git Hub Link of the project and report	**https://github.com/Dheepaksuresh/resume_relevance_system**

REFERENCES

1. Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed., draft). [Online Book Draft]
    - https://web.stanford.edu/~jurafsky/slp3/
2. Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
3. Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. arXiv:1908.10084.
    - https://arxiv.org/abs/1908.10084
4. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
5. McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.
6. Brownlee, J. (2020). *Master Machine Learning Algorithms*. Machine Learning Mastery.
    - https://machinelearningmastery.com/master-machine-learning-algorithms/