**Resume Ranker Based on Job Description**

This project allows users to upload multiple resumes and compare them with a given job description to determine the most relevant candidates using **TF-IDF** and **Cosine Similarity**.

**🧠 Technologies Used:**

* **Python**
* **Streamlit**
* **pdfplumber**
* **python-docx**
* **scikit-learn (TF-IDF, Cosine Similarity)**

**📁 Project Files**

**utils.py**

python

CopyEdit

import pdfplumber

import docx

def read\_pdf(file):

text = ""

with pdfplumber.open(file) as pdf:

for page in pdf.pages:

text += page.extract\_text() or ""

return text

def read\_docx(file):

doc = docx.Document(file)

return "\n".join([para.text for para in doc.paragraphs])

**resume\_matcher.py**

python

CopyEdit

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

import os

from utils import read\_pdf, read\_docx

def extract\_text(file):

ext = os.path.splitext(file.name)[1].lower()

if ext == '.pdf':

return read\_pdf(file)

elif ext == '.docx':

return read\_docx(file)

return ""

def rank\_resumes(resume\_files, job\_description):

resume\_texts = [extract\_text(file) for file in resume\_files]

all\_texts = [job\_description] + resume\_texts

vectorizer = TfidfVectorizer()

vectors = vectorizer.fit\_transform(all\_texts)

similarities = cosine\_similarity(vectors[0:1], vectors[1:]).flatten()

ranked = sorted(zip(resume\_files, similarities), key=lambda x: x[1], reverse=True)

return ranked

**app.py**

python

CopyEdit

import streamlit as st

from resume\_matcher import rank\_resumes

st.title("🔍 Resume Ranker Based on Job Description")

st.write("Upload multiple resumes and a job description to rank them based on relevance.")

jd\_text = st.text\_area("Paste Job Description", height=200)

resume\_files = st.file\_uploader("Upload Resumes (PDF/DOCX)", type=["pdf", "docx"], accept\_multiple\_files=True)

if st.button("Rank Resumes"):

if not jd\_text or not resume\_files:

st.warning("Please upload resumes and provide a job description.")

else:

ranked = rank\_resumes(resume\_files, jd\_text)

st.success("Resumes Ranked Successfully!")

st.write("### 📊 Ranked Resumes:")

for i, (file, score) in enumerate(ranked, 1):

st.write(f"\*\*{i}. {file.name}\*\* – Similarity Score: `{score:.2f}`")

**💻 Output Screenshot (Description)**

On running the app with:

* A job description for **Full Stack Developer**
* Multiple resumes in PDF/DOCX format

The application displays:

* Ranked list of resumes
* Each with a similarity score (0.00 to 1.00)

Example:

yaml

CopyEdit

1. john\_resume.pdf – Similarity Score: 0.85

2. jane\_cv.docx – Similarity Score: 0.77

3. mike\_resume.pdf – Similarity Score: 0.68

