Smart Resume Screener — Colab Demo

Project Title: Smart Resume Screener
Author: Aamisha Nagpal
Notebook (Colab):

https://colab.research.google.com/drive/12jAYUJCTRdwwpp1ROOCvzqtizTvY67zF?usp=sharing

A Colab-based tool that ingests PDF/DOCX/TXT resumes and a job description, extracts structured fields (name, contact, education, experience, skills), computes an automated match score (0–10) using semantic embeddings (sentence-transformers) and optional LLM justification (OpenAI), and saves shortlisted candidates to a downloadable CSV.

Scope of Work

- **Input:** PDF/TXT/DOCX resume file(s) + Job Description (plain text).
- **Extraction:** Parse name, email, phone, education lines, experience (estimated years), and canonicalized skills.
- **Matching:** Compute semantic similarity using sentence-transformers (all-MiniLM) and map to 0–10 score; add skill-match bonuses.
- **LLM Justification (optional):** Generate a 3-bullet human-readable justification using OpenAI (if OPENAI_API_KEY provided).
- **Shortlisting:** Automatically mark and save shortlisted candidates (configurable threshold) to a SQLite DB and exportable CSV.
- **Interactive UI:** Gradio interface for uploading resumes, pasting JD text, and viewing output & shortlist.

How to run (quick instructions)

1. Open the Colab notebook link above.
2. Run cells in order (top → bottom).
    - First cell installs packages and downloads the spaCy model.
    - Later cells load embedding model (may take ~30s).
3. (Optional) To enable LLM justification, set your OpenAI key.
4. Launch the Gradio UI cell.
    - Upload a resume (PDF/DOCX/TXT).
    - Paste a job description (plain text).
    - Leave the OpenAI key blank to skip LLM.
    - Click run → wait a few seconds → view Output panel.
5. Export shortlist (CSV): run the export_shortlisted_csv() helper cell or use the Gradio “Show Shortlist” interface

Shortlisting logic

- Final score = blend of embedding-derived score (mapped 0–10) and optional LLM score.
- Default shortlisting threshold used in demo: 7.0 (configurable).
- Result stored per run in SQLite table resumes with fields:
    - id, filename, parsed_json, embed_json, llm_json, final_score, shortlisted (0/1), created_at
- You can export shortlisted rows as CSV for submission/presentation.

Example outputs

Screenshot of an example