NAME- SHUBH TANDON

Resume Matching with Job Descriptions Using PDF CVs

Objective: Build a PDF extractor to pull relevant details from CVs in PDF format, and match them against the job descriptions from the Hugging Face dataset.

Data sources-

- Download CVs zip fil from https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset
- Access JD dataset using - load_dataset("jacob-hugging-face/job-descriptions") URL- https://huggingface.co/datasets/jacob-hugging-face/job-descriptions/viewer/default/train?row=0

Libraries installed

- PyPDF2
- Dataset
- Transformers

Explanation of Code

1. Loading JDs- job description loaded with load_dataset(“”) and trainset is extracted having 833 jobs. Stored all features into a dataset names- “job_profile”
2. Loaded all libraries needed for pdf extractor
3. Imported DistilBertTokenizer for embedding extracting.
4. Function extract_text_from_pdf and extract_Education_And_skills are self explainatory
5. Process_resume adds education , skills, category of each resume mapped to resume_id in a dictionary “resume_info”
6. Zip file extraction takes place and to access resume pdf, we should moved data->catogary->pdf. We ran through every categories and moved the “resume_path” to “process_resume” function to store details of every resume in a dictionary
7. Take initially for small size i.e., run only for 1 category to see if result is correct for Skills, education, category. Then expand to whole resume dataset.
8. The code ran for long time. In order to prevent long running of extractor multiple times, the pdf extractor is stored in a csv file “**resume_info.csv**” with columns of resume_id, skills,education,category
9. This csv file can directly be loaded and can be used for matching of resume to JDs without running extracted anytime again in future. This csv this attached to solution presented by me.
10. Define “extract_Embeddings” function which can be used for extraction of embeddings of JDs and resume content.
11. Store both resume csv fil and JD into datasets.

12. Find embeddings of every “model_response” column of every job profiles and store into “job_profile_embeddings” list as “model_response” only contain the detail about skills and education required.
13. Do similar thing for resume. Combine Skills and education together and then find their embeddings to save processing time as in JD also only model_response column contain all details
14. I skipped matching embeddngs of category because there appears ambiguity in JD about this. Like is resume category is Engineer and job profile is software developer then embeddings comparison would give poor result but they are more or less same and must be collected.
15. “Similarity_scores” list stores resume_id, corresponding company name , similarity_Score between them by iterating each resume to each job. DataFram this similarity_score
16. Sort the “similarity_scores” list in descending order according to their similarity.
17. “top_matches_dict” is a dictionary which maps company name to top 5 matched resume done after grouping the “similarity_score” according to company name and then iterating to each company.
18. Concat every results of “top_matches_dict” and convert to excel sheet
19. The excel sheet “top_matches_df.xlsx” contains 5 resume with their similarity score to each company in order. Columns of resume_id, company name and similarity score is there.
20. The obtained excel sheet is attached to solution.

Challenges

To extract the details from resume is something to which I was new. From zip file, reached to each resume, I saw pattern that education and skills details are in line next to their section so from their I developed idea to store lines next to them into list with section of skills and education and category can be extracted from the path only by splitting it with “/” and storing the second last name.

Running code multiple time could go worse, so stored the pdf extractor into excel sheet with all details extracted and used it further without running extractor code again.

The category of resume and JD somewhat has synonyms in their name so embeddings was giving wrong comparision. So neglected that category column.

Running iteratively to each resume and JD can take a lot of time so bifurcated the embeddings values for resume and JD separately and then compared them one-by-one.

Rest all sorting and grouping was easy.