AMERICAN UNIVERSITY OF BEIRUT

MAROUN SEMAAN FACULTY OF ENGINEERING & ARCHITECTURE

EECE 490: Introduction to Machine Learning

Documentation

Career Guidance System

Dimitri Bassil

Charbel Abou Younes

Clara Fakhry

Date : 13 / 12 /2024

Table of contents:

1. Introduction: What is career guidance system?

Career Guidance System is an AI-powered application designed to assist fresh graduates in finding jobs that best suit their profiles. The application analyzes the user's CV to extract features and skills, then presents career recommendations tailored to their qualifications. For each suggested career path, the system evaluates the candidate's strengths and weaknesses, providing insights to help them assess their chances of success and identify areas for improvement.

To enhance the user experience, the application invites users to rate the proposed job suggestions and provide feedback. This feedback is used to refine career recommendations. Additionally, the program creates specific goals for each career path, broken into manageable steps. Users can track their progress toward these goals, and the app offers resources to support them in completing the remaining steps.

2. Code details

2.1. Imported libraries:

- tempfile: to save the cv in a temporary pdf file
- os: to work with the temporary pdf file
- fitz (PyMuPDF) : to be able to extract a text from a pdf file
- openai (0.28) : to use chatgpt API for prompting
- streamlit (as st): to create a user interface

2.2. Career guidance functions

The following function are the elementary function that are used to do each step of the career guidance process

- get_cv_feature: extract CV features based on the CV text using prompt
- generate_potential_roles: get potential roles that suits the person based on CV features
- analyze_roles_with_cv: get weakness and strength for each role based on the CV and the suggested roles
- get_career_path_description: get a description for the input suggested roles
- Calculate_skill_match: get a match (percentage) between the user skills and the required skills for a job based on CV features and suggested roles (or refined roles)

- collect_user_feedback_and_refine_recommendations : generate a refined recommendation and skill match percentages based on user feedback and rating, initial recommendation and cv feautures

- identify_skill_gaps_and_recomend_resources: identify the skill gaps based on CV features and refined recommendation then generate recommended resources to help the user

- get_motivational_quote : generate a motivational quote for career guidance

- calculate_goal_progress : calculate the percentage of completed steps

- generate_smart_goals: based on the final refined recommendation get smart goals for every career with detail steps for each goal

- interactive_feedback_loop: based on initial recommendation and cv features, take user feedback and rating and generate refined recommendation and skill match percentages. loop over the process until the user is satisfied.(at each iteration the user provide a new feedback and rating and gets new recommendation and skill match)

2.3. Backend functions:

These functions combine the career guidance functions together. They were used to develop the backend so the outputs are printed instead of being displayed on the UI. The initial steps are repetitive but each function adds a new feature. Some of these functions were later adapted to be used in the UI with streamlit

- career_recomendation_system_with_feedback_and_skill_match: based on the pdf extract cv feautures, do job analysis and initial recommendation then collect feedback and generate refined recommendations and skill match

- skill_gap_analysis_with_recomendation: based on the pdf do the precious steps to get refined recommendations and skill match then identify skill gaps and recommend resources

- career_goal_tracking: based on pdf do all the previous steps to get refined recommendations then generate SMART goals and allow the user to select a goal to track and input completed steps then provide the user recommended resources to help with the missing steps

- interactive_recomendation_system_with_feedback: based on the pdf do all the previous steps to get cv features, initial recommendation and job analysis then use user feedback to create refined recommendation in a loop until the user is satisfied

2.4. Frontend functions:

These functions are adapted versions of the backend functions to work with streamlit. They are used in the user interface.

- collect_user_feedback_and_refine_recomendation_ui: based on initial recomedation, job analysis and cv features, ask the user for feedback and generate refined recomendations

- career_recomendation_engine_with_feedback_and_skill_match_ui: based on the pdf do all the steps to get initial recommendation, job analysis and cv features, ask the user for feedback then generate refined recommendation and skill match

- generated_smart_goals : based on the final refined recommendation get smart goals for every career with detail steps for each goal and parse the goal into a list to use for the UI

- interactive_feedback_loop_ui: based on initial recommendation and cv features, take user feedback and rating in the UI and generate refined recommendation and skill match percentages. Loop over the process until the user is satisfied.

2.5. Helper functions:

The following functions are helper that were used to translate, call openai API or process pdf or text

- dynamic_translate : translate a text to the desired language (by using open ai API)

- get_chat_gpt_response : used to get a response from a certain prompt (input) by calling openai API using gpt-4

- Extract_text_from_pdf: get a text from the pdf file uploaded by the user by using fitz (PyMuPDF)

- Extract_career_titles: get a list of titles by processing refined job recommendations (text)

2.6. User Interface

The user interface was created using streamlit. The function app() is the main function that creates the user interface using the other function.

2.6.1. Set page configuration

This is the first step in creating a streamlit app page. It configures the page title and layout.

2.6.2. Customize the page

The page is customized using st.markdown:

- set the color to green (we chose this color to represent success)
- set the text font to arial sans serif
- create the slidebar design
- create the buttom design (like upload cv)

2.6.3. Navigation:

- add a title to the sidebar and a radio button to choose options from : Home , Upload CV , View Recommendations, Feedback and Refinements, Generate and track SMART Goals
- selected option from the radio button is stored in the variable navigation
- The navigation variable is then used in conditional statements to dynamically render the corresponding section of the app
- Create a dropdown menu for the user to select a language and save it in the variable language. (the translation is integrated in all the following steps)

2.6.4. Home page:

- Display title of the page
- Display a motivational quote
- Display an image

2.6.5. Upload CV:

- Add a button to upload cv and save the file in uploaded_file
- Extract text using extract_text_from_pdf and save it in cv_text
- Display the cv_text

2.6.6. View Recommendation:

- Check if the cv is uploaded and text extracted (if not ask the user to do this first)
- Extract features using get_cv_features and save it in cv_features
- Generate the job roles using generate_potential_roles and save it in potential_roles

- Display potential roles
- Analyze roles using analyze_roles_with_cv and save it to in role_analysis
- Display role_analysis

2.6.7. Feedback and refinement:

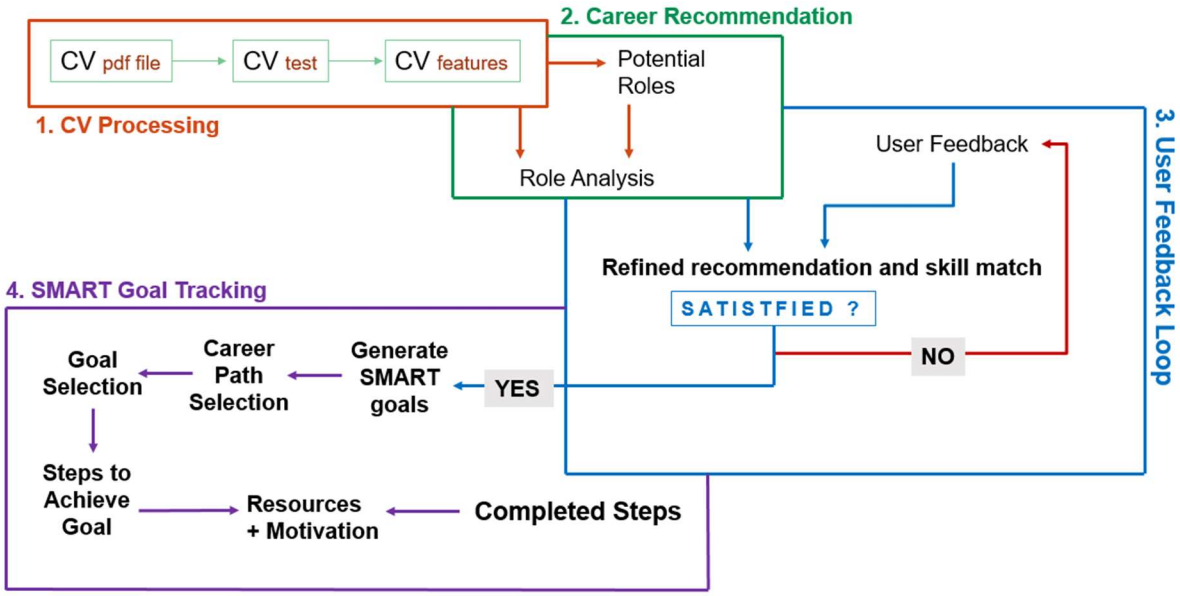
- Check that cv_text, cv_features and potential_roles are available (if not ask the user to complete previous steps first)
- Check if role_analysis is available (if not do the analysis as we did in the previous step (2.6.6))
- Based on the feedback create refined_recommendation and refined_skill_match_data using interactive_feedback_loop_ui (keep on refining based on feedback until the user is satisfied)
- Display the final refined_recommendation and the refined_skill_match

2.6.8. Generate and Track SMART goals:

- Check that refined_recommendation is available (if not ask the user to complete previous steps first)
- Extract the career titles from refined recommendation using extract_carrer_titles and save it in career_titles
- Create SMART goals with steps for each career title using generated_smart_goals and display all the goals and steps
- Add a dropdown menu where the user can select a specific career recommendation
- display the goals of the selected career
- Add a dropdown menu where the user can select a specific goal within the selected career
- Display the steps for the selected goal
- Add a text input for the user to enter the steps he finished (comma separated)
- Display complete goals and the percentage of completion
- Provide motivational advice and resources for the uncompleted steps using
- Add a finish button for the user to stop the goals tracking
- Create a motivational quote and display it

2.7. Code workflow:

All the above code is summarized in the below figure :



3. Usage

3.1. Using Docker

Step 1: provide your openai API key in the code

openai.api_key= "insert key here"

Step 2: build and run the docker image using the following command

- For latest stable version (systemenglish):

  docker build -f Dockerfile.systemenglish -t systemenglish-image .

  docker run -p 8505:8505 systemenglish-image

- For latest release with multilanguage support (finalapp):

  docker build -f Dockerfile.finalapp -t finalapp-image .

  docker run -p 8506:8506 finalapp-image

Step 3: a link will appear on the terminal. Click on it to open the UI in your browser.

3.2. Without Docker

Step 1: make sure to install all the needed libraries (check requirements.txt)

Step 2: provide your openai API key in the code

openai.api_key= "insert key here"

Step 3: Run the code and write the following command in the terminal

- For latest stable version (systemenglish):
  streamlit run systemenglish.py

- For latest release with multilanguage support (finalapp):
  streamlit run finalapp.py

4. Versions and updates:

- 26 October 2024: version 1.1.1 | cv analysis - CV extraction with initial recommendation
- 4 November 2024: version 1.1.2 | advanced job analysis - Added role analysis and skill match
- 12 November 2024 : version 1.1.3 | feedback and goal tracking - Added user feedback and SMART goal generation
- 27 November 2024: version 1.2.1 | User interface - Integrated backend program into UI
- 7 December 2024: version 1.2.2 | system English - Fixed bugs and added loop for feedback
- 10 December 2024: version 1.2.3 | final app - Added multi-language support

The first few versions (1.1) where only developed as a backend and are not released. Versions 1.2 contain both the backend and the frontend UI.

Note that the latest version (1.2.3 | finalapp) contains multi-language support but it is slower and still contain unresolved issues or bugs. If you face problems, use version 1.2.2|systemenglish which is stable and fully tested.

5. Challenges Faced

While developing the app we faced many challenges:

- **Data acquisition and modeling:** initially we wanted to train our own career recommendation model however, we were not able to find dataset and the approach would not give good result. To fix this issue we decided to use an LLM approach instead by using chatgpt through openai API.
- **Consistency between steps:** given that there are many sequential steps from CV features extraction to the goals tracking, and given that these steps are interdependent, we had some issues where a small error in the first generated response would create big problems later. We fixed this issue by adjusting the prompts and specifying the format of the response. We also added function to extract career titles from a text to a list by processing the text and searching for keywords.
- **Input Feedback into UI:** while integrating the feedback system in the UI the feedback was not working properly and had to be inputed through the terminal so we redesigned the feedback function to make it compatible with streamlit UI

6. Acknowledgement

We would like to thank Doctor Ammar Mohanna for his help throughout the project.