SHL recommender app

Akshay P R

April 2025

0.1 Description:

- First the SHL website was scrapped using selenium. The scrapped output was passed through a reprocessing pipeline to format it in the desired dictionary form with relevant metadata such as the assessment description, href, assessment time etc. The extracted data are available at the Github page as pkl files:
  - individual SHL assessment with description.pkl
  - prepacked SHL assessment with description.pkl

  The code written to scrape the website is available in the scrapper folder at Github.
- The data was converted to documents and the search index was created using FAISS. Similarity search on the vector embeddings created by an openai embedding model serves the retriever part of the RAG system.
- RetrievalQAWithSourcesChain from langchain is used to create the RAG system. Through various trail and error a final documentprompt and a questionprompt was created to be used in RAG sytem. The RAG logic is present in the shlbot.py file at Github.
- The Render.com website was used to create the fasapi backend where one can query and retrive the answers in json format. The url is https://shl-recommender-system-be7b.onrender.com/ask. One can check the health of
  the backend by testing https://shl-recommender-system-be7b.onrender.com/health. The api.py file at Github shows the fastapi code.
- Streamlit was used for the frontend of the app. The code is available as app.py file at Github. This code uses the requests module to send and receive data from the fastapi backend.
- The core functionality of the application is in place and working as intended. However, I'd like to note that the current evaluation metric (MAP@3 and mean Recall@3) is not yet at the level I had aimed for.

I'm continuing to iterate on this aspect and would be happy to discuss my approach, the challenges I faced, and how I plan to improve the model's performance further. Thank you for considering my submission despite the delay.