Run csp.py and modelv1.py after installing all packages in requirements.txt

Project Overview

This project aims to develop an AI-driven system to enhance Queen's University's course scheduling and student advising processes. The system integrates three key components:

1. **CSP-Based Room Assignment Module:** Utilizes constraint satisfaction problem (CSP) techniques to optimize room scheduling under multiple constraints:
    a. No room can be assigned to overlapping time slots
    b. No same department, same year courses can overlap
    c. Professor preferences
    d. Etc.
2. **Probabilistic Inference Model:** Predicts student success in courses based on various performance metrics (to be implemented later).
3. **Deep Learning Course Search Engine:** Interprets natural language queries to provide personalized course recommendations.

A front-end interface is also planned to allow users to access these components and view results in a visually appealing manner.

Progress Summary

1. **CSP-Based Room Assignment:**
    a. **Status:** Initial implementation completed and works using test data. Currently experimenting with constraints – like classroom type, number of courses a week (these default to 3), and class duration, but the current implementation has that removed. The main constraints in the current model are:
        i. Each course must be assigned to a room with a capacity greater than or equal to the number of students in the course
        ii. No room can be assigned to overlapping time slots for different courses
        iii. Courses from the same department and year cannot have overlapping time slots. For example, no 100 CISC courses can run at the same time. This is to simulate allowing all mandatory courses to be taken.
        iv. Prof Preference (soft constraint) Prof can put the times and days they don't want to work. These are weighted to simulate prof seniority so older profs can have first pick. This data will be implemented by hand.

b. **Next Steps:** Integrate the module with curated databases to replace the test data. The database still needs to be finalized. Upon successful integration, this component will be considered complete.

2. **Deep Learning-Powered Course Search Engine:**
   a. **Status:** All required data has been scraped from multiple websites and structured into a database containing the information about all courses at Queen's in the Arts and Science and Engineering Faculties. This database can be found in the repository as Final_DB.csv in the deep_learning folder. This database has the attributes ['course_code', 'course_name', 'units', 'description', 'requirements', 'faculty', 'outcomes', 'prerequisites', 'corequisite', 'exclusions', 'recommended', 'department_name', 'year', 'department_code', 'instructor'].

      As well, an initial encoder-decoder Seq2Seq model implementation has been created to test the viability of our idea of training the model to translate natural language search queries to SQL queries that can be used the query the DB we have developed. This model, with extremely limited test data to train on, has shown promise – which can be observed by running the file modelv1.py in the deep_learning folder.

   b. **Next Steps:** The information contained in our database needs to be further refined for ease of querying in SQL. We have begun to make openai api calls to help us extract key words from course descriptions/outcomes, to create attributes thereof such that someone could query "courses about morality", for example. Moreover, there is significant variation in the structure of the 'prerequisite', 'corequisite', 'exclusion' and 'recommended' information, for example the perquisites of some courses are just other course codes, some are course codes and a minimum grade, some are a certain 'Level' of achievement and others require permission from the department/professor. We are working to unify the structure of this information, once again for the ease of querying.

      Once the database is completed, we will extract the unique values from each attribute and create test/train data by subbing those values into different natural language query templates using python scripts. We expect to have a significant number of training data, for the number of question permutations that exist from our 10-15 templates. We will then use that train/test data to train our Seq2Seq model on GoogleColab. Ultimately, the translated SQL queries from natural language will query the DB we have developed and return those results to the user.

3. **Probabilistic Inference Model:**

a. **Status:** All necessary data is collected; however, development on the actual model has not yet commenced. This component will be completed once the CSP and deep learning modules are finalized. We plan to use Pyro and Scikit-learn.

Feedback Request

1. Is the level of detail provided for the two implemented segments adequate, or are there specific areas where we should expand further?
2. Assuming the CSP and Deep Learning segments were completed as planned, what grade would they receive?