vlAggiando

Un agente intelligente per consigliare mete di viaggio

Abilities

- Generazione di consigli di destinazione: L'agente è in grado di generare consigli di destinazione personalizzati in base alle preferenze dell'utente e ai criteri specificati, utilizzando un albero decisionale addestrato sui dati di viaggio.
- Gestione delle preferenze dell'utente: L'agente può aggiungere, rimuovere o modificare le preferenze dell'utente riguardo alle città che desidera visitare o evitare. Questo viene fatto attraverso le funzioni **addCityPreferences** e **addCityDislike**.
- Calcolo del percorso più breve tra due città: L'agente è in grado di calcolare il percorso più breve (usando solo città che l'utente visiterebbe) tra due città all'interno della matrice delle città utilizzando l'algoritmo di ricerca BFS (Breadth-First Search). Ciò viene eseguito tramite la funzione **cityPath** e **coupleCity**.

Goal

- Stampare tre mete che sono compatibili con i gusti dell’utente e il percorso più breve per arrivare in quella scelta

Prior knowledge

- Una matrice delle città (**cityM**) e un dizionario di associazione (**cityD**) che forniscono informazioni sulla disposizione delle città nella matrice e le corrispondenze tra i nomi delle città e le coordinate nella matrice.
- Il file "**travel_data.csv**" contiene le informazioni sulle diverse destinazioni di viaggio, come il nome della città, le attività disponibili, il budget richiesto, la cultura predominante e la temperatura. Queste informazioni sono utilizzate per valutare le preferenze dell'utente e generare consigli basati sui criteri selezionati.
- Il file "**user_data.csv**" contiene le preferenze dell'utente, come le città che l'utente desidera visitare e quelle che non desidera visitare. Queste preferenze vengono utilizzate per addestrare un albero decisionale (decision tree) che aiuta a generare consigli personalizzati per l'utente.

Stimuli

Input dell’utente:

- Aggiungere/rimuovere una città da quelle che visiterebbe
- Aggiungere/rimuovere una città da quelle che non visiterebbe
- Richiedere all’agente di consigliare tre mete e successivamente come raggiungere la meta che l’utente preferisce

Modifiche ai file CSV "**travel_data.csv**" o "**user_data.csv**":

- Quando viene aggiunta o rimossa una città preferita o non preferita, l'agente può ricevere uno stimolo per aggiornare i dati e regolare i consigli di conseguenza.

Reasoning

- Generazione dei consigli delle città (funzione **vlAggiando**): Qui il ragionamento viene utilizzato per addestrare un albero decisionale (decision tree) basato sui dati presenti nel dataframe userdata. L'albero decisionale viene utilizzato per predire le città che l'utente visiterebbe in base alle sue preferenze. Il ragionamento avviene durante l'addestramento del decision tree e durante la predizione delle città da consigliare.
- Aggiunta/Rimozione di città preferite/non preferite (funzione **addCityPreferences/ addCityDislike**): Qui il ragionamento viene utilizzato per gestire le preferenze dell'utente riguardo alle città da visitare. L'agente verifica se la città è già presente nel dataframe **userdata** e, in base alla situazione, aggiunge, rimuove o modifica il valore di "Visiterei" per la città selezionata.
- Calcolo del percorso più breve tra due città (funzione **cityPath** e **coupleCity**): Qui il ragionamento viene utilizzato per trovare il percorso più breve tra due città all'interno della matrice delle città **cityM** (utilizzando solo città che l'utente visiterebbe). L'agente utilizza l'algoritmo di ricerca BFS (Breadth-First Search) per esplorare la matrice e trovare il percorso desiderato.

Search

- La funzione **cityPath** implementa l'algoritmo di ricerca BFS. Prende in input la matrice delle città **cityM**, le coordinate della città di partenza e della città di destinazione, e restituisce il percorso più breve tra le due città, usando solo città che l'utente visiterebbe, come una lista di coordinate.
  La funzione utilizza una coda (deque) per eseguire la ricerca in ampiezza, esplorando i pixel adiacenti in modo sistematico e mantenendo traccia del percorso.

Planning

- La funzione **cityPath** utilizza l'algoritmo di ricerca Breadth-First Search (BFS) per trovare il percorso più breve tra due città nella matrice **cityM**. Questo percorso rappresenta il "planning" dell'agente per raggiungere la città di destinazione dall'attuale città di partenza.

Learning

Decision Tree Classifier:

- Il componente di learning è un Decision Tree Classifier implementato utilizzando la libreria scikit-learn. Il modello è addestrato sui dati di preferenza dell'utente riguardanti le destinazioni di viaggio (user_data.csv).
  Le caratteristiche considerate includono attività, temperatura, budget, e cultura.
  La caratteristica da predirre è se l'utente visiterebbe o meno la meta (Visiterei).

Constraint satisfaction

- Il constraint satisfaction è rappresentatato dalla matrice **cityM**. Questa matrice rappresenta le restrizioni o vincoli che l'agente deve rispettare quando suggerisce le destinazioni di viaggio.
  La matrice **cityM** è una matrice 7x7 in cui ogni cella rappresenta una coppia di città (pixel della matrice) e contiene un valore binario: 1 se l'agente non può viaggiare direttamente da una città all'altra, 0 se il viaggio è possibile.