FOL - Tour offer

سیدمحمدحسین هاشمی ۴۰۲۲۳۶۳۱۴۳

دی ۱۴۰۲

فهرست مطالب

۱ TODO 1 – آماده سازی پایگاه دانش

۱.۱ شهرها و امکانات با روش flat fact وارد پایگاه دانش می‌شود .

FOL.py

```python
import csv
from pyswip import Prolog

prolog = Prolog()


def loadData():
    prolog.retractall("destination(_)")
    prolog.retractall("country(_, _)")
    prolog.retractall("region(_, _)")
    prolog.retractall("climate(_, _)")
    prolog.retractall("budget(_, _)")
    prolog.retractall("activity(_, _)")
    prolog.retractall("demographics(_, _)")
    prolog.retractall("duration(_, _)")
    prolog.retractall("cuisine(_, _)")
    prolog.retractall("history(_, _)")
    prolog.retractall("natural_wonder(_, _)")
    prolog.retractall("accommodation(_, _)")
    prolog.retractall("language(_, _)")

    dataset = open('Destinations.csv')
    skip = True
    for i in csv.reader(dataset):
        if skip:
            skip = False
            continue
        destination = i[0].replace("'", '"').replace(' ', '-').lower()
        prolog.assertz("destination('" + destination + "')")
        prolog.assertz("country('" + destination + "', '" + i[1].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("region('" + destination + "', '" + i[2].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("climate('" + destination + "', '" + i[3].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("budget('" + destination + "', '" + i[4].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("activity('" + destination + "', '" + i[5].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz(
            "demographics('" + destination + "', '" + i[6].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("duration('" + destination + "', '" + i[7].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("cuisine('" + destination + "', '" + i[8].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("history('" + destination + "', '" + i[9].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz(
            "natural_wonder('" + destination + "', '" + i[10].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz(
            "accommodation('" + destination + "', '" + i[11].replace("'", '"').replace(' ', '-').lower() + "')")
        prolog.assertz("language('" + destination + "', '" + i[12].replace("'", '"').replace(' ', '-').lower() + "')")


loadData()


def FlatFactQuery(factor, parameter, city='City'):
    output = set()
    for city in list(prolog.query(factor + "(" + city + ", " + parameter + ")")):
        output.add(city['City'])
    return output


if __name__ == '__main__':
    print(FlatFactQuery("budget", "'Low'"))
```

۲.۱ اتصالات شهرها ۳ مرحله بررسی و با ویژگی **connected** مورد بررسی هستند.

FOL.py

```python
import csv
from pyswip import Prolog

prolog = Prolog()


def loadData():
    prolog.retractall("directly_connected(_, _)")
    prolog.retractall("connected(_, _)")

    dataset = open('../Datasets/Adjacency_matrix.csv')
    cities = []
    currentIndex = 1
    for i in csv.reader(dataset):
        if len(cities) == 0:
            cities = i
            continue
        currentCity = i[0].replace("'", '"').replace(' ', '-').lower()
        for j in range(1, len(i)):
            if currentCity != cities[j] and i[j] == '1':
                prolog.assertz("directly_connected('" + currentCity + "', '" + cities[j].replace("'", '"') + "')")
        currentIndex += 1
    prolog.assertz("connected(X, Y) :- directly_connected(X, Y)")
    prolog.assertz("connected(X, Y) :- directly_connected(Y, X)")


loadData()


def GraphQuery(factor, X, Y=None, isBool=False):
    if isBool:
        return bool(prolog.query(factor + "(" + X + ", " + Y + ")"))
    output = []
    for city in prolog.query(factor + "(" + X + ", Y)"):
        output.append(city['Y'])
    return output


if __name__ == "__main__":
    print(GraphQuery("directly_connected", "'mexico-city'"))
    # print(GraphQuery("connectedL3", "'Ottawa'", "'tokyo'", True))
```

۲ TODO 2 - یافتن کلیدهای انواع ویژگی‌ها

FOL.py

```python
import csv

destination = set()
country = set()
region = set()
    climate = set()
    budget = set()
    activity = set()
    demographics = set()
    duration = set()
    cuisine = set()
    history = set()
    natural_wonder = set()
    accommodation = set()
    language = set()


    def loadData():
        dataset = open('Destinations.csv')
        skip = True

        for row in csv.reader(dataset):
            if skip:
                skip = False
                continue
            destination.add(row[0].replace(' ', '-').lower())
            country.add(row[1].replace(' ', '-').lower())
            region.add(row[2].replace(' ', '-').lower())
            climate.add(row[3].replace(' ', '-').lower())
            budget.add(row[4].replace(' ', '-').lower())
            activity.add(row[5].replace(' ', '-').lower())
            demographics.add(row[6].replace(' ', '-').lower())
            duration.add(row[7].replace(' ', '-').lower())
            cuisine.add(row[8].replace(' ', '-').lower())
            history.add(row[9].replace(' ', '-').lower())
            natural_wonder.add(row[10].replace(' ', '-').lower())
            accommodation.add(row[11].replace(' ', '-').lower())
            language.add(row[12].replace(' ', '-').lower())


    loadData()

    if __name__ == "__main__":
        print("destination: ", destination)
        print("country: ", country)
        print("region: ", region)
        print("climate: ", climate)
        print("budget: ", budget)
        print("activity: ", activity)
        print("demographics: ", demographics)
        print("duration: ", duration)
        print("cuisine: ", cuisine)
        print("history: ", history)
        print("natural_wonder: ", natural_wonder)
        print("accommodation: ", accommodation)
        print("language: ", language)
```

۳ TODO 3 - دریافت کلمات کلیدی از جمله ورودی

FOL.py

```python
from keyFeature import *


def keySearch(userInput):
    userInput = [x.lower() for x in userInput.replace(".", '').split(' ') if x != '']

    destinationKey = []
    countryKey = []
    regionKey = []
    climateKey = []
    budgetKey = []
    activityKey = []
    demographicsKey = []
    durationKey = []
    cuisineKey = []
    historyKey = []
    natural_wonderKey = []
    accommodationKey = []
    languageKey = []

    for currentWord in userInput:
        if currentWord in destination:
            destinationKey.append(currentWord)
        if currentWord in country:
            countryKey.append(currentWord)
        if currentWord in region:
            regionKey.append(currentWord)
        if currentWord in climate:
            climateKey.append(currentWord)
        if currentWord in budget:
            budgetKey.append(currentWord)
        if currentWord in activity:
            activityKey.append(currentWord)
        if currentWord in demographics:
            demographicsKey.append(currentWord)
        if currentWord in duration:
            durationKey.append(currentWord)
        if currentWord in cuisine:
            cuisineKey.append(currentWord)
        if currentWord in history:
            historyKey.append(currentWord)
        if currentWord in natural_wonder:
            natural_wonderKey.append(currentWord)
        if currentWord in accommodation:
            accommodationKey.append(currentWord)
        if currentWord in language:
            languageKey.append(currentWord)

    return destinationKey, countryKey, regionKey, climateKey, budgetKey, activityKey, demographicsKey, durationKey, \
           cuisineKey, historyKey, natural_wonderKey, accommodationKey, languageKey
```

۴ TODO 4 - دریافت لیست شهرها بر اساس ویژگی‌ها

FOL.py

```
locations, inputText = self.extract_locations(text)

output = "Flat Fact"
output += "\nInput text: " + inputText
output += "\nkey features:" + str(locations)

results = findCities(locations)

output += "\nfunded locations: " + str(results)

finalResults = self.check_connections(results)
output += "\n finale results: " + str(finalResults)

output += "\n------------------\n"

print(output)
file = open('output.txt', 'a')
file.write(output + '\n')
file.close()
```

در این قسمت تابع findCities فراخوانی و همچنین محتوای فایل outputs آماده می‌شود.

FOL.py

```
from FlatFact import FlatFactQuery

features = ["destination", "country", "region", "climate", "budget", "activity", "demographics", "duration", "cuisine",
            "history", "natural_wonder", "accommodation", "language"]


def findCities(keyFeatures):
    output = set(keyFeatures[0])
    for feature in range(1, len(keyFeatures)):
        for option in keyFeatures[feature]:
            output = output.intersection(FlatFactQuery(features[feature], "'" + option + "'")) if len(
                output) > 0 else FlatFactQuery(features[feature], "'" + option + "'")
    return list(output)


def findCitiesUnion(keyFeatures):
    output = set(keyFeatures[0])
    for feature in range(1, len(keyFeatures)):
        for option in keyFeatures[feature]:
            output = output.union(FlatFactQuery(features[feature], "'" + option + "'")) if len(
                output) > 0 else FlatFactQuery(features[feature], "'" + option + "'")
    return list(output)
```

این تابع با فراخوانی FlatFactQuery و اشتراک گرفتن خروجی آن با خروجی سایر ویژگی‌ها شهرهای مطابق را پیدا می‌کند.

۵ TODO 5 - بررسی اتصال شهرها

FOL.py

```
import graph


def checkConnection(cities):
    calc = {x: dict() for x in cities}
    for k, v in calc.items():
        for i in graph.GraphQuery('connected', "'" + k + "'"):
            if i != k:
                calc[k][i] = list()
                for j in graph.GraphQuery('connected', "'" + i + "'"):
                    if j != i and k != j:
                        calc[k][i].append(j)

    outputs = calculate(cities, calc)

    if len([len(x) for x in outputs if x[-1] in cities]):
        print('Tours:')
        for i in outputs:
            print('->'.join(i))

        maxL = max([len(x) for x in outputs if x[-1] in cities])
        bTour = list()
        for i in outputs:
            if len(i) == maxL:
                bTour = i
                break
        print('\nBest Tour: ', '->'.join(bTour))
        return list(set(bTour))
    else:
        return -1
def calculate(cities, calc):
    print('1st query: ', cities)
    print('Graphs:')
    for k1, v1 in calc.items():
        print(k1)
        for k2, v2 in v1.items():
            print('\t', k2, '\t', v2)

    outputs = []
    for k1, v1 in calc.items():
        for k2, v2 in v1.items():
            outputs.append([k1, k2])
            for city in v2:
                if city in cities:
                    outputs.append([k1, k2, city])

    length = len(outputs)
    for i in range(length):
        for j in range(length):
            if outputs[i][-1] == outputs[j][0] and outputs[i][0] != outputs[j][-1]:
                outputs.append(outputs[i] + outputs[j][1:])
    return outputs
```

این تابع براساس پایگاه دانش شهرهای متصل به یکدیگر را برمی‌گرداند.

۶ TODO 6 - بررسی تعداد و نمایش خروجی

FOL.py

```
if 6 > len(finalResults) > 1:
    tkinter.messagebox.showinfo(title='Please wait ...',
                                message='Please wait ... marking the tour route in map.')
    print('start mapping ------')
    self.mark_locations(finalResults)
    tkinter.messagebox.showinfo(title='Successful',
                                message='Successfully marked. now you can see tour route in map.')
    print('down -------\n\n\n')
else:
    tkinter.messagebox.showerror(title='Error', message='Information is not enough for specific destinations.')
    print('Information is not enough for specific destinations.')
    self.adderrorinfo('Information is not enough for specific destinations.')
```

در این قسمت در صورتی که تعداد شهرها بین ۱ تا ۶ شهر باشد آن را روی نقشه چاپ وگرنه خطا برمی‌گرداند.