

گزارش پروژه پنجم

پروژه پیشنهاد تور گردشگری

درس: مبانی و کاربردهای هوش مصنوعی

استاد راهنما: دکتر حسین کارشناس نجف آبادی

اعضای گروه 2 :

علی‌اکبر احراری- 4003613001

مهرآذین مرزوق- 4003613055

پاییز 1402

فهرست مطالب

[**گزارش کار الگوریتم** 3](#_Toc156699064)

[بخش اول 3](#_Toc156699065)

[بخش دوم 4](#_Toc156699066)

[بخش سوم 5](#_Toc156699067)

[بخش چهارم 6](#_Toc156699068)

[بخش پنجم 7](#_Toc156699069)

[بخش ششم 8](#_Toc156699070)

[کتابخانه‌های مورد استفاده 9](#_Toc156699071)

[منابع 9](#_Toc156699072)

**گزارش کار الگوریتم**

## بخش اول

# *TODO 1: read destinations' descriptions from Destinations.csv and add them to the prolog knowledge base*################################################################################################  
prolog = Prolog()  
df = pd.read\_csv('Destinations.csv')  
df = df.apply(lambda s: s.str.lower() if s.dtype == 'object' else s)  
df\_size = df.shape[0] # Use number of rows  
prolog.retractall("destination(\_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)")  
  
for i in range(df\_size):  
 fact = f"destination(\"{df['Destinations'][i]}\", \'{df['country'][i]}\', \'{df['region'][i]}\'," \  
 f" \'{df['Climate'][i]}\', \'{df['Budget'][i]}\', \'{df['Activity'][i]}\', \'{df['Demographics'][i]}\'," \  
 f" \'{df['Duration'][i]}\', \'{df['Cuisine'][i]}\', \'{df['History'][i]}\', \'{df['Natural Wonder'][i]}\'," \  
 f" \'{df['Accommodation'][i]}\', \'{df['Language'][i]}\')"  
 prolog.assertz(fact)

در بخش نخست به خواندن دیتاست Destinations.csv و اضافه‌کردن اطلاعات آن به پایگاه دانش خود پرداخته شده‌ است. برای استخراج دانش مدنظر از روش یکپارچه استفاده شده است. این روش به دلیل ساختار ساده و مستقیم خود، و همچنین کارآمدی بالا در پایگاه دانش کوچک ما بسیار مناسب است.

در ابتدا، اطلاعات موجود در دیتافریم را برای استفاده راحت تر به شکل lower-case در آوردیم.

سپس تمام حقایق موجود در پایگاه دانش خود را پاک کردیم تا از تکرار اطلاعات جلوگیری کنیم. سپس به هر مقصد گردشگری یک حقیقت destination با تمام ویژگی‌های مربوطه را اضافه کردیم.

## بخش دوم

# *TODO 2: extract unique features from the Destinations.csv and save them in a dictionary*################################################################################################  
unique\_attributes = {'country': [item.lower() for item in df['country'].unique().tolist()],  
 'region': [item.lower() for item in df['region'].unique().tolist()],  
 'climate': [item.lower() for item in df['Climate'].unique().tolist()],  
 'budget': [item.lower() for item in df['Budget'].unique().tolist()],  
 'activity': [item.lower() for item in df['Activity'].unique().tolist()],  
 'demographics': [item.lower() for item in df['Demographics'].unique().tolist()],  
 'duration': [item.lower() for item in df['Duration'].unique().tolist()],  
 'cuisine': [item.lower() for item in df['Cuisine'].unique().tolist()],  
 'history': [item.lower() for item in df['History'].unique().tolist()],  
 'natural wonder': [item.lower() for item in df['Natural Wonder'].unique().tolist()],  
 'accommodation': [item.lower() for item in df['Accommodation'].unique().tolist()],  
 'language': [item.lower() for item in df['Language'].unique().tolist()]}

این بخش مربوط به جداسازی ویژگی‌های خاص می‌باشد. در اینجا، ویژگی‌های خاص را برای استفاده از کلمه‌های مهم و حساس وارد شده توسط کاربر استخراج و در یک دیکشنری ذخیره می‌کنیم.

## بخش سوم

def extract\_locations(self, text):  
 # *TODO 3: extract key features from user's description of destinations* ################################################################################################  
 # Convert the text to lowercase and split it into words  
 text = text.lower()  
 # Define the separators  
 separators = ["!", "@", "#", "$", "%", "^", "&", "\*", "(", ")", "-", "+", "=",  
 "~", "`", "{", "}", "[", "]", ";", ":", "'", "\"", "/", ".", ",",  
 " ", "\t", "\n"]  
  
 # Start with the original text  
 split\_text = [text]  
 # Apply the split method for each separator  
 for sep in separators:  
 split\_text = [substr.split(sep) for substr in split\_text]  
 split\_text = [item for sublist in split\_text for item in sublist]  
  
 two\_word\_combinations = [" ".join(pair) for pair in zip(split\_text, split\_text[1:])]  
  
 # Create a dictionary to store the important words found in the text  
 important\_words\_found = {'country': [], 'region': [], 'climate': [],  
 'budget': [], 'activity': [], 'demographics': [],  
 'duration': [], 'cuisine': [], 'history': [],  
 'natural wonder': [], 'accommodation': [], 'language': []}  
  
 # Iterate over each text word  
 for word in split\_text:  
 for key in unique\_attributes:  
 for value in unique\_attributes[key]:  
 if word == value:  
 important\_words\_found[key].append(word)  
 for word in two\_word\_combinations:  
 for key in unique\_attributes:  
 for value in unique\_attributes[key]:  
 if word == value:  
 important\_words\_found[key].append(word)  
  
 return important\_words\_found

برای درک کلمات ورودی کاربر از این تابع استفاده می‌کنیم. ابتدا تمام کلمات دریافتی را به شکل lower-case در آورده و سپس برای جلوگیری از ورود کارکترها به کلمات، آن‌ها را حذف می‌کنیم. کلمات جدا شده را برای اطمینان با کلمات موجود در دیکشنری ویژگی‌های خاص بدست آمده از قسمت قبل مقایسه می‌کنیم و شکل مناسب آنها را باز می‌گردانیم.

## بخش چهارم

def process\_text(self):  
 # Extract locations from the user's input  
 text = self.text\_area.get("1.0", 'end-1c')  
 locations = self.extract\_locations(text) # Extract locations (you may use a more complex method here)  
  
 # *TODO 4: create the query based on the extracted features of user description* ################################################################################################  
 # Initialize an empty list to store the values  
 values\_list = []  
 for values in locations.values():  
 if values:  
 values\_list.append(f"'{values[0]}'")  
 else:  
 values\_list.append("\_")  
  
 values\_str = ", ".join(values\_list)  
 query\_str = f"destination(Destination, {values\_str})"  
 results = list(prolog.query(query\_str))

تابع process\_text وظیف دریافت و فراخوانی توابع مناسب برای اجرای عملیات اصلی این پروژه را بر عهده دارد. در این بخش، پس از دریافت ورودی از کاربر، به وسیله تابع extract\_locations که قبل‌تر توضیح داده شد ویژگی‌های مدنظر کاربر را استخراج می‌کنیم. سپس یک کوئری برای ایجاد عملیات پرسش از KB خود ایجاد کرده و آن را ارسال می‌کنیم.

## بخش پنجم

def check\_connections(self, results):  
 # *TODO 5: create the knowledgebase of the city and its connected destinations using Adjacency\_matrix.csv* print('result2 ', results)  
 locations = []  
 AdjMatrixDf = pd.read\_csv("Adjacency\_matrix.csv")  
  
 AdjMatrixDf = lower\_case\_df(AdjMatrixDf.copy())  
  
 # Clear any previous Prolog knowledge base  
 prolog.retractall("connected(\_, \_)")  
 prolog.retractall("path(\_, \_)") # Also clear any existing path rules  
  
 # Assert Prolog rules for direct and indirect connections  
 prolog.assertz("path(X, Y) :- connected(X, Y)") # Direct connection  
 prolog.assertz("path(X, Y) :- connected(X, Z), path(Z, Y)") # Indirect connection (recursive)  
  
 for j in range(AdjMatrixDf.shape[0]):  
 for k in range(AdjMatrixDf.shape[1]):  
 if AdjMatrixDf.iloc[j, k] == 1:  
 CityJ = AdjMatrixDf.iloc[0,j]  
 CityK = AdjMatrixDf.iloc[k - 1,0]  
 query = f"connected(\"{CityJ}\", \"{CityK}\")"  
 prolog.assertz(query)  
  
 # Extract city names and create Prolog facts for connected cities  
 for result in results:  
 city = result["City"]  
 locations.append(city)  
  
 connected\_cities = []  
 # Query for paths between cities  
 if locations.\_\_sizeof\_\_() > 1:  
 for start\_city in locations:  
 for end\_city in locations:  
 query = "path({}, {})".format(start\_city, end\_city)  
 answer = prolog.query(query)  
 if start\_city != end\_city and answer:  
 print("Path found between {} and {}".format(start\_city, end\_city))  
 connected\_cities.append(start\_city)  
 connected\_cities.append(end\_city)  
  
 else:  
 print("No path found between {} and {}".format(start\_city, end\_city))  
  
 return locations

وظیفه این تابع بررسی ارتباط بین شهرها بر اساس اطلاعات موجود در یک فایل CSV و یک پایگاه دانش پرولوگ می‌باشد.

اقدامات اصلی:

بارگیری داده‌های اتصال شهرها از فایل CSV به یک DataFrame.

آماده‌سازی قوانین پرولوگ برای اتصالات مستقیم و غیرمستقیم بین شهرها.

پر کردن پایگاه دانش پرولوگ با حقایق مربوط به اتصالات از فایل CSV.

استخراج نام شهرها از ورودی results (حاوی اطلاعات شهری).

استعلام از پایگاه دانش پرولوگ برای یافتن مسیر بین جفت‌های شهر.

چاپ پیام‌هایی که نشان می‌دهد مسیر وجود دارد یا خیر.

برگرداندن لیستی از نام‌های اصلی شهرها

## بخش ششم

# *TODO 6: if the number of destinations is less than 6 mark and connect them* ################################################################################################  
 if len(results) > 5:  
 tkinter.messagebox.showinfo("Too many destinations", "Information is not enough for specific destinations")  
 self.text\_area.delete('1.0', tkinter.END)  
 else:  
 self.mark\_locations(locations)  
  
def mark\_locations(self, locations):  
 *"""Mark extracted locations on the map."""* for address in locations:  
 marker = self.map\_widget.set\_address(address, marker=True)  
 if marker:  
 self.marker\_list.append(marker)  
 self.connect\_marker()  
 self.map\_widget.set\_zoom(1) # Adjust as necessary, 1 is usually the most zoomed out

در انتها، نتایج بدست آمده را روی نقشه به کاربر نشان می‌دهیم. در صورتی که نتایج بدست آمده بیشتر از 5 باشند، یک پیام اخطار مبنی بر کمبود اطلاعات به کاربر نشان داده‌ می‌شود و ورودی مجددا دریافت می‌شود. در غیر اینصورت، تور مناسب به وسیله تابع mark\_locationsروی نقشه نشان داده‌خواهد شد.

# کتابخانه‌های مورد استفاده

import pandas as pd

مورد استفاده در ایجاد و استفاده دیتافریم‌ها

# منابع

Programming in Prolog, William F.Clocksin, Christopher S. Mellish

[prolog - Logic programming in python - Stack Overflow](https://stackoverflow.com/questions/61099308/logic-programming-in-python)

[prolog · PyPI](https://pypi.org/project/prolog/)