پیشنهاد تور گردشگری

در این پروژه، هدف پیاده‌سازی یک سیستم پیشنهاد دهنده تور گردشگری بر اساس متن دریافتی از کاربر است. سیستم در گام‌های مختلف، ویژگی‌های گردشگری شهرها را دریافت می‌کند و براساس دانش ذخیره شده در پایگاه دانش و استنتاج منطقی، شهرهایی که بیشترین تطابق را با توضیحات کاربر دارند برمی‌گرداند و سپس براساس ارتباط این شهرها با یک‌دیگر، یک تور گردشگری پیشنهاد می‌شود. برای پیاده‌سازی این سیستم از زبان‌های برنامه‌نویسی پایتون و پرولوگ استفاده می‌کنیم. در ادامه گام‌های مختلف تمرین توضیح داده شده‌اند.

ساخت پایگاه دانش

به عنوان اولین گام، باید پایگاه دانشی از نقاط گردشگری و ویژگی‌های آن‌ها ساخته شود. به شما دیتاستی با نام `Destinations.csv` داده شده است. توضیحات هر ستون در دیتاست به شرح زیر می‌باشد:

نام ستون	توضیح
Destination	نام شهری که در نهایت شما می‌بایست پیش‌بینی کنید
Country	نام کشوری که شهر پیش‌بینی شده در آن قرار گرفته است
Region	موقعیت منطقه کشور (به عنوان مثال ایتالیا در جنوب اروپا می‌باشد.)
Climate	توصیف کننده آب و هوای مقصد است که می‌تواند یکی از مقادیر: Tropical, Cold, Temperate, Desert باشد.
Budget	بودجه سفر ما که یکی از مقادیر: Low, Medium, High می‌باشد
Activity	نوع فعالیت که ماجراجویانه یا فرهنگی دو گزینه ممکن هستند (Adventure, Cultural)
Demographic	از لحاظ جمعیت شناختی مناسب چه گروهی است (Family-friendly, Solo, Senior)

نام ستون	توضیح
Duration	مدت زمان توصیه شده برای اقامت. (Short, Medium, Long)
Cuisine	نوع غذایی که در مقصد محبوب یا غالب هستند (Asian, European, Middle Eastern)
History	توصیف کننده زمینه تاریخی مقصد (Ancient, Medieval, Modern)
Natural Wonder	نوع منظره‌های طبیعی یا شگفتی‌هایی که در مقصد برجسته هستند (Mountains, Beaches, Forests)
Accommodation	سطح معمولی اقامتگاه موجود یا توصیه شده (Luxury, Mid-range, Budget)
Language	زبان غالب در آن منطقه (English, French, Italian, etc...)

برای ایجاد پایگاه دانش روش‌های متعددی به منظور دریافت حقایق و ایجاد روابط وجود دارد که دو روش را به شکل مختصر توضیح می‌دهیم.

روش اول: یکپارچه

```
from pyswip import Prolog

prolog = Prolog()

prolog.retractall("destination(_, _, _, _, _, _, _, _, _, _, _, _, _, _)")
prolog.assertz("destination('Tokyo', japan, 'East Asia', temperate, high, cultural, solo, long, asian, modern, mountains, luxury,
prolog.assertz("destination('Ottawa', canada, 'North America', cold, medium, adventure, family_friendly, medium, european, modern,
prolog.assertz("destination('Mexico City', mexico, 'North America', temperate, low, cultural, senior, short, latin_american, ancie
prolog.assertz("destination('Rome', italy, 'Southern Europe', temperate, high, cultural, solo, medium, european, ancient, beaches,
prolog.assertz("destination('Brasilia', brazil, 'South America', tropical, low, adventure, family_friendly, long, latin_american,

query = "destination(City,_, _, _, low, _, _, _, _, _, _, _, _)"
results = list(prolog.query(query))
# Print the results
for result in results:
    print(result["City"])  #
```

با استفاده از روشی که در کد بالا نشان داده شده، یک پایگاه دانش در پرولوگ (Prolog) با استفاده از پایتون و کتابخانه PySwi-Prolog ایجاد می‌کنیم. در این روش، هر مقصد گردشگری به عنوان یک حقیقت (fact) در نظر گرفته می‌شود که شامل تمام ویژگی‌های آن مقصد (مانند آب و هوا، بودجه، فعالیت‌ها، جمعیت هدف، طول اقامت و غیره) است.

- ابتدا، تمام حقایق موجود با نام `destination` پاک می‌شوند تا از تکرار اطلاعات جلوگیری شود.
- سپس، برای هر مقصد گردشگری یک حقیقت `destination` با تمام ویژگی‌های مربوطه اضافه می‌شود.
- در نهایت، یک پرسش (query) برای یافتن مقاصدی که بودجه کم دارند اجرا می‌شود و نتایج چاپ می‌شوند.

مزایا و معایب این روش

- ساختار ساده و مستقیم: این روش به طور واضح تمام ویژگی‌های یک مقصد را در یک حقیقت جمع‌آوری می‌کند، که خواندن و فهم آن آسان است.
- پرسش‌های یکپارچه: می‌توان با یک پرسش ساده تمام ویژگی‌های مربوط به یک مقصد را استخراج کرد.
- مناسب برای پایگاه دانش کوچک: برای پایگاه‌های دانش که تعداد محدودی موجودیت و ویژگی دارند، این روش کارآمد است.
- ناکارآمد برای پایگاه دانش بزرگ: در پایگاه‌های دانش با تعداد زیادی موجودیت و ویژگی، این روش ممکن است ناکارآمد و دشوار برای مدیریت باشد.
- تکرار اطلاعات: اگر ویژگی‌های مشابه بین چندین مقصد وجود داشته باشد، این ویژگی‌ها در هر حقیقت تکرار می‌شوند که می‌تواند منجر به افزایش حجم داده‌ها و پیچیدگی شود.
- محدودیت در تعاملات پیچیده: ایجاد روابط پیچیده و استنتاج‌ها بین ویژگی‌های مختلف ممکن است در این ساختار دشوارتر باشد.

در نهایت، انتخاب این روش بستگی به نیازها و محدودیت‌های پروژه شما دارد. برای پروژه‌های کوچک‌تر با تعداد محدودی ویژگی، این روش می‌تواند مفید و کارآمد باشد.

روش دوم: Flat Fact

```
# Clear existing facts to avoid duplicates
prolog.retractall("climate(_,_)")
prolog.retractall("budget(_,_)")
prolog.retractall("activity(_,_)")
prolog.retractall("demographic(_,_)")
prolog.retractall("duration(_,_)")
prolog.retractall("cuisine(_,_)")
prolog.retractall("history(_,_)")
prolog.retractall("natural_wonder(_,_)")
prolog.retractall("accommodation(_,_)")
prolog.retractall("language(_,_)")
prolog.retractall("region(_,_)")
prolog.retractall("my_destination(_)")

# Tokyo
prolog.assertz("my_destination('Tokyo')")
prolog.assertz("region('Tokyo', 'East Asia')")
prolog.assertz("climate('Tokyo', temperate)")
prolog.assertz("budget('Tokyo', high)")
prolog.assertz("activity('Tokyo', cultural)")
prolog.assertz("demographic('Tokyo', solo)")
prolog.assertz("duration('Tokyo', long)")
prolog.assertz("cuisine('Tokyo', asian)")
prolog.assertz("history('Tokyo', modern)")
prolog.assertz("natural_wonder('Tokyo', mountains)")
prolog.assertz("accommodation('Tokyo', luxury)")
prolog.assertz("language('Tokyo', japanese)")
```

```
query = "budget(City, low)"
results = list(prolog.query(query))
for result in results:
    print(result["City"])
```

در روش جدید ایجاد پایگاه دانش (KB) که در کد بالا نمایش داده شده، ما از ساختار "فلت فکت" (Flat Fact) استفاده می‌کنیم. در این روش، به جای ذخیره‌سازی تمام ویژگی‌های یک مقصد در یک حقیقت واحد،

هر ویژگی به صورت جداگانه به عنوان یک حقیقت در پایگاه دانش ثبت می‌شود.

- ابتدا، تمام حقایق موجود برای هر دسته از ویژگی‌ها پاک می‌شوند.
- سپس، برای هر مقصد، حقایق جداگانه‌ای برای هر ویژگی (مانند آب و هوا، بودجه، فعالیت‌ها و غیره) ایجاد می‌شود.
- در نهایت، یک پرسش برای یافتن مقاصدی که بودجه کم دارند اجرا می‌شود و نتایج نشان داده می‌شوند.

مقایسه با روش قبلی

- ساختار: در روش قبلی، تمام ویژگی‌های یک مقصد در یک حقیقت واحد ذخیره می‌شدند، در حالی که در روش جدید، هر ویژگی به صورت جداگانه ذخیره می‌شود.
- انعطاف‌پذیری: روش فلت فکت انعطاف‌پذیری بیشتری برای پرسش‌های مختلف فراهم می‌کند، زیرا می‌توان به راحتی بر اساس ویژگی‌های خاصی پرسش‌ها را مطرح کرد.
- کارایی: در پایگاه دانش‌های بزرگ، روش فلت فکت ممکن است کارآمدتر باشد، زیرا از تکرار اطلاعات جلوگیری می‌کند و مدیریت ویژگی‌های مشابه بین مقاصد مختلف را ساده‌تر می‌کند.
- پیچیدگی: روش فلت فکت ممکن است در مواردی که تعداد ویژگی‌ها زیاد باشد، پیچیدگی بیشتری داشته باشد، زیرا هر ویژگی به صورت جداگانه مدیریت می‌شود.

در مجموع، روش فلت فکت برای پایگاه دانش‌هایی که نیاز به انعطاف‌پذیری بالاتر در پرسش‌ها و کاهش تکرار اطلاعات دارند، مناسب‌تر است، در حالی که روش قبلی برای پایگاه دانش‌های کوچکتر و با ساختار ساده‌تر مناسب‌تر می‌باشد.

ساخت گراف جهت‌دار

پس از دریافت شهرها، شما می‌بایست براساس ماتریس مجاورتی که به شما در دیتاست `Adjacency_matrix.csv` داده می‌شود، ارتباط میان مقاصد را بررسی کرده و پایگاه دانش مربوط بر هر شهر را ایجاد کنید.

```python
from pyswip import Prolog

prolog = Prolog()

def search(query):
    connected = list(prolog.query(query))
    for destination in connected:
        print(destination["X"])
# Clear previous facts and rules from the knowledge base
prolog.retractall("directly_connected(_,_)")
prolog.retractall("connected(_,_)")

# Define one-way connections
prolog.assertz("directly_connected(mexico_city, rome)")
prolog.assertz("directly_connected(brasilia, ottawa)")
prolog.assertz("directly_connected(brasilia, tokyo)")
prolog.assertz("directly_connected(brasilia, rome)")

# Ensure mutual connectivity for direct connections
prolog.assertz("connected(X, Y) :- directly_connected(X, Y)")
prolog.assertz("connected(X, Y) :- directly_connected(Y, X)")

# Recursive definition for connected to find indirect connections
prolog.assertz("connected(X, Y) :- directly_connected(X, Z), connected(Z, Y)")

# Query the knowledge base for destinations connected to Brasilia
query = "connected(mexico_city, X)"
search(query)
```

روش‌های متعددی برای ایجاد گراف وجود دارد. کد بالا روابط بین شهرها را برای نمایش اتصالات (مانند مسیرهای سفر) تعریف کرده و یک تابع برای جستجوی تمام شهرهای متصل به یک شهر مشخص ارائه می‌دهد. یک قاعده برای $connected(X, Y)$ تعریف شده است تا این اتصالات دو طرفه را تفسیر کند. یعنی اگر شهر A به شهر B متصل باشد، آنگاه برعکس آن نیز صادق است. در ادامه یه قاعده برای ارتباط همسایه‌های سطح دو تعریف می‌شود.

به عنوان مثال شهر Mexico-City به Rome متصل است و شهر Brasilia به شهرهای Ottawa, Tokyo و Rome متصل است. بنابراین ارتباط شهر Mexico-city و Brasilia از سطح دو می‌باشد.

خروجی کوئری به شرح زیر می‌باشد:

```
rome
mexico_city
brasilia
```

لازم به ذکر است شما میبایست شهر Brasilia را از لیست خروجی حذف کنید. همچنین در هنگام ساختن پایگاه دانش و کوئری زدن **دقت کنید که همه حروف به شکل کوچک وارد شود**.

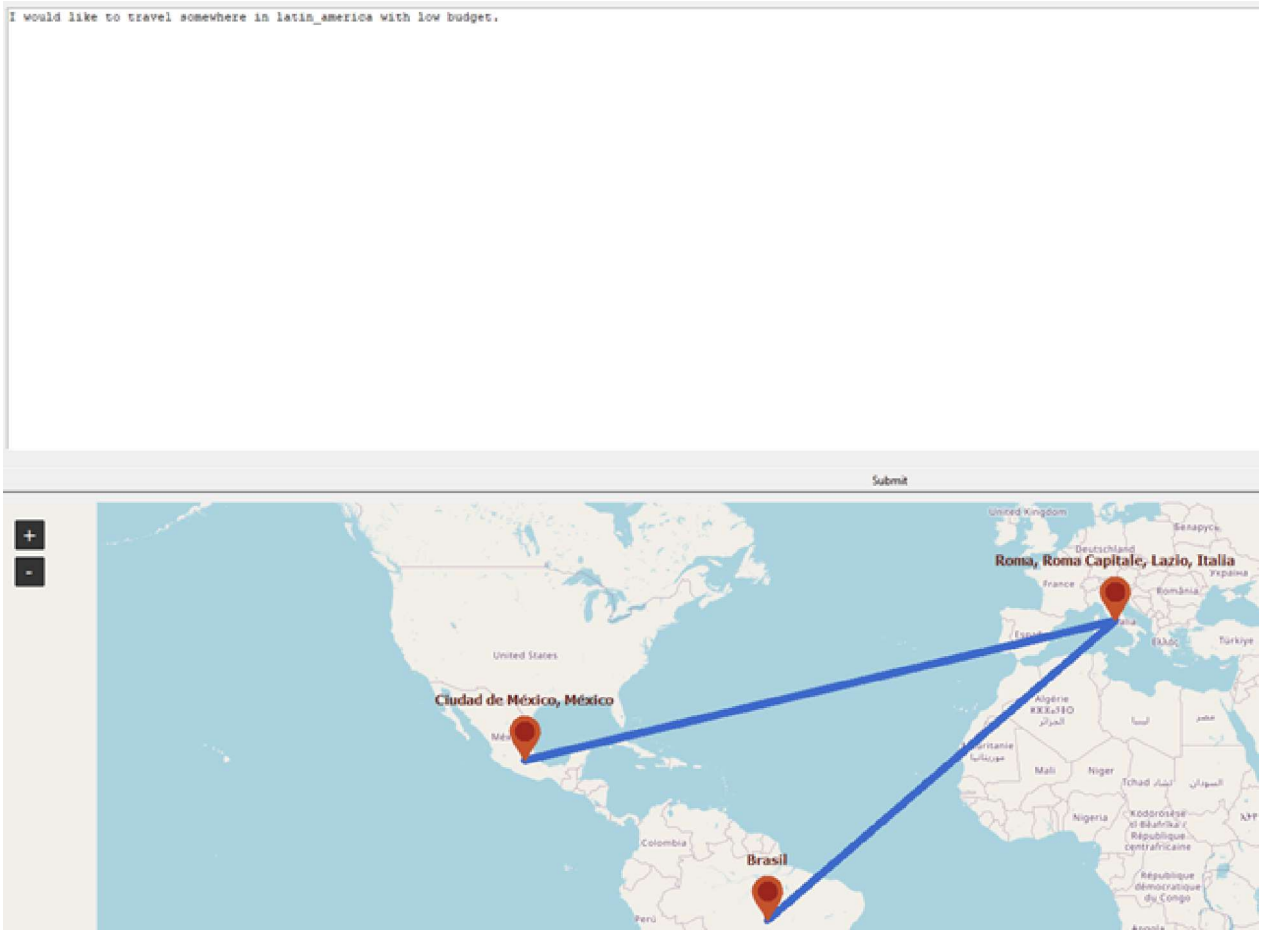
دریافت نظر کاربر، استخراج فکت‌ها و پیشنهاد تور

در این گام باید در تعامل با کاربر یک متن از ورودی دریافت شده و ویژگی‌های مقصد مشخص شود. در ادامه مقاصدی که بیشترین تطابق را با ویژگی‌های بیان شده دارند اعلام می‌شوند و ارتباط این شهرها با یک‌دیگر بررسی می‌شود. دو حالت قابل در نظر گرفتن است:

۱. **تعداد شهرهای تور پیشنهاد شده کمتر از پنج شهر می‌باشد**. در این صورت این مقاصد می‌بایست به عنوان محل سفر در قالب یک تور به کاربر پیشنهاد و در نقشه رابط کاربری نمایش داده شوند. به عنوان مثال اگر درخواست کاربر به شکل زیر باشد: `I would like to travel somewhere in latin_america with low budget.` در این صورت مقاصد پیشنهاد شده به کاربر شامل شهرهای (Mexico City, Rome, Brasilia) می‌باشد که بر روی نقشه رابط کاربری نشان داده می‌شود. دلیل قرارگرفتن Rome در لیست شهرها، پل ارتباطی این شهر با دو شهر دیگر است. به همین دلیل یه مسافر برای سفر به دو شهر در آمریکای جنوبی می‌بایست از Rome عبور کند.

۲. **اگر تور پیشنهادی بیش از پنج مقصد داشته باشد**، آنگاه شما می‌بایست یک پیام اخطار در رابط کاربری `Tkinter` با محتوای (`Information is not enough for specific destinations`) ایجاد کرده و از کاربر توضیحات بیشتری بخواهید. این کار تا زمانی ادامه خواهد یافت که اطلاعات وارد شده توسط کاربر برای مشخص کردن حداقل پنج مقصد کافی باشد.

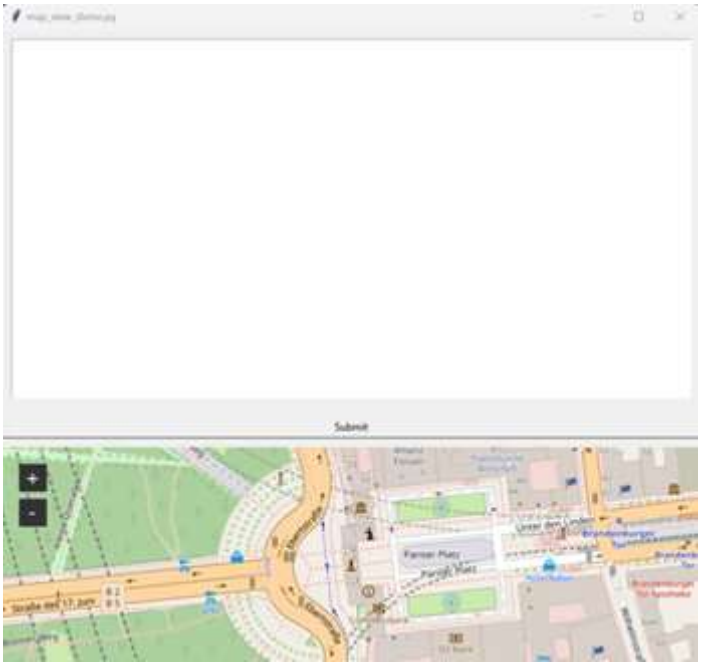
۳. اگر مدل براساس ورودی کاربر نتواند هیچ توری به او پیشنهاد دهد، می‌بایست پیامی مبتنی بر عدم وجود تور نمایش داده و از کاربر ورودی جدیدی بخواهد.



پیاده‌سازی

برای پیاده‌سازی این پروژه، از زبان برنامه‌نویسی پایتون و پرولوگ استفاده خواهید کرد. برای اینکه بتوانید با استفاده از زبان پایتون با پایگاه دانش پرولوگ ارتباط برقرار کنید، باید از رابط PySwip استفاده کنید.

در ادامه، لازم است که زبان پرولوگ را نیز در سیستم خود نصب کنید. دقت کنید که برای اینکه زبان و رابط `PySwip` بتوانند با هم ارتباط برقرار کنند، تنها از نسخه 3.8.4 این زبان که در پوشه مربوط به فایل‌های پروژه قرار گرفته است استفاده کنید. برای انجام تمرین، شما باید فایل `template.py` که در اختیار شما قرار داده شده است را تکمیل کنید (قسمت‌های مشخص شده با TODO را کامل کنید). اجرای این فایل، محیط گرافیکی ساده‌ای به شکل زیر را در اختیار شما قرار می‌دهد:



نحوه کار این محیط به این شکل است که شما در ابتدا می‌بایست پایگاه دانش مربوط به مقاصد را به کمک دیتاست `Destinations.csv` ایجاد کنید. در ادامه متنی از کاربر را دریافت می‌کنید. این متن شامل کلید واژه‌های اصلی مربوط به هر مقصد است. هیچ محدودیتی برای متن دریافتی از کاربر وجود ندارد و وظیفه شما این است که کلیدواژه‌های از قبل تعریف شده را از درون متن استخراج کرده و براساس آن کلیدواژه‌ها کوئری مشخصی را ایجاد کنید. به عنوان مثال: `I want to travel somewhere that everybody speaks` در مثال بالا کلید واژه یافت شده `English` می‌باشد اما از آنجایی که تعداد مقاصد شامل این ویژگی بیشتر از پنج مقصد متصل است، یک پیام اخطار به شرحی که قبلا بیان شده می‌بایست نمایش داده شود. پس از تعیین مقاصد، شما می‌بایست ارتباط آن‌ها با یک‌دیگر را چک کرده و نقاط نهایی را در نقشه به یکدیگر وصل کنید.

آنچه شما باید انجام دهید

۱. در فایل `template.py` ، بخش‌های TODO مشخص شده‌اند. در قدم اول باید خواندن دیتاست `Destinations.csv` و ساخت پایگاه دانش را انجام دهید.

۲. در دومین TODO، باید ویژگی‌های مقاصد را استخراج کنید تا در هنگام تحلیل متن کاربر، از این ویژگی‌ها استفاده شود (راهنمایی: با استخراج ویژگی‌های یکتا از دیتاست و ایجاد یک دیکشنری متناظر با آن به راحتی می‌توان ویژگی‌های تعریف شده را از متن کاربر دریافت کرد).

۳. در سومین و چهارمین TODO، باید تابعی را تکمیل کنید که از متن دریافتی کاربر ویژگی‌های تعریف شده را استخراج کرده، کوئری برنامه پرولوگ آن را نوشته و در پنجمین TODO، می‌بایست ارتباط مقاصد پیدا شده را چک کنید و یک لیست از مقاصد مرتبط را برگردانید.
۴. در ششمین TODO می‌بایست بررسی کنید که اگر تعداد مقاصد بیشتر از پنج باشد، از کاربر اطلاعات بیشتری دریافت کرده و تمام عملیات‌های بیان شده را دوباره اعمال کنید.

- **مهم: شما در این تمرین موظف هستید که همسایه‌های سطح دو را پیدا کنید و یافتن همسایه‌های سطح بالاتر اختیار و همراه با نمره اضافه می‌باشد.**
- دانلود فایل‌های موردنیاز برای پیاده‌سازی