دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پروژه :

پیشنهاد تور گردشگری ، کار با گرولوگ

اعضای گروه :

حمید مهران فر (۴۰۰۳۶۱۳۰۵۸)

رادمهر آقاخانی (۴۰۰۳۶۶۳۰۰۲)

علی کثیری (۴۰۰۳۶۱۳۰۵۱)

استاد : حسین کارشناس

گروه ۱۰

زمستان ۱۴۰۲

تابع readCsv :

این تابع ، ویژگی های موجود در فایل داده شده را میخواند و به پایگاه دانش اضافه می کند . کوئری زدن به روش یکپارچه انجام می شود .

تابع getFeatures :

این تابع تمام ویژگی های موجود در فایل را خوانده و داخل یک دیکشنری قرار می دهد . از انجایی که از set برای این کار استفاده می کند ، داده های تکراری اضافه نمی شوند . این قسمت برای استخراج ویژگی ها از متن مورد استفاده قرار می گیرد .

تابع extractData :

این تابع یک متن گرفته ، ویژگی های موجود در متن را استخراج کرده و انها را در یک ارایه قرار می دهد . به این صورت که هر خانه ی ارایه متناظر ستون های داده است . از انجایی که ۱۲ تا ستون داریم ، پس ارایه ۱۲ عضوی است . اگر ویژگی وجود داشته باشد ، داخل ان خانه قرار میگیرد ، در غیر این صورت کاراکتر خالی قرار می گیرد .

تابع connectCities :

این تابع مقادیر موجود در ماتریس مجاورت را میخواند و انرا در پایگاه دانش ذخیره می کند . در این قسمت ۲ تابع جدید و ۱ مسند برای پرولوگ تعریف می کنیم . یکی از انها برای بررسی ارتباط یک شهر با شهر بعدی مورد استفاده قرار می گیرد (connected) . یکی از انها برای بررسی اتصالات سطح اول و دیگری برای اتصالات سطح دوم مورد استفاده قرار می گیرد (chech_first_connection و check_second_connection) .

نحوه ی خواندن از فایل ، به صورت بالا مثلثی است . یعنی ماتریس به صورت ماتریس بالا مثلثی در نظر گرفته می شود و اگر مثلا بین دو نود ، فقط یک ارتباط باشد ، این ارتباط را دو جهته فرض میکنیم . یعنی ماتریس جهت دار را به بدون جهت تبدیل می کنیم .(دستور connected پرولوگ را هم بین گره ی مبدا و مقصد و هم بین گره ی مقصد و مبدا اجرا می کنیم .)

تابع getConnections :

این تابع یک شهر می گیرد و با استفاده از کوئری های مناسب (chech_first_connection و check_second_connection) تمام شهر های متصل به این شهر را (چه در سطح ۱ و چه در سطح ۲) برمیگرداند . فقط قبل از ان ، این لیست ها را به set تبدیل می کند (برای جلوگیری از تکراری بودن شهرها)

تابع getSecondConnectedCity :

این تابع زمانی صدا زده می شود که یک شهری ، در سطح دو شهر دیگری باشد . درواقع این تابع شهر بین دو شهر که با یک دیگر در سطح دو ارتباط دارند را بر میگرداند .

تابع getSimilarCities :

این تابع ابتدا شهر های متصل به یک شهر را بدست می اورد (اجتماع شهر های سطح ۱ و سطح ۲) . سپس اشتراک این شهر ها را با مجموعه داده شده به تابع را برمیگرداند . این تابع زمانی مورد استفاده قرار می گیرد که بخواهیم تعداد ارتباط بین شهر های با ویژگی های دریافتی از کاربر را بدست بیاوریم .

تابع checkCitiesSimilarity:

این تابع تمام شهر هایی را که با ویژگی های کاربر همخوانی دارد را به عنوان ورودی می گیرد و سپس برای هر شهر ارتباط ان شهر با بقیه ی شهر های این مجموعه را حساب میکند . سپس شروع می کند و تور های پیشنهادی را با شروع از گره ی خودش و گره های همسایه اش ، جستجو می کند . اگر هم هیچ شهری با هیچ شهر دیگر ارتباط نداشت ، none بر میگرداند که بعدا برای تخمین شهرها مورد استفاده قرار می گیرد .

تابع search :

این تابع برای بدست اوردن تور های با بیشترین شهر های با ویژگی های مشترک استفاده می شود . به این صورت که اگر بتوان تا سطح دو ، به شهر های بیشتری متصل شد ، این تابع تور های مربوط به انها را برمیگرداند . یعنی این تابع جستجو را تا جایی که دیگر نتوان شهر با ویژگی های مشترک پیدا کرد ، ادامه می دهد .

تابع getQuery:

این تابع ویژگی های دریافتی از کاربر را میگیرد و شهر های مرتبط به این ویژگی ها را به صورت جداگانه بدست می اورد و اشتراک انها را بر می گرداند . مثلا اگر سه تا ویژگی داشته باشیم ، این تابع یک بار شهر هایی که مرتبط با ویژگی اول هستند را حساب میکند . سپس شهر هایی که مرتبط با ویژگی دوم و به همین ترتیب سوم را محاسبه و اشتراک این ها را برمیگرداند . این تابع برای زمانی مورد استفاده قرار می گیرد که شهر های با ویژگی یکسان به هم ارتباطی نداشته باشند .

تابع checkDifferentQueries:

این تابع برای بررسی شهر های با بیشترین تعداد ویژگی مشترک استفاده می شود . به این صورت که ابتدا شهر های فقط با یک ویژگی مشترک را حساب و به یک لیستی اضافه میکند . سپس شهر های با فقط دو ویژگی مشترک و به همین صورت پیش میرود و همه ی این شهر ها را به لیست اضافه می کند و سپس بر میگرداند .

تابع getApproximateCities:

این تابع زمانی مورد استفاده قرار می گیرد که شهر های با ویژگی مشترک ارتباطی با هم نداشته باشند . این تابع ابتدا با استفاده از تابع getQuery تمام شهر های دارای ویژگی های وارده از سوی کاربر را پیدا می کند . سپس اشتراک این شهر ها با هر کدام از شهر های با ویژگی مشترک را بدست می اورد . شهری که بیشترین ارتباط را داشته باشد ، انتخاب و ۵ تا از شهر هایی که بیشترین ویژگی های وارد شده را دارند ، بر گردانده می شود .

تابع getBestTour:

این تابع تورهای پیدا شده را جستجو میکند و توری که بیشترین تعداد شهر های با ویژگی مشترک را دارا باشد ، برمیگرداند .

تابع check_connections:

در این تابع گراف های شهر ها ساخته می شوند . همچنین جستجو و ساخت تور هم در این تابع انجام می شود .

در قسمتی از این تابع یک سری توابع به زبان پرولوگ نوشته شده است که یکی از انها برای بررسی ارتباط سطح ۱ و یکی از انها برای بررسی ارتباط سطح ۲ است .

منابع :

https://stackoverflow.com/questions/613183/how-do-i-sort-a-dictionary-by-value

https://www.geeksforgeeks.org/python-reversing-list/

https://stackoverflow.com/questions/58301544/how-to-make-a-negative-fact-in-prolog

https://stackoverflow.com/questions/8523608/what-is-the-logical-not-in-prolog

کتابخوانه ها :

Tkinter , tkinter.messagebox , tkintermapview , pyswip , csv