isySUR

it's easy!

Manual

Published January 7, 2015

Contents

1. Introduction to isySUR

Welcome to isySUR (pronounce isy as [’i:zi]), a tool helping you to find the sphere of influence of a space usage rule. For starting the command line version please see section 1.1.1. For using the window version please read section 1.1.2. You can also run the window version on your android device. This is described in section 2.3.3. For more details about the program see section 1.2.

1.1 Quick start

First decide if you like to use the command line version of isySUR or if you prefer a graphical user interface. Please proceed with section 1.1.1 or section 1.1.2 depending on your choice.

1.1.1 isySUR in command line version

Explain how to quick start

1.1.2 isySUR in window version

Explain how to quick start

1.2 What is isySUR?

isySURis a program that calculates the sphere of influence of a space usage rule by its coordinates and data from OpenStreetMap. It was developed within the informatiCup, a programming competition for students organized by the German Informatics Society (Gesellschaft für Informatik e.V.). Our team is called isy and consists of Adriana-Victoria Dreyer, Jacqueline Hemminghaus, Jan Pöppel and Thorsten Schodde, four students of the master study course ‘intelligent systems’ at Bielefeld University.

Using is very easy. You give an input file with the coordinates of the SURs and the sphere of influence is computed. Optionally you can add a configuration file that classifies SURs in those that are applicable only indoor, those that are only outdoor or those that are applicable indoor as well as outdoor (both) to get a better result. The calculated areas are shown on a map (GUI version) and you can choose which ones you want to be displayed. They can be saved into a file to reload them whenever you want to have a look at them.

1.3 Content of isySUR

In the archive you can find the following data (bold printed files will be installed with the install command):

noch up-to-date?

add GUI when finished

MAKE UP TO DATE!

- **isySUR** (directory) - the source files for the isySUR Python package
  - tests (directory) - isySUR was developed using test driven development. In this directory you will find the test files.
- testData (directory) - in here one possible input file named Data.txt is found
  - dataOnlyForTests (directory) - Some of the test files need data that is stored in this directory
- kmlComperator.py - comparison tool for comparing our computed .kml-files with the given .truth.kml-files (the areas of the placemarks), used for development
- manual.pdf - this manual
- README.txt
- **run_isySUR.py** - the Python script to run the command line version of isySUR
- run_tests.py - script that runs all tests in isySUR/tests/, used for development
- setup.py - the setup script to install isySUR
- surConfig.cfg - a configuration file that can optionally be used for computation (see section 2.3)

onieren die pat-
eigentlich noch?

2. Using isySUR

In this chapter the usage of isySUR is described in detail. First you can find installation instructions in section 2.1, detailed usage instruction for both program versions are following in section 2.3.

2.1 Requirements and installation

isySUR is written in Python so you get a Python script, which is the main program, and a package with the tools. For running the command line version you need not much more than basic Python. If you prefer to use a graphical user interface you will have to install some more packages.

2.1.1 Requirements

1. **requirements for the command line version:**
   - Python 2.7
   - requests (HTTP library)
   - internet connection

   To run a Python script you need Python. Python comes with most Linux distributions. For the use on Windows machines you will have to install Python manually[1]. isySUR uses data from OpenStreetMap[2] to calculate the sphere of influence. Therefore an internet connection is required. The data transfer in Python is realised with the requests[3] library. You can easily install it using pip[4], a Python installation tool

   ```
   $ pip install requests
   ```

   or see requests web page[3].
2. **additional requirements for the GUI version:**
   - kivy
   - concurrent.futures

[1]get Python here: `https://www.python.org`
[2]`http://www.openstreetmap.org`
[3]`http://docs.python-requests.org`
[4]`https://pip.pypa.io`

For the graphical user interface kivy is used. Kivy[5] is a cross-platform Python Framework for NUI Development. For installation on Ubuntu we recommend the installation with apt-get as describe on the web page. For use on Windows and starting kivy applications from the command line we recommend to install kivy and pygame (neede for kivy) with the unofficial precompiled binaries[6][7]. On both OS futures can be installed via pip

```
$ pip install futures
```

2.1.2 Installation

You got an archive of type .tar.gz. Please unpack the archive into a directory you want. Because isySUR is a Python script you do not need to install it. It is also possible to install only the dependencies and run isySUR through typing:

```
$ python run_isySUR.py [parameters]
```

For installing the script and tool package browse the directory you chose and install isySUR with help of Python:

```
$ python setup.py install
```

Now you are able to start isySUR anytime through

```
$ run_isySUR.py [parameters]
```

2.2 Input and output

The input of the space usage rules was specified in the programming task as well as the output in .kml format. For more information about the decision to add a configuration file please see section 3.1.3.

2.2.1 Space usage rules

The SUR file is a plain text file. The first line holds the number of space usage rules in the file. Line-serially the SURs follow. Each of this lines starts with the id of the SUR. Then latitude and longitude follow and the last entry is the name or description of the SUR. The entries are comma separated. Decimal separator in latitude and longitude is a point.

Example file:

```
1
0072, 50.9313, 5.39570, smoking="no"
```

2.2.2 Configuration file

The configuration file is a plain text file. In this file one or more blocks starting with '[Indoor]', '[Outdoor]' or '[Both]' are found. After these headlines the names of the SURs follow that belongs to these categories of area of influence.

Example file:

[5]http://kivy.org
[6]kivy: http://www.lfd.uci.edu/%7Egohlke/pythonlibs/#kivy
[7]pygame: http://www.lfd.uci.edu/%7Egohlke/pythonlibs/#pygame

```
[Indoor]
access:age="21+"
camera="no"
[Outdoor]
fishing="no"
littering="no"
[Both]
access:dog="no"
open_fire="no"
```

2.2.3 Keyhole Markup Language

Output files are in .kml format where kml stands for Keyhole Markup Language. For syntax definition and documentation please visit `https://developers.google.com/kml`. For output kml version 2.1 is used.

The GUI version provides also the possibility to read in kml files to display the areas. For this kml version 2.1 as well as 2.2 is supported.

2.3 Command line and GUI version

isySUR comes with two version: One for command line use and another one for users who prefer to use a graphical user interface. For using the command line interface type:

```
$ run_isySUR.py cli [parameters]
```

For using the graphical user interface type:

```
$ run_isySUR.py gui [parameters]
```

Notice that the gui version has more requirements than the command line version. See 2.1.1 for more details.

2.3.1 Command line version

Usage of the command line version:

MAKE UP TO DATE!

```
$ run_isySUR.py [-h] [-c CONFIG] input output
```

Parameters

up to date?

MAKE UP TO DATE!

The parameters of run_isySUR.py (bold arguments are required):

- **input** - Path to the input file containing the SURs. See 2.2 for the required format.
- **output** - Path where to put the output file(s). If this path points to a file, a single output .kml-file containing all computed areas is created. If it points to a directory, one .kml-file for each SUR is created in this directory as well as one .kml-file containing all computed areas.
- -h - If this parameter is given, a short help message containing all parameters is shown.
- -c CONFIG (–config CONFIG) - With this parameter an optional config file with SUR classifications (indoor, outdoor, both) is used for computation. For the format of the file see 2.2.

2.3.2 GUI version

add -c parameter

fill me!

2.3.3 Using android

Of course calculating SURs at home is nice but in general you would like to know a sphere of influence of a space usage rule standing in front of it. This is possible on your android device (with internet connection). In the archive you can find the isySUR.apk that you can install on your device. Just copy it on the device e.g. via USB and enable the device option to install from unknown sources. Browsing to the .apk and clicking on it starts the installation. After that you will find isySUR in your apps and can use it exactly like the GUI version described in section 2.3.2.

of file correct?

3. Technical and implementation details

add some words here

3.1 Development

Korrektur lesen + rework

Code documentation -> pydoc

This project was created using test driven development on the unittest layer. This does not only allow us to keep track of unwated side effects of new functionality, but also helps assigning work load to the different members, because it can be easier to write a failing unittest with a bit of information, that is assigned to someone, than it is to explain all the requirements for some new functionality.

First of all, we all wanted to challenge and expand upon our Python knowledge. Furthermore python is mostly os independent and can even be ported to android for tablet usage.

3.1.1 Data usage

Korrektur lesen + rework

We deliberatly do not use the images in our analysis. This is because they do not give us reliable information about the interesting polygons. The variety of possible signs, including totally unkown ones, (and their placement in/on/around the intended area) make it infeasible to perform robust image recognition or classification. Furthermore the only relevant information we think one could extract from images in special cases refers to an inside/outside classification of the rule. However even this classification would not be reliable and robust enough in the general use case.

One might consider using images and compute vision techniques in the future, when the google StreetView recordings cover virtually everything and these images are available to OSM as well, but until then we do not believe we can extract useful information from the images.

We do use the ruleTypes/Names (e.g. access:21="no") to some minimal extend in the way that we pre classified all the rules in the training set to be applicable indoor,outdoor or both, which we use to narrow down the possible OSM elements around the target coordinates. Unknown rules will always get the "both" classification so that we do not ignore relevant information in those cases. We hope to improve our results (or at least or computational costs) by doing so, but unfortunatly most rules are applicable in indoor as well as outdoor, so this might only be a minor improvement.

Maybe an image of the pipeline?

Our main focus lies upon the coordinate of the SUR. In a bounding box [currently 100m might be reduced further] around the given coordinates, we start by searching for the closest osm relations and ways (we emit nodes for now, since they hardly provide any connectivity information, necessary to be able to construct reasonable polygons). We use a bounding box of limited size because we want to reduce the amount of data that we need to request from OSM and we work on the assumption that these coordinates should always relatively close to the intended area. Unfortunatly some coordinates (at least in the test data) are quite far away from the intended area, which can lead to finding unintented polygons that are closer to the coordinates. Furthermore the OSM data is not complete and even some elements from the test data are not present in OSM, which means, that we have no way of ever finding the intended polygon.

3.1.2 Data structure

Korrektur lesen + rework

In order to be able to work freely with all the data that we require when computing suitable polygons for SURs we have introduced several data structures.

- sur: A wrapper object for the SURs. We have them so that we have to handle the file only once at the beginning.
- osmData: Wrapper for the OSM elements Node, Way and Relation. We convert the xml data we receive from osm to these objects, so that we do not have to work on a xml for computation.
- kmlData: Wrapper objects for a KML and a Placemark. These will be created by our algorithm and can be saved as a XML to a file.

Furthermore, our data structure splits our program into nice modular chunks that can be reused independently in other project if desired.

add Thorstens stuff

3.1.3 The configuration file

said in data usage?

3.2 Evaluation

Berechnen Sie nun die Geltungsbereiche der Space Usage Rules aus Ihrer Umgebung mit Ihrer Implementierung aus der ersten Runde. Diskutieren Sie die Korrektheit und die Präzision der von Ihnen berechneten Lösungen für die Space Usage Rules des Testdatensatzes und denen aus Ihrer Umgebung. Vergleichen Sie dazu die berechneten mit den intuitiv beabsichtigten Geltungsbereichen.

4. Added SURs

For the competition ten new space usage rules should be found.

Add our SURs (maybe with screenshot of intuitive area?)

Todo list

Explain how to quick start . . . 5
Explain how to quick start . . . 5
noch up-to-date? . . . 6
add GUI when finished . . . 6
MAKE UP TO DATE! . . . 6
funktionieren die patterns eigentlich noch? . . . 6
MAKE UP TO DATE! . . . 9
up to date? . . . 9
MAKE UP TO DATE! . . . 9
add -c parameter . . . 10
fill me! . . . 10
name of file correct? . . . 10
add some words here . . . 11
Korrektur lesen + rework . . . 11
Code documentation -> pydoc . . . 11
Korrektur lesen + rework . . . 11
Maybe an image of the pipeline? . . . 12
Korrektur lesen + rework . . . 12
add Thorstens stuff . . . 12
said in data usage? . . . 12
Berechnen Sie nun die Geltungsbereiche der Space Usage Rules aus Ihrer Umgebung mit Ihrer Implementierung aus der ersten Runde. Diskutieren Sie die Korrektheit und die Präzision der von Ihnen berechneten Lösungen für die Space Usage Rules des Testdatensatzes und denen aus Ihrer Umgebung. Vergleichen Sie dazu die berechneten mit den intuitiv beabsichtigten Geltungsbereichen. . . . 12
Add our SURs (maybe with screenshot of intuitive area?) . . . 13