GIS F2E Python – Features to Edge List in Python – Installation and Tutorial

Authors: Rodrigo Marinho Moreira[1] (rodrigommoreira13@gmail.com), Felipe Macena Alves[1] (felipe.amacena@gmail.com), Sybil Derrible[1] (derrible@uic.edu).

Table of Content **Page**

1. Main Function and Features

The main function of this python code is to convert any Geographic Information Systems (GIS) polyline feature (i.e., lines in a shapefile) into a network. The code outputs three files: nodes GIS layer; links GIS layer; and list of links/edges in .csv format with start node and end node coordinates, edge ID and link length. The code essentially cleans raw shapefiles that may have overlapping or split lines to render a clean network, for which line intersections become nodes, and lines connecting nodes become links. Please contact any of the three authors above to report any bugs.

A specific version of the code was produced to process Open Street Map (OSM) road data that possess some elevation information in the form of bridges and tunnels, where lines may be intersecting but no nodes should be created.

This code is an adaptation of the ArcGIS GISF2E developed by A. Karduni, A. Kermanshah, and S. Derrible. Although the process followed in the code is not the same as the process followed in the ArcGIS toolbox, the end result should be identical.

The code reads any shapefile. OSM shapefiles can be downloaded from various sources, including the website GEOFABRIK: https://www.geofabrik.de/data/download.html. The OSM raw data contain a shapefile of links for the selected region, with information about the presence of tunnels or bridges (1 if it applies, 0 otherwise) as well as link names and their osm_id. The links, however, are not correctly split and there is no information about their start/end nodes (Figure 1). The code first finds all intersections whenever two links intersect and cut overlapping links if needed (if the link is not a tunnel or a bridge). From that, the two GIS layers are created in addition to .csv file. Before running the code, you have to determine where these files will be

[1] Complex and Sustainable Urban Networks (CSUN) Laboratory, University of Illinois at Chicago, Chicago, IL

saved in your computer.

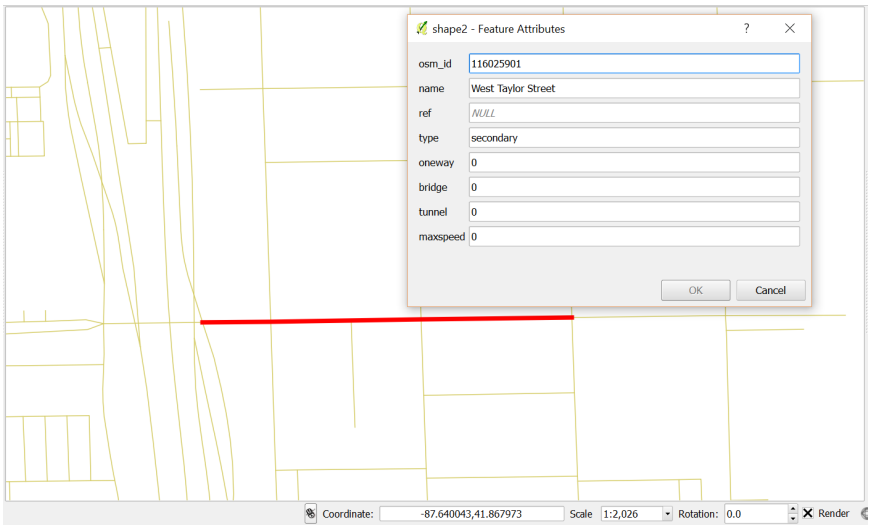

Figure 1 – Link in red selected and its attributes table generated using QGIS. We can see that the link is not correctly cut where it should be.

2. Pre-Requirements

1. The code is written in python. Thus, you will first need to install python 2.7.x (other version will not work), which can be downloaded freely at https://www.python.org. Several different software packages exist that install python along with all of the necessary libraries needed to run the tool, such as Anaconda: https://store.continuum.io/cshop/anaconda/.

2. Six standard python libraries need to be installed to be able to run the code. These libraries may already be installed with your version of python; type "import *library_name*" in your python shell and see whether you receive an error. Moreover, you are recommended to check the version for each library, which can be done typically by first importing the library and then typing "*library_name*.__version__" (note the double underscore) in your python shell (e.g., for pandas, type "import pandas", and then "pandas._version_"). The libraries are:

   a. csv (version 1.0 or later, typically already installed by default with python, if "import csv" returns an error, see: https://pypi.python.org/pypi/csv/1.0)
   b. os (should already be installed)
   c. pyshp (version 1.2.3 or later, https://pypi.python.org/pypi/pyshp)
   d. pyproj (version 1.9.5 or later, https://pypi.python.org/pypi/pyproj)
   e. GDAL (version 2.0.2 or later https://pypi.python.org/pypi/GDAL)
   f. pandas (version 0.14.1 or later, https://pypi.python.org/pypi/pandas/0.14.1)

These libraries themselves may require additional libraries. Follow the correct installation procedure for all of them. Mac users are recommended to use *pip* or *homebrew*, which can be installed from the command prompt. Windows users may find their required libraries at: http://www.lfd.uci.edu/~gohlke/pythonlibs/.

3. Required Files

The dataset to analyze will need to be in .shp format (shapefile), which can be downloaded from the website given in section 1 for OSM. Besides that, you may need to have either ArcGIS or QGIS to view the results outside of python. Note that ArcGIS requires a license. QGIS is open source. Both packages are available at:

- ArcGIS: http://www.esri.com/software/arcgis/arcgis-for-desktop
- QGIS: https://www.qgis.org/en/site/forusers/download.html

4. Running the code for OSM maps

In this section, we run the files provided with the tutorial:

- Chicago_cut for OSM data that was downloaded and cut a smaller shapefile from: http://download.geofabrik.de/north-america/us/illinois.html
- Shape3 for non-OSM data that was downloaded and cut a smaller shapefile from: http://www.mapcruzin.com/download-shapefile/us/connecticut_highway.zip

Step 1: Before running the code, make sure the shapefile that you want to work with is in the GISF2E folder named as *input* (Figure 2). Make sure that there is just one shapefile in this folder.

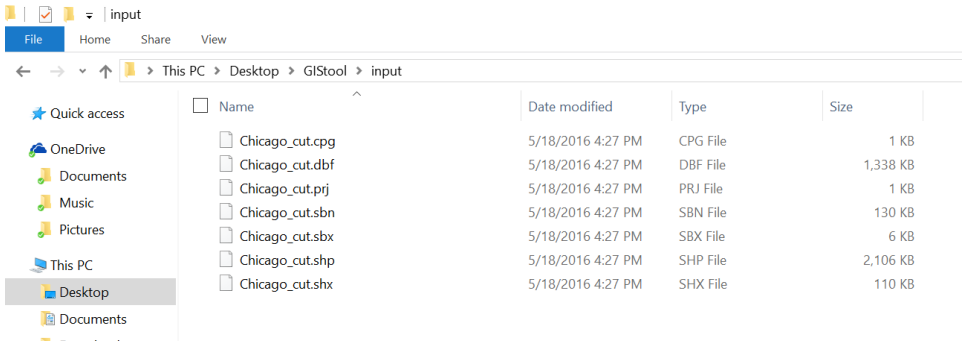

Figure 2 – Correct folder where the shapefile must be located.

Step 2: After you run the code, the first question will be related whether you want to use the shapefile from the folder above or type its location in your computer (Figure 3).

```
Do you want to use input/output folders (1) or type your files
addresses (2)? Option 1 or 2: 1
```

Figure 3 – If you followed *step 1*, just type 1 in this question.

Step 3: For the next question, as in this case the shapefile is from OSM, just type *y* or *yes* or *Y* (Figure 4).

```
Do you want to use input/output folders (1) or type your files
addresses (2)? Option 1 or 2: 1

Is your root file from Open Street Maps (Y/N)? y
```

Figure 4 – Second question made after the code runs.

Step 4: Once you have followed *step 3*, the code will give you updated information as soon as it completes each line below as well as the cumulative time. At the end, you will have the message shown in Figure 5 and you will be able to open the outputs generated.

```
0.0s - Reading and arranging the root file information...

36.3919999599s - Reading and arranging are done!

37.2580001354s - Adding nodes...

59.875s - Nodes are done!

60.0150001049s - Adding links and creating the Edgelist...

994.714999914s - Links and Edgelist are done!

994.717000008s - Saving Edgelist...

996.782999992s - You are all set.
```

Figure 5 – Message shown after the code finishes running.

Step 5: You can look for the shapefiles generated as well as the edgelist csv. file located in the GISF2E folder named as *output* (Figure 6).

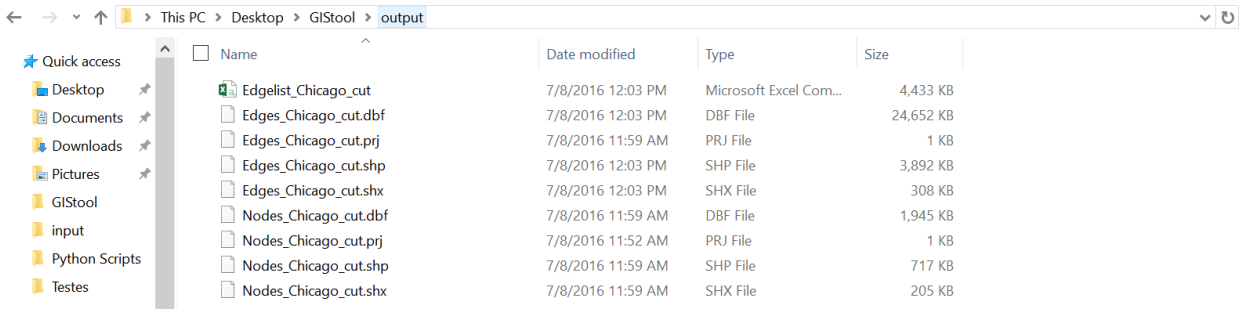

Figure 6 – Correct location of the outputs from the code.

Step 6: To open the shapefiles, you will need to have either QGIS or ArcGIS. Figure 7 shows how the two shapefiles are presented using QGIS.

5. Running the code for non-OSM maps

Step 1: The only difference between running the code for OSM and non-OSM data is that in the *step 3* of running the code for OSM maps, you have to change the answer to *n, no or N*.

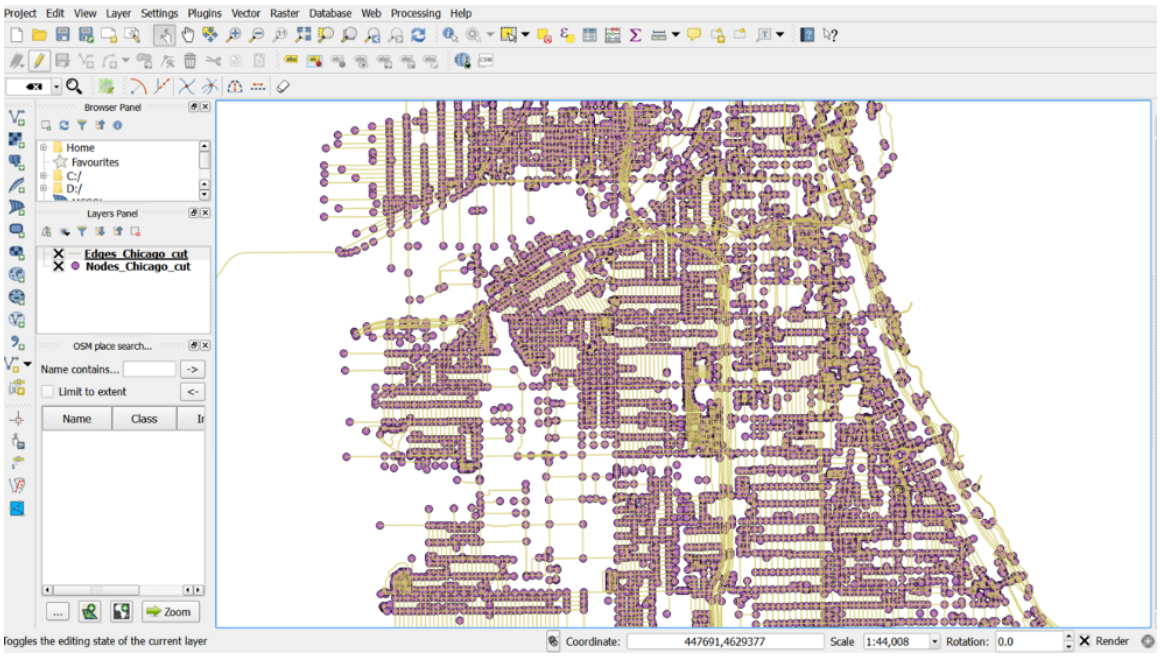

Figure 7 – Edges and Nodes shapefiles as outputs from the raw OSM data.

6. Limitations

For OSM maps, this code uses the English language, so the tests are made based on the words "*bridge*" and "*tunnel*". As a variation for other languages that can be used, you just have to change these words with the ones that appear as a field name on the shapefile. For example, in a Portuguese shapefile, the changes should be: *bridge* → *ponte* and *tunnel* → *túnel*. Also, these adjustments should be made only in the follow lines directly in the code: ***260, 289* and *291.***