Real Speed OSM (PBF augmenter script)

Text Description

The script takes in the OpenStreetMap road network after being converted to .osm. This format is in XML which can be read by the Python library ElementTree.

The first step in the code is to read the input files. One input file is the OSM Network which is read using ElementTree with elements for each way. The other input file is the real speed data which is read from a csv as a dataframe using the pandas library. For Uber, we expect the csv to have the following columns: [osm_way_id, osm_start_node_id, osm_end_node_id, speed_mph_mean]. One challenge that needed to be overcome is the granularity and direction of the real speeds which may be more granular and in the reverse direction compared to the OSM road network.

	osm_way_id	osm_start_node_id	osm_end_node_id	speed_mph_mean
0	4304424	5288682196	26029084	34.935
1	4311275	32927063	2262493188	55.618
2	4311275	259458272	2262534392	53.094
3	4311275	259458354	259458272	54.639
4	4311275	989842042	1096003072	54.593
...	...	...	...	...

We iterate over each way in the OSM tree to perform the following function which checks the way's nodes for a speed from the speed dataframe.

1. Create a list of the node IDs in the way, in the order which they are traversed according to the road network, call it *nodelist*
2. Also create a copy of the *nodelist* in reverse order for speeds found in the opposite direction
3. Query the speeds dataframe for rows whose origin and destination nodes appear in the *nodelist*. This will give us a dataframe of relevant speeds only
4. Iterate over each row of the relevant speeds dataframe and check if the speed reported is for the nodes in the same order/direction of the way or the opposite direction.
5. If the nodes are in the same direction, using another function copy the nodes along with all their tags and attributes to a new synthetic way and a new way ID (which counts sequentially up from a seed given as a variable in the start of the script). The new way gets an extra tag "synthetic = yes" as well as a *maxspeed* tag with its value as the real speed from the speeds dataframe. If there was an existing *maxspeed*, it is overwritten,

otherwise a new *maxspeed* tag is created. Correct sequence nodes are added to a node pair dictionary, *nodepairdf*, for a later step to remove nodes from the original way

6. [Optional] If the nodes in the row are in the reverse direction of the way, check if the reverse speed also has the same way ID, if not then skip the row since the same node pair will be iterated over with the other way. If it does have the same way ID then use the function to copy it to a new way as above but with the reverse node order. (This step is simple for Uber data but too time consuming for Mapbox data. It was skipped)
7. Remove the nodes copied in the correct direction from the original way using the following logic:
    a. First, using the *nodepairdf* created above, delete nodes making up copied segments at the start of the way. We do so by keeping a variable with the start node ID, *startPos*, and another variable with the end node ID, *endPos*, of the original way. If there is a row in the *nodepairdf* whose start node is equivalent to *startPos*, delete all the nodes in the original way from *startPos*, inclusive of *startPos*, until the row's end node, not inclusive of the end node. Set the new *startPos* as the row's end node. Repeat this until all node pairs from the start of the way are removed sequentially. In the event that all the segments of the way are included in the node pair, then all the nodes of the way will be removed.
    b. Next, delete the nodes of the copied segments from the end of the way. If there is a row in the *nodepairdf* whose end node is equivalent to endPos, delete all the nodes from the original way from the row's start node +1 to the row's end node. Set the new endPos as the row's start node. Repeat this until all the node pairs that include the last node of the original way are removed sequentially.
    c. Finally, if there are remaining node pairs that were copied from the original way, they should only include middle segments not including the new start or end of the original way. Before removing the middle segments from the original way, retain a list of the sequences of nodes that do not have a real speed. Using this list, create new ways, one for every continuous sequence of nodes, without a realspeed tag. This is so that ways that were chopped up, leaving gaps in their node sequence can be replaced with ways that have continuous node sequences.
8. Finally, delete the original ways if they contain fewer than 2 nodes.

Pseudocode

Algorithm to make a copy of OSM ways with real speed where it is available. Real speed data may be available for some segments making up only part of a way, not the entire way. These segments may be in the start, middle or end of the way. They may also be in the reverse direction compared to the original OSM way.

Input:

Original OSM road network in PBF format
Real speed data for segments: *SpeedDataframe*

For each *original way in OSM road network* **do**

Get relevant speeds in *SpeedDataFrame: relevantSpeeds*

For each *segment in relevantSpeeds* **do**

Copy the *segment* and its tags from the *original way* to create a synthetic way with a real speed tag

Delete the copied *segments* from the *original way*

end

If {*original way has gaps*} **do** // because middle segments were removed

Create new ways using remaining segments without adding real speed //* The routing algorithm will choose the default speed for the road based on its type *//

end

If {*original way has fewer than 2 nodes*} **do**

Delete original way

end

end