Merge 2 Networks

Description

This algorithm is used to merge identical networks. Emphasis is put on merging edge attributes.

Pros & Cons

This algorithm is very generic, in that you can input networks in any format and it will attempt to merge them based on unique nodes & edges. The merged network can be used to analyze the parent networks side by side. Even if a particular node or edge does not exist in one of the two parent networks, the merged network will contain all nodes and edges from both. The algorithm assumes that the user knows which node attribute has unique values across the entire network and uses that parameter to disambiguate nodes from both the networks. If the user has made a mistake in providing unique node identifiers, unexpected results will occur. This algorithm does not currently accept directed networks, nor does it parse the unique node identifying attribute candidates during the user input stage. In the future, this algorithm will display only those attributes in the drop down box which actually have unique values.

Implementation Details

Once the input networks are validated the algorithm starts. It works as follows,

1. Metadata consisting edge & node schema is captured for both the networks.
2. Based on this data along with (node & edge) attributes that are to be ignored we initiate merging of the network metadata.
3. First consider node schemas of both the input networks.
    a. For each attribute in the first schema do,
        i. Check if the attribute name is present in other schemas. If yes then use the collision resolving prefix provided by the user for the first network to create a unique attribute name in the format - "<PREFIX>_<Current Attribute Name>". Add this new attribute name, old attribute name & the data type to the resolved schema definitions.
        ii. If no, then just add the current attribute name & data type to the resolved schema definitions.
    b. Do the above for the second network as well.
4. Do the above (3.1) for the edge schemas of both the input networks.
5. We will use the unified schema of nodes & edges later on when generating a new network conatining data from both the input networks.
6. We create new maps of unique node identifier to node objects & unique edge identifiers to edge objects so that if duplicate nodes are encountered in other networks they can be promptly merged.
7. Start processing each network file & create new node & edge objects, if needed.
8. Processing of each file assumes the following flow,
    a. When node elements are encountered we first check which is the unique identifier for each node element.
    b. Map of unique identifiers to node objects is checked for the current unique identifier.
    c. If it is present then we use the node object attached to that identifier for further processing.
    d. If not then we create a new node object & create a map entry from unique identifier to newly created node object.
    e. For each incoming node attribute we first check our mapping from old attribute name to new (resolved/unified) attribute & then use that to add attributes to the node object.
    f. When edge elements are encountered we first get the unique edge identifier. Unique edge identifier is constructed using the format - "<SOURCE NODE ID>$<TARGET NODE ID>".
    g. Map of unique identifiers to edge objects is checked for the current unique identifier.
    h. If it is present then we use the edge object attached to that identifier for further processing.
    i. If not then we create a new edge object & create a map entry from unique identifier to newly created edge object.
    j. For each incoming edge attribute we first check our mapping from old attribute name to new (resolved/unified) attribute & then use that to add attributes to the edge object.
    k. After the network assets (nodes & edges) objects are created we use this & final schema to create an output network file.
        i. First the node schema is set.
        ii. Then each node object is traversed and node row is creted in the output file.
        iii. Then edge schema is set.
        iv. Then each edge object is traversed and edge row is creted in the output file.
9. The file thus created is provided to the user through the data manager. It is placed below the second network file.

Usage Hints

The user has to provide 3 initial inputs; the two parent network files storing the information about the network and the attribute name that represents the unique node identifier. The algorithm checks before hand if the two networks have overlapping attribute names for nodes or edges; if so, two further inputs are requested from the user. Both are text inputs which will be used as prefixes when renaming the attributes in the final network. The output will be provided as a .NWB file with a merged network. The following conditions should be met to successfully run the algorithm:

1. Input networks should be undirected.
2. There should be at least 1 common node attribute between the 2 networks. This will be used to disambiguate nodes.
3. Prefix values (if required) should be valid as described below,
    a. Prefix cannot be empty.
    b. Prefix cannot start with a number.
    c. Prefix for different networks cannot be the same.
    d. Prefix cannot be more than 10 characters in length.

Links

- Source Code

Acknowledgments

The aggregate data plugin was authored, implemented, integrated and documented by Chintan Tank.