Summary

Executing sna 6-18.py will produce an author citation network named G1 along with variables listed in the table below. The edges of G1 are stored in a list of lists named edge, where each list <- [Source ID, Destination ID, # of Citations, Edge ID].

A one hop network (G2) may also be produced by running function onehop and passing the returned object to function onehopnetwork. All nodes in network G1 are integer IDs and all google scholar (authors) have respective string IDs. Dictionaries userdict and nodedict will "translate" from an author's string ID to the author's integer ID (vice-versa). The list of lists named edge contains all the edges of the author citation network.

The contents of this repository (6-18-2019) include sna 6-18.py, the workspace of an execution in spyder (Python 2.7), dictionaries to go between string/integer IDs for authors, a dropbox link to a data dump of Breadth-First-Search graph files, author citation network edges, and hopsatnodes. Hopsatnodes contains for each node, the number of connections a hop distance away from that node.

Functions & network objects in sna 6-18.py rely on SNAP (Stanford Network Analysis Project) which requires Python 2.7.x and requirements may vary.
SNAP: http://snap.stanford.edu/

Generated Variables & Objects

Name	Type	Size	Description
DstNId	Int	1	Destination Node
SrcNId	Int	1	Source Node
destnodes	list	794	nodes being cited
directory	str	1	dropbox path
edge	list	1395205	author citation network edges
file	dict	2892	current indexed dataset from jsons
filepaths	list	794	file names of datasets
files	dict	2892	---
i	str	1	index
jsonindex	int	1	index
jsons	list	794	author citation data
key	unicode	1	index
line	str	1	index
node	int	1	index
nodedict	dict	433638	dictionary
nodeid	int	1	number of unique authors

output	str	1	output path
path	str	1	index
reject	int	1	count of non-unique authors when building userdict
row	int	1	index
rowindex	int	1	index
self_cite	list	794	self citations
self_cite_ratio	list	794	self citation ratios
short	unicode	1	current indexed author ID
size	int	1	total number of edges
sizeindex	int	1	index
sum_cite	list	794	total citations of an author
userdict	dict	433638	dictionary
weights	list	2892	weights of currently indexed dataset

Table 1

Functions

self_citation :finds instances of self citations and return a list of lists where each list is of the form: [nodeid, # of self citations]

Inputs

- destnodes: destination nodes, typically user IDs named in the file name of author citation datasets "jsons"
- edges: network edges (list of lists)
- jsons: list containing raw json data
- size: number of edges in network

Function Call

self_cite = self_citation(destnodes,edge,jsons,size)

sum_citation: finds the total number of citations by each author and returns a list of lists where each list is of the form: [nodeid, total # of citations]

Inputs

- destnodes: destination nodes, typically user IDs named in the file name of author citation datasets "jsons"
- edges: network edges (list of lists)
- size: number of edges in network

Function Call

sum_cite = sum_citation(destnodes,edge,size)

self_citation_ratio: returns the self citaiton ratio of each author in a list of lsits where each list is of the form: [userid, self citation ratio]

Inputs

- nodedict: dictionary of node IDs to user IDs
- self_cite: returned list from self_citation function
- sum_cite: returned list from sum_citation function

Function Call

self_cite_ratio = self_citation_ratio(nodedict, self_cite,sum_cite)

printOutDegV: prints outdegree for each node in the network

Inputs: None

printNumNodes: this function prints the results of how many unique/non-unique authors are added to userdict (userID to nodeID dictionary) after building userdict

sortmatrix: returns a sorted matrix in ascending order by col, uses timsort

Inputs

- matrix: typically a list of lists
- col: column of matrix to sort in ascending order

Function Call

sortedmatrix = sortmatrix(matrix,col)

asscedge : creates an associative array mapping node IDS to edges associated with respective IDs
i.e. node ID 0 -> edge[0:5]
Inputs
- sorteditem: a presorted matrix with edges sorted in asscending order
- col: col number to use for mapping

Function Call
asscSrc = asscedge(edgeSrc,0)

onehop : generates a matrix containing one hop edges
Inputs
edge: matrix with a row composed of: Source node, Destination Node, No. of Citations, Edge ID
userdict: dictionary converting user IDs to integer node IDs
Function Call
*onestep = onehop(edge, userdict)

onehopnetwork : generates a SNAP network object containing one hop edges
Inputs
- onestep: matrix containing onehop edges

function onehopnetwork must accept function onehopnetwork's return value
Function Call
G2 = onehopnetwork(onestep)

getbfs : Returns a directed Breadth-First-Search tree with root at starnode
Inputs
- G1: network
- nodekeys: keys of nodedict -> *nodedict.keys()
- search(optional): if 0 -> generate 1 tree at startnode = nodeid
  if 1-> generate trees for all node ids
- startnode(optional) : if search = 0: choose a root for the bfs tree using startnode
  Do not use if search = 1
- FollowOut(Bool): graph constructed by following outward links
- FollowIn(Bool): graph constructed by following inward links

Function Call
bfstrees = getbfs(G1,nodekeys,search=0,startnode=0,FollowOut=True, FollowIn=False)
  nodekeys = *nodedict.keys()

getnodesathops: generates a dictionary of dictionaries with the outer dictionary being a node key/ user id and the inner dictionary containing the number of nodes at hop distance i where I is an integer.
Inputs

- G1: G1 is a snap network object
- nodedict: dictionary of node IDs (int) to user IDs (string)
- keychoice: 0 for node integer IDs, 1 for user string IDs

Function Call

hopobject = getnodesathops(G1,nodedict)

dictdump : dumps the contents of a dictionary into a json file

Inputs

- item: should be a dictionary
- filename: should be a file name as a string with .json ending

Function Call

temp = dictdump(python_dictionary, 'contents.json')

matrixdump: will dump the contents of a matrix (list of lists) into a csv file

Inputs

- matrix: item should be a list of lists or matrix
- filename: should be a file name as a string with .csv ending

example: temp = matrixdump(list of lists, 'contents.csv')

Note:

- *Functions typically accept generated variables, named in Table 1, as input*
- *Any mention of matrix refers to a list of lists.*
- *Some objects/variables produced by the functions above are not immediately generated and must be called separately.*
- *bfs trees dropbox link: https://www.dropbox.com/sh/54jwmzw5o4vdqfd/AABOXP-W0bw0Ll-XFNR_ui5Ja?dl=0*