**GEPHI TEST CODE**

This code explains how, by using various libraries of python we can create a Graph Visualization of our neural network in the Gephi format(.gexf), **without** having to use the Gephi GUI.

The following script makes use of NeuronList.txt, reads the neural network and creates a graph visualization of the neural net. It works by making use of NLI\_main.py, which converts the neuron data from NeuronList.txt into a Neuron class with one class element for each neuron. Each neuron has various attributes like name, inputs, types of synapse, and weights of synapses. The following code can be used to get an understanding of how Gephi can be used to add nodes and edges through Python code,i.e, without the Gephi GUI. At the end of the code, the script saves two output files of the given neural net, one in the .gexf format which is the standard Gephi format and the other in .png format.

The **networkx** python library is used to make the graph.

The **matplotlib.pyplot** python library is used to display the graph and save it in .png format.

The networkx library gives options to create many types of graphs, which can be chosen by declaring the graph element in the following way:

**g = nx.MultiDiGraph()**

This creates a MultiDirectional Graph element called 'g'. Other types of graphs are: **nx.sedgewick\_maze\_graph() ,nx.complete\_graph() ,nx.tetrahedral\_graph()** and **nx.petersen\_graph()**

**g.add\_node(Node\_Name) :** This is used to create nodes of the graph.

**g.add\_edge(Source, Targer) :** Used to create graph edges.

Various other commands which are commented in the code can be tried by uncommenting and executing the script given below to see their effect.

**CODE**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Jul 4 14:20:15 2013

@author: sidharth

"""

"""

Creates a graph out of chase's data from NLI\_main.py in the class format and saves

as a .gexf file format, which can be imported and drawn by gephi, the data

visualization software.

"""

#Used networkx to make the graph: nodes and edges

#used matplotlib.pyplot to draw graph

import matplotlib.pyplot as plt

import networkx as nx

g = nx.MultiDiGraph()#Create graph object, with multi directed graph

#g = nx.sedgewick\_maze\_graph()

#g = nx.complete\_graph(5)

#g = nx.tetrahedral\_graph()

#g = nx.petersen\_graph()

# Chase's code to organize data into classes

# Reads a list that descripbes neurons and their inputs

# then creates a class that stores that information for

# the respective neuron

class neuron:

Ncnt = 0

def \_\_init\_\_(self, name, inputs, types, weights):

self.name = name

self.inputs = inputs

self. types = types

self. weights = weights

neuron.Ncnt += 1

def display\_attribs(self):

print 'Neuron Name: ',self.name

print 'Inputs: ',self.inputs

print 'Type : ',self.types

print 'Weight: ',self.weights

def get\_file(filename):

array =[]

fp = open(filename,"r")

for line in fp:

array.append(line.split())

return array

fp.close()

def display\_buffer(array):

for line in array:

print line

def get\_names(array):

names = []

for element in array:

for ele in element[0]:

if ele in names:

pass

else:

names.append(ele)

return names

def get\_weights(array):

weightstmp = []

for element in array:

for ele in element[2]:

if ele in weightstmp:

pass

else:

weightstmp.append(ele)

return weightstmp

def get\_in\_wei(name):

i = 0

tmparr = []

tmparr2 = []

tmparr3 = []

fintmparr = []

while i < len(ar):

tmp1 = ar[i][1]

tmp2 = ar[i][0]

tmp3 = ar[i][2]

tmp4 = ar[i][3]

if tmp1 == name:

tmparr.append(tmp2)

tmparr2.append(tmp3)

tmparr3.append(tmp4)

else:

pass

i += 1

fintmparr.append(tmparr)

fintmparr.append(tmparr2)

fintmparr.append(tmparr3)

return fintmparr

#MAIN

ar = get\_file('/home/sidharth/Desktop/NeuronList.txt')

del ar[0]

Neuron = get\_names(ar)

inputs = []

for ele in Neuron:

inputs.append(get\_in\_wei(ele))

tst = []

tst1 = []

tst2 = []

for line in inputs:

tst.append(line[0])

tst1.append(line[1])

tst2.append(line[2])

i = 0

while i < len(Neuron):

Neuron[i] = neuron(Neuron[i],tst[i],tst2[i],tst1[i])

i += 1

for i in Neuron:

i.display\_attribs()

array\_gephi = []

#adding nodes and edges using networkx

for i in range(len(Neuron)):

g.add\_node(Neuron[i].name)

if Neuron[i].inputs:

for k in range(len(Neuron[i].inputs)):

g.add\_edge(Neuron[i].inputs[k],Neuron[i].name)

g.get\_edge\_data(Neuron[i].inputs[k],Neuron[i].name)

#g.add\_edge(Neuron[1].inputs[0],Neuron[1].name)

#g.[A][C]['color'] = 'blue'

#nx.write\_gml(g,"test.gml")

nx.write\_gexf(g, "sid.gexf")#saving as gexf file

nx.draw(g)

plt.savefig("simple\_graph.png")

plt.show()#plots graph

#g=nx.path\_graph(4)

#nx.write\_gexf(g, "sid.gexf", encoding='utf-8', prettyprint=True, version='1.1draft')

#run\_layout(Force\_Atlas, iters = 500)

**Example**

The graph shown is the Graph Visualization created using the above code for the following NeuronList.txt:

**Neuron Target Weight Type**

A B 2 i

B C 4 e

C B 1 e

A C 2 e

A D 3 i

D F 2 e

F E 1 e

E B 4 i

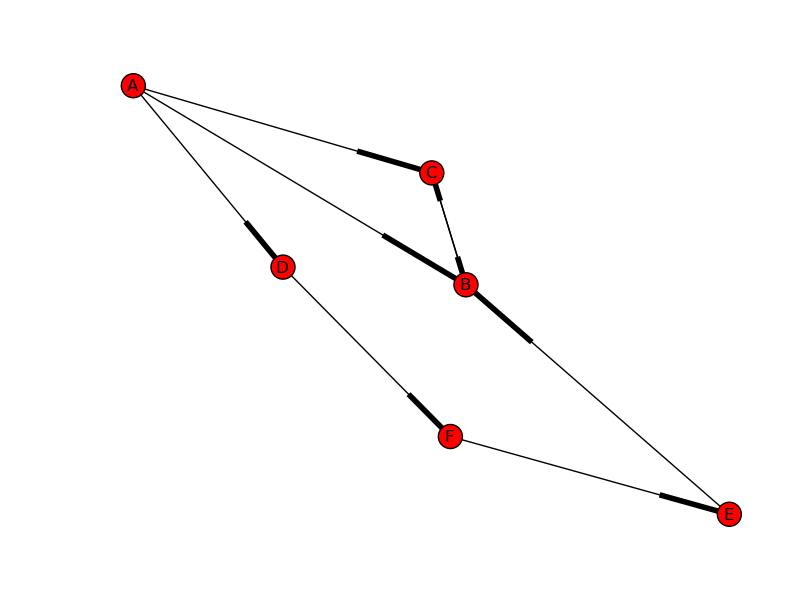


Figure : Graph Visualization for the given NeuronList.txt