Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_ / .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.0.1

/\_/

Using Python version 2.7.6 (default, Mar 22 2014 22:59:56)

SparkSession available as 'spark'.

>>> sc

<pyspark.context.SparkContext object at 0x7f333bf3ecd0>

>>> rdd=sc.textFile("file:////home/shalbigdata/Documents/spark/Assignments/assignment3/input/higgs-social\_network\_edgelist.txt")

# generate ((person,’follows’),friend) and ((friend,’followed by’), person) pairs

>>> def followpairs(x):

... tokens = x.split(' ')

... list = []

... tup = ()

... tup = ((tokens[0],'follows'),tokens[1])

... list.append(tup)

... tup = ((tokens[1],'followed by'),tokens[0])

... list.append(tup)

... return list

# flat map transformation with the above function

>>> follopairs = rdd.flatMap(lambda x:followpairs(x))

>>> follogrped = follopairs.groupByKey().mapValues(lambda x:list(x))

# change the key from compound to simple, so that they can be sorted and grouped by the simple key

>>> changekey = follogrped.map(lambda (x,y) : (x[0],(x[1],y))).sortByKey()

>>> grppairs = changekey.groupByKey().mapValues(lambda x:list(x))

#generate friends of friends

>>> def generatefof(k):

... x = k[1]

... tup1 = ()

... tup2 = ()

... personfollows = []

... personfollowedby = []

... output = []

... optup = ()

... tup1 = x[0]

... if len(x) == 2:

... tup2 = x[1]

... if(len(tup1) and tup1[0] == "follows"):

... personfollows = tup1[1]

... elif(len(tup1) and tup1[0] == "followed by"):

... personfollowedby = tup1[1]

... if(len(tup2) and tup2[0] == "follows"):

... personfollows = tup2[1]

... elif(len(tup2) and tup2[0] == "followed by"):

... personfollowedby = tup2[1]

... if(len(personfollows) > 0 and len(personfollowedby) > 0):

... for followed in personfollows:

... for follower in personfollowedby:

... if(follower != followed):

... optup = (follower,followed)

... output.append(optup)

... return output

>>> fof = grppairs.flatMap(lambda x:generatefof(x))

# group the friends of friends pairs by key

>>> fof1 = fof.groupByKey().mapValues(lambda x:list(x))

# filter only the records that show ‘follows’ relation. This is to compare with the friends of friends and eliminate the friends already being followed

>>> follows = changekey.filter(lambda (x,y): y[0]=="follows")

# generate ‘should follow’ pairs

>>> fof2 = fof1.map(lambda (x,y):(x,('should follow',y)))

# join already followed friends and ‘should follow’ friends by key

>>> recom = follows.join(fof2)

# compare and remove already followed friends from ‘should follow’ friends

>>> def removeCommon(x):

... followspair = x[0]

... recompair = x[1]

... follows = followspair[1]

... recom = recompair[1]

... for i in follows:

... if i in recom:

... recom.remove(i)

... return recom

...

>>> recomresult = recom.mapValues(lambda x:removeCommon(x))