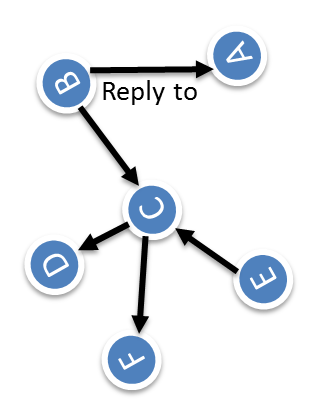
**Praktikum 2**

**Social Network Analysis (SNA)**

Sesi praktikum ini akan mendemonstrasikan langkah-langkah dalam Social Network Analysis (SNA). Langkah pertama adalah pengumpulan data. Data yang digunakan dalam praktikum ini adalah data tweet terkait dengan topik tertentu yang sedang ramai diperbincangkan. Proses pengumpulan data menggunakan crawling data melalui Twitter API seperti yang telah dipelajari pada praktikum pertama. Berdasarkan anatomi tweet, kita dapat mengetahui bahwa terdapat beberapa atribut yang dapat digunakan untuk mengkonstruksi jejaring sosial dari pengguna twitter yang terlibat dalam topik perbincangan tersebut. Interaksi antara pengguna dalam topik yang diperbincangkan dapat berupa reply, retweet atau mention. Pada praktikum ini kita akan menggunakan atribut “in\_reply\_to\_user\_id” untuk menggambarkan interaksi antara pengguna berupa membalas tweet dari user lain.

SNA sangat erat kaitannya dengan graph. Graph terbentuk dari node atau vertex yang menggambarkan actor yang terlibat dalam jejaring sosial serta link atau edge yang merepresentasikan hubungan / keterkaitan antara aktor tersebut. Praktikum ini menggunakan pengguna twitter sebagai node dan interaksi membalas tweet sebagai link/ edge, sebagaimana ditunjukan pada Gambar 1. Graph yang akan digunakan berupa directed graph karena interaksi antara pengguna ketika saling membalas tweet merupakan hubungan yang memiliki arah. Pembentukan graph serta analisis ukuran-ukuran dalam SNA dilakukan dengan bantuan library NetworkX (https://networkx.github.io/documentation/stable/). Visualisasi dari graph dilakukan dengan memanfaatkan aplikasi Gephi (<https://gephi.org/>).



Gambar 1. Graph Interaksi Pengguna

Praktikum ini dilakukan dengan menggunakan Jupyter Notebook sebagaimana praktikum sebelumnya. Selain itu pada praktikum ini diperkenalkan juga penggunaan tools visualisasi graph yaitu Gephi, yang bisa diunduh pada tautan yang telah disebutkan di atas.

Praktikum ini akan dibagi ke dalam 3 (tiga) bagian yaitu pengumpulan data, analisis graph, serta visualisasi graph.

1. Pengumpulan Data
   1. Mencari informasi mengenai topik yang sedang ramai diperbincangkan di Twiter. Tweepy menyediakan fungsi untuk memanggil Twitter API untuk informasi mengenai trending topic

import tweepy

import json

#isi data key dan token sesuai dengan twitter developer anda

consumer\_key = ' '

consumer\_secret = ' '

access\_token = ' '

access\_token\_secret = ' '

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

api = tweepy.API(auth,wait\_on\_rate\_limit=True)

#Kode where on earth identifier lookup pada http://woeid.rosselliot.co.nz/lookup/indonesia

WOE\_ID = 23424846

#menggunakan api tweepy untuk memperoleh data trending topic twitter di Indonesia

indonesia\_trend = api.trends\_place(WOE\_ID)

#print data trending topic yang telah didapatkan

trends = json.loads(json.dumps(indonesia\_trend, indent=1))

#mencetak trending topic name

trends = json.loads(json.dumps(indonesia\_trend, indent=1))

for trend in trends:

for t in trend["trends"]:

print(t["name"])

* 1. Melakukan crawling data berdasarkan kata kunci mengenai trending topic yang menarik untuk dipelajari. Crawling data dilakukan dengan teknik yang sama dengan praktikum pertama mengenai crawling data Twitter. Perubahan hanya dilakukan pada query, sehingga pencarian data tweet mencakup tujuh hari ke belakang hingga saat ini. Pada praktikum ini kita akan menggunakan topik mengenai radikalisme, sehingga pencarian data akan menggunakan kata kunci berupa hash tag yang sempat menjadi trending topik pada pekan ke-4 Bulan Oktober, yaitu “#SapuBersihRadikalisme”. Hasil crawling tanggal 27 Oktober 2019 diperoleh 5.272 tweets, dari jumlah tersebut terdapat 2.688 yang merupakan tweet balasan terhadap tweet lain, tweet tersebut yang dapat digunakan untuk menggambarkan interaksi pengguna dalam pembicaraan mengenai topic tersebut.
  2. Mengolah data tweet yang telah dikumpulkan untuk mendapatkan data mengenai nodes dan edges yang membentuk jejaring interkasi antar pengguna dalam topik pembicaraan yang ditentukan. Pada tahap ini kita akan membentuk data tentang node/ actor yang terlibat dalam jejaring sosial, data node akan berisi informasi mengenai id, name, screen\_name, location, followers\_count, dan friends\_count. Setelah itu, data edge dibentuk dengan membuat pasangan id antara id dari pengguna yang membalas suatu tweet dengan id dari pengguna yang membuat tweet yang dibalas tersebut.

#import library untuk mengolah data dalam format JSON

import json

import csv

import sys

import tweepy

#Load data tweet untuk mengambil pasangan id pengguna yang berinteraksi dengan membalas tweet

tws=[]

for l in open('radikalisme\_Tweets.json'):

tw = json.loads(l)

if(tw["in\_reply\_to\_user\_id"] is not None):

tws.append([tw["user"]["id"],tw["in\_reply\_to\_user\_id"]])

#menggabungkan id pengguna yang berinteraksi ke dalam sebuah list

nodes=[]

for t in tws:

if(t[0] not in nodes):

nodes.append(t[0])

if(t[1] not in nodes):

nodes.append(t[1])

#menggabungkan id pengguna yang berinteraksi ke dalam sebuah list

nodes=[]

for t in tws:

if(t[0] not in nodes):

nodes.append(t[0])

if(t[1] not in nodes):

nodes.append(t[1])

#menyiapkan api twitter untuk melengkapi data user

consumer\_key = ' '

consumer\_secret = ' '

access\_token = ' '

access\_token\_secret = ' '

auth = tweepy.AppAuthHandler(consumer\_key,consumer\_secret)

api = tweepy.API(auth, wait\_on\_rate\_limit=True)

#mengumpulkan data user twitter yang berinteraksi dalam topik pembicaraan

users=[]

count=1

for n in nodes:

try:

user=api.get\_user(n)

users.append(user)

except:

continue

sys.stdout.write("\r");sys.stdout.write(str(count));sys.stdout.flush()

count+=1

#mengumpulkan data user twitter yang berinteraksi dalam topik pembicaraan

users=[]

count=1

for n in nodes:

try:

user=api.get\_user(n)

users.append(user)

except:

continue

sys.stdout.write("\r");sys.stdout.write(str(count));sys.stdout.flush()

count+=1

#menyiapkan file csv yang akan digunakan untuk menampung data nodes

csvNode = open('radikalisme-node-list.csv', 'w',newline='')

nodeWriter = csv.writer(csvNode)

#menulis data node ke dalam file csv

for user in users:

nodeWriter.writerow([user.id,user.name.encode('utf-8'),user.screen\_name.encode('utf-8'),user.location.encode('utf-8'),user.followers\_count,user.friends\_count])

csvNode.close()

#menyiapkan file csv untuk menampung data edge

csvEdge = open('radikalisme-edge-list.csv', 'w',newline='')

edgeWriter = csv.writer(csvEdge)

#menyiapkan file csv untuk menampung data edge

csvEdge = open('radikalisme-edge-list.csv', 'a')

edgeWriter = csv.writer(csvEdge)

#menulis data ege ke dalam file csv

for t in tws:

edgeWriter.writerow(t)

csvEdge.close()

1. Analisis Graph
2. Import Modul yang dibutuhkan, apabila belum modul-modul tersebut belum terinstall, silahkan install terlebih dahulu melalui command line dengan perintah Python –m pip install [nama modul]

import networkx as nx

from operator import itemgetter

import csv

1. Menyiapkan data node dan edge untuk membentuk graph menggunakan file csv yang telah disiapkan pada langkah sebelumnya

#membaca file node list

with open('radikalisme-node-list.csv', 'r') as nodecsv: # Buka file csv

nodereader = csv.reader(nodecsv) # membaca file csv

nodes = [n for n in nodereader if len(n)>0] #memastikan hanya baris yang ada isinya yang diambil

node\_ids = [n[0] for n in nodes] # menyiapkan list yang berisi id pengguna

#membaca file edge list

with open('radikalisme-edge-list.csv', 'r') as edgecsv: # buka file csv

edgereader = csv.reader(edgecsv) # membaca file csv

rows = [[row[0],row[1]] for row in edgereader if len(row)>0] #memastikan hanya baris yang ada isinya yang diambil

edges =[tuple(e) for e in rows]

1. Membentuk Graph menggunakan data dalam file node dan edge

G = nx.DiGraph() #membuat objek graph

G.add\_nodes\_from(node\_ids) #menambahkan node ke dalam graph

G.add\_edges\_from(edges) #menambahkan edge ke dalam graph

1. Menambahkan atribut lainnya pada data nodes, untuk melengkapi informasi mengenai node tersebut

#membuat dictionary untuk user name twitter

screen\_name\_dict = {}

for node in nodes:

screen\_name\_dict[node[0]] = node[2].replace("b'","").replace("'","")

#membuat dictionary untuk nama jumlah followers

follower\_dict = {}

for node in nodes:

follower\_dict[node[0]] = node[4]

#menambahkan attribute ke dalam graph node

nx.set\_node\_attributes(G, screen\_name\_dict, 'nama\_twitter')

nx.set\_node\_attributes(G, follower\_dict, 'followers\_count')

1. Menampilkan informasi mengenai graph

print(nx.info(G))

1. Menghitung indegree dan outdegree centrality

#menghitung

indegree\_dict = dict(G.in\_degree())

outdegree\_dict = dict(G.out\_degree())

#menyimpan ke dalam attribute

nx.set\_node\_attributes(G, indegree\_dict, 'indegree')

nx.set\_node\_attributes(G, outdegree\_dict, 'outdegree')

1. Menampilkan daftar 20 node dengan indegree dan outdegree centrality terbesar

sorted\_indegree = sorted(indegree\_dict.items(), key=itemgetter(1), reverse=True)

print("Top 20 nodes by indegree centrality:")

for d in sorted\_indegree[:20]:

print(d[0],"|",G.node[d[0]]['nama\_twitter'],"|", d[1])

sorted\_outdegree = sorted(outdegree\_dict.items(), key=itemgetter(1), reverse=True)

print("Top 20 nodes by outdegree centrality:")

for d in sorted\_outdegree[:20]:

print(d[0],"|",G.node[d[0]]['nama\_twitter'],"|", d[1])

1. Menghitung betweenness centrality

betweenness\_dict = nx.betweenness\_centrality(G)

nx.set\_node\_attributes(G, betweenness\_dict, 'betweenness')

1. Menampilkan 20 node dengan betweenneess centrality terbesar

sorted\_betweenness = sorted(betweenness\_dict.items(), key=itemgetter(1), reverse=True)

print("Top 20 nodes by eigenvector centrality:")

for d in sorted\_betweenness[:20]:

print(d[0],"|",G.node[d[0]]['nama\_twitter'],"|", d[1])

1. **Visualisasi Graph**

NetworkX memiliki kemampuan untuk visualisasi Graph, baik untuk graph berarah maupun graph yang tidak berarah. Selain itu, terdapat pula beberapa tools yang tersedia untuk visualisasi graph, salah satunya adalah Gephi. Bagian ini akan mencakup penggunaan kedua tools tersebut untuk visualisasi graph dari data yang telah kita olah.

1. Visualisasi graph dengan NetworkX

Visualisasi ini masih menggunakan objek graph yang sama dengan langkah yang telah dilakukan sebelumnya. Terdapat komponen library yang perlu ditambahkan yaitu matplotlib.

#import komponen matplotlib

from matplotlib import pyplot as plt

#visualisasi graph dengan matplotlib

pos = nx.spring\_layout(G)

nx.draw(G, pos)

node\_sizes = []

labels = []

nodeList = G.nodes

nx.draw\_networkx\_nodes(G, pos, node\_list=nodeList)

node\_labels = nx.get\_node\_attributes(G,'nama\_twitter')

nx.draw\_networkx\_labels(G, pos, labels = node\_labels, font\_size=10)

plt.savefig('this.png')

plt.show()

1. Visualisasi dengan Gephi

NetworkX memiliki kemampuan untuk melakukan export graph ke dalam berbagai format file. Salah satunya ke dalam Graph Exchange XML Format (.gexf). Format tersebut merupakan bahasa standar yang digunakan untuk pertukaran data mengenai struktur jejaring yang kompleks. Pada bagian ini kita akan mencoba melakukan export graph yang telah kita buat ke dalam format GEXF, kemudian kita gunakan untuk visualisasi graph mgnggunakan aplikasi Gephi. Apabila pada perangkat anda belum terinstal Gephi, silahkan lakukan instalasi Gephi terlebih dahulu, versi yang digunakan dalam praktikum ini adalah Gephi 0.9.2.

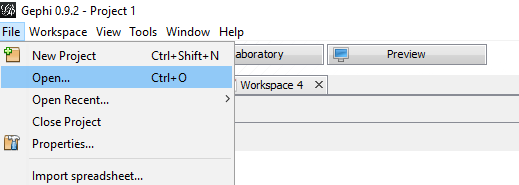
1. Export graph ke dalam format GEXF

#export graph

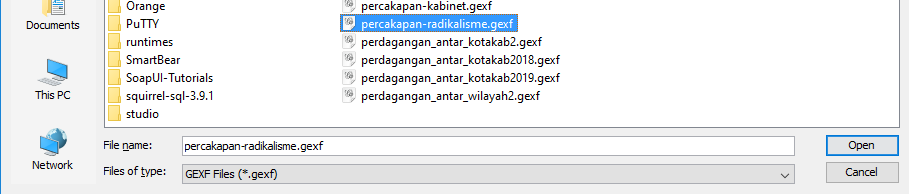
nx.write\_gexf(G, 'percakapan-radikalisme.gexf')

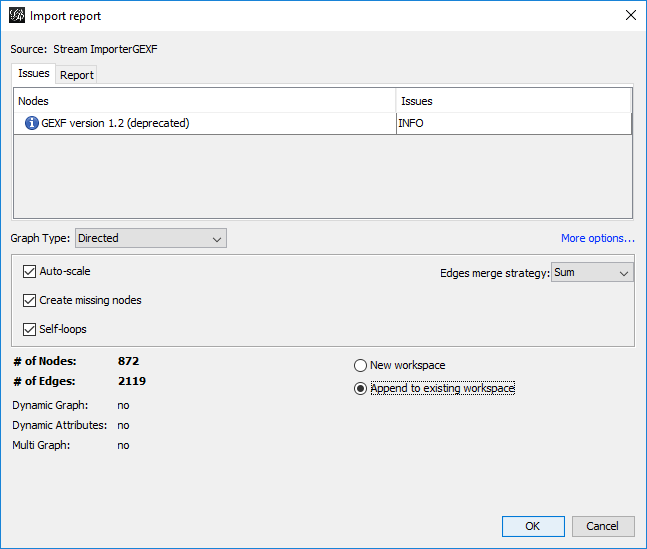
Setelah dijalankan file dengan nama 'percakapan-radikalisme.gexf' akan terbentuk pada directory tempat anda menjalankan Jupyter Notebook.

1. Jalankan program Gephi, dan pilih menu File 🡪 Open

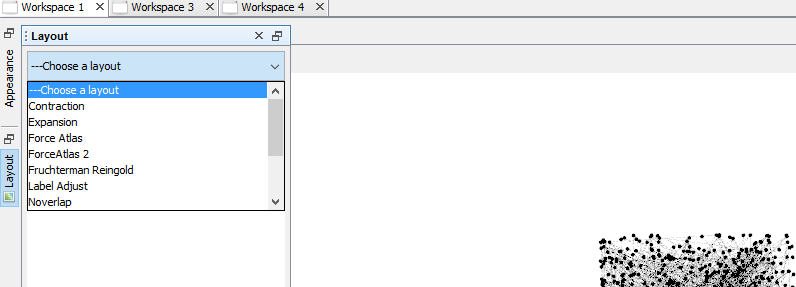


1. Pilih file dengan extensi gexf yang telah dibuat

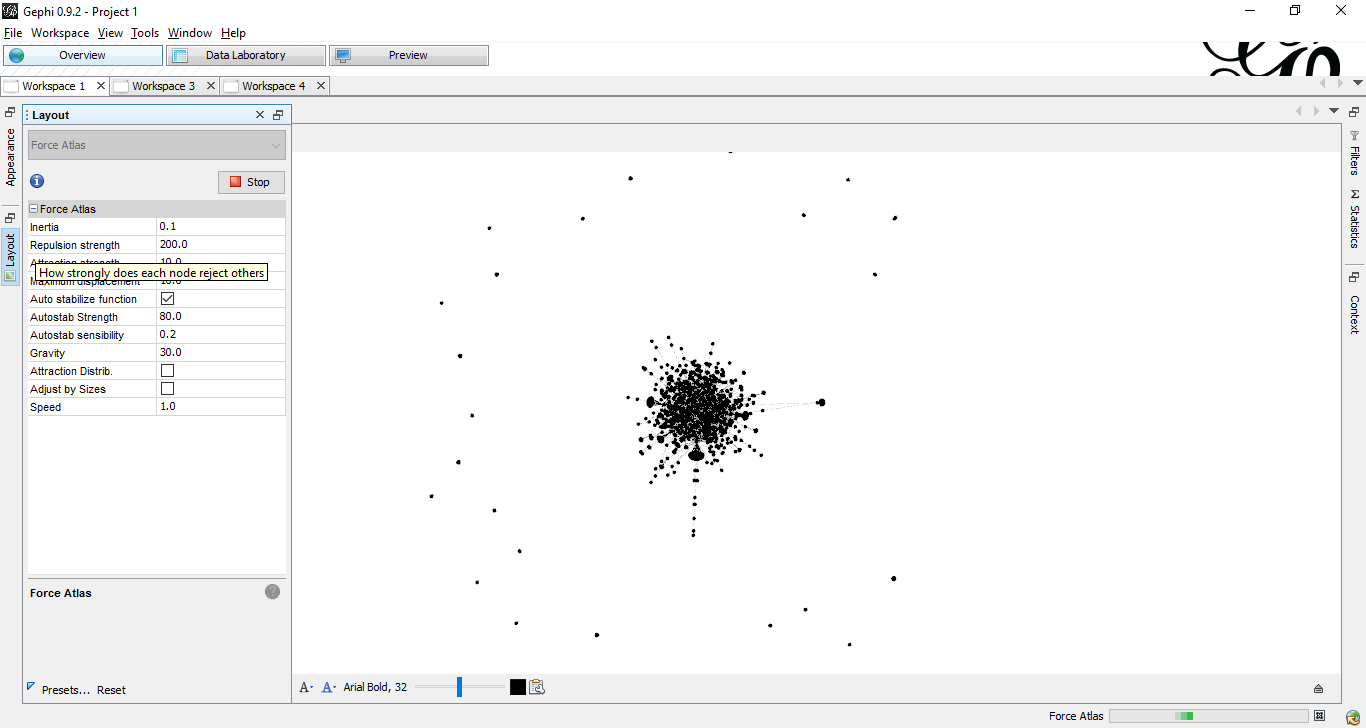




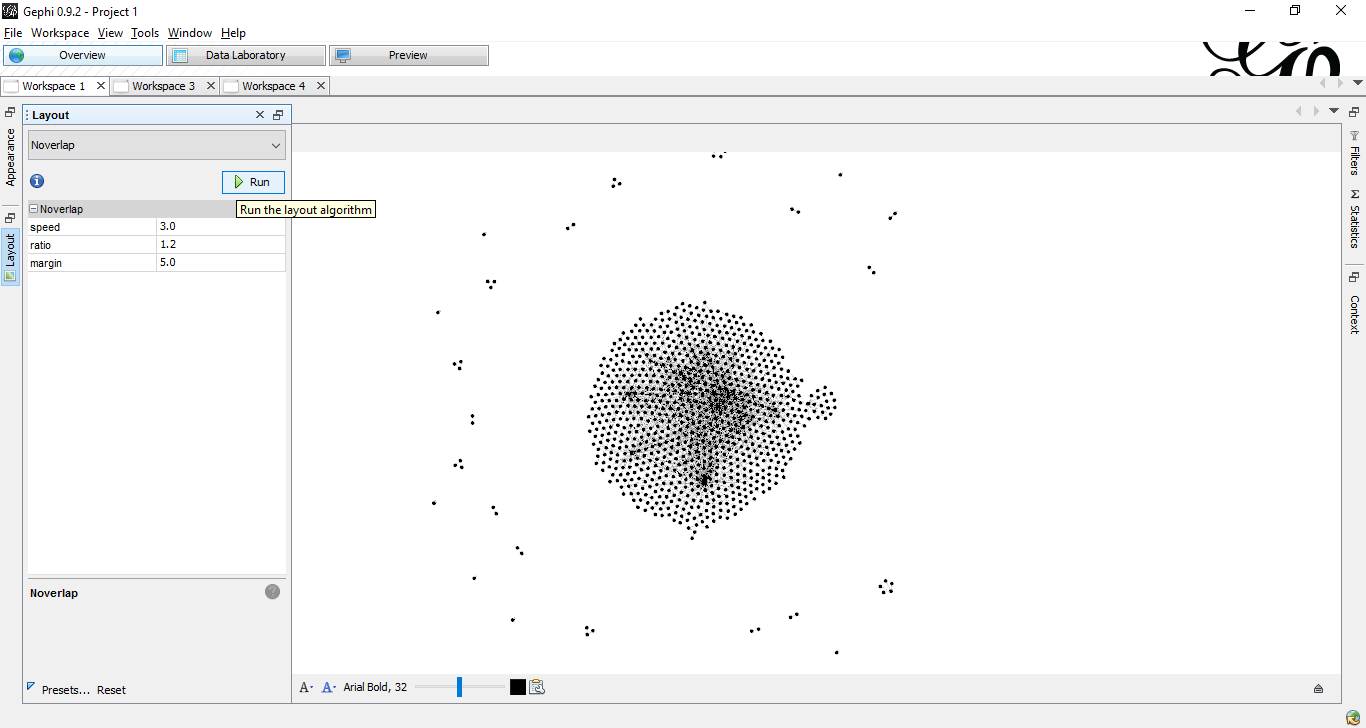
1. Atur layout dari graph yang ditampilkan, dengan memilih salah satu bentuk layout, kita akan coba menggunakan layout Fors Atlas, kemudian kita terapkan juga Noverlap.



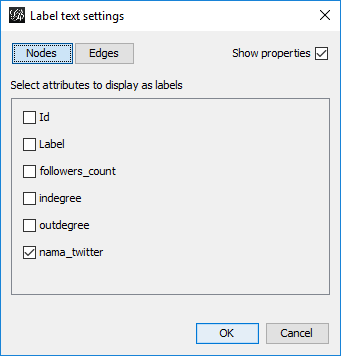
Menerapkan layout Foce Atlas



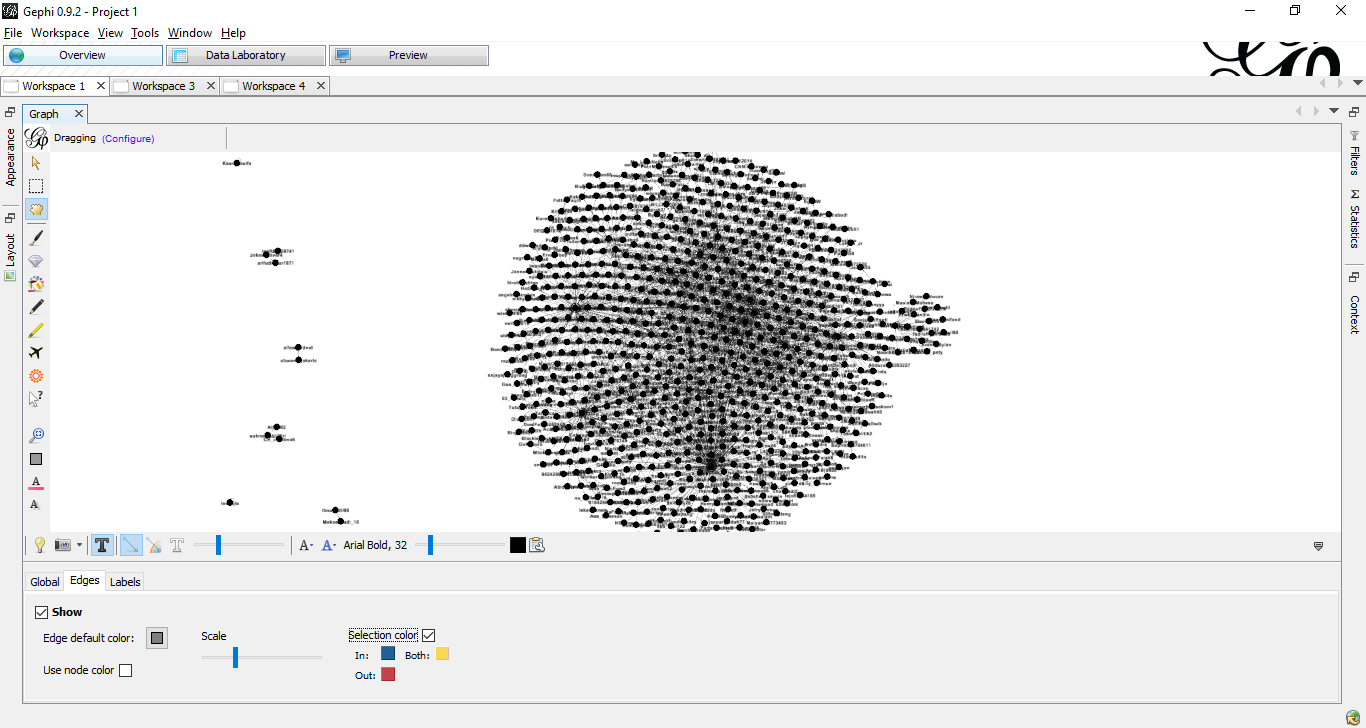
Menerapkan layout Noverlap



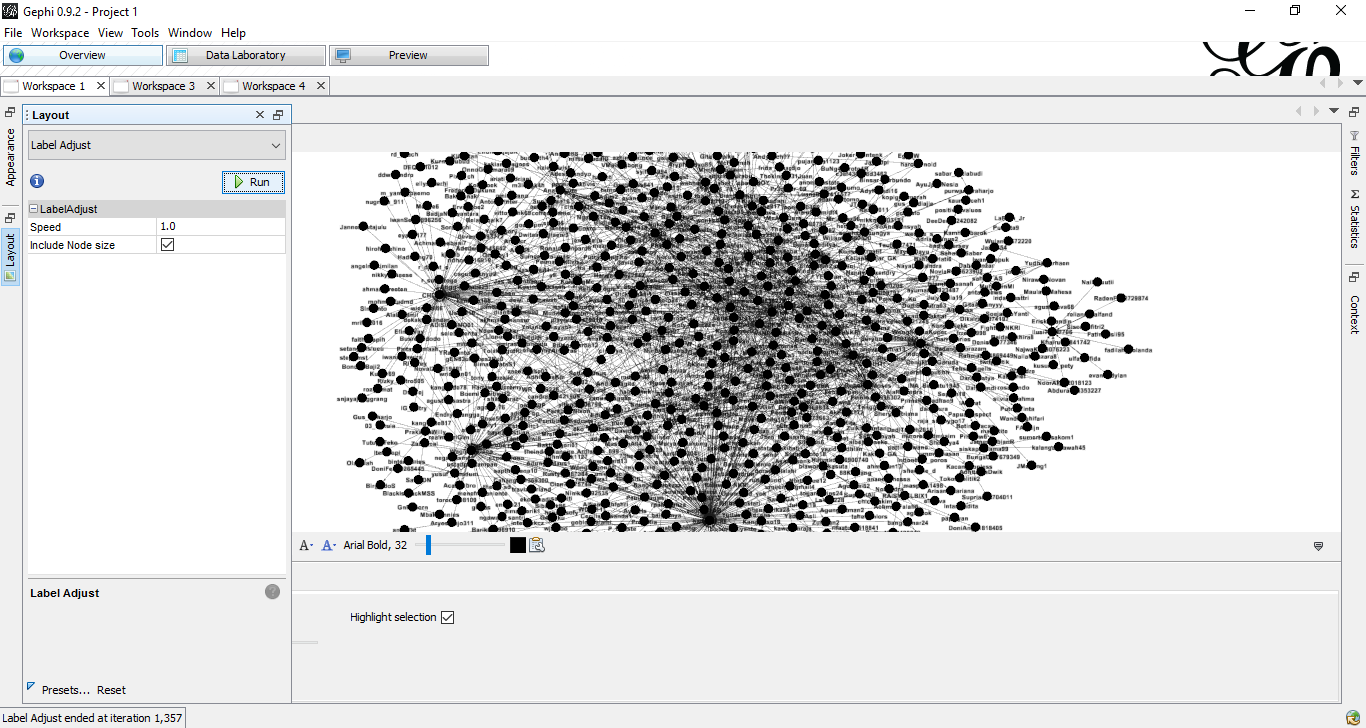
Mengatur label pada node



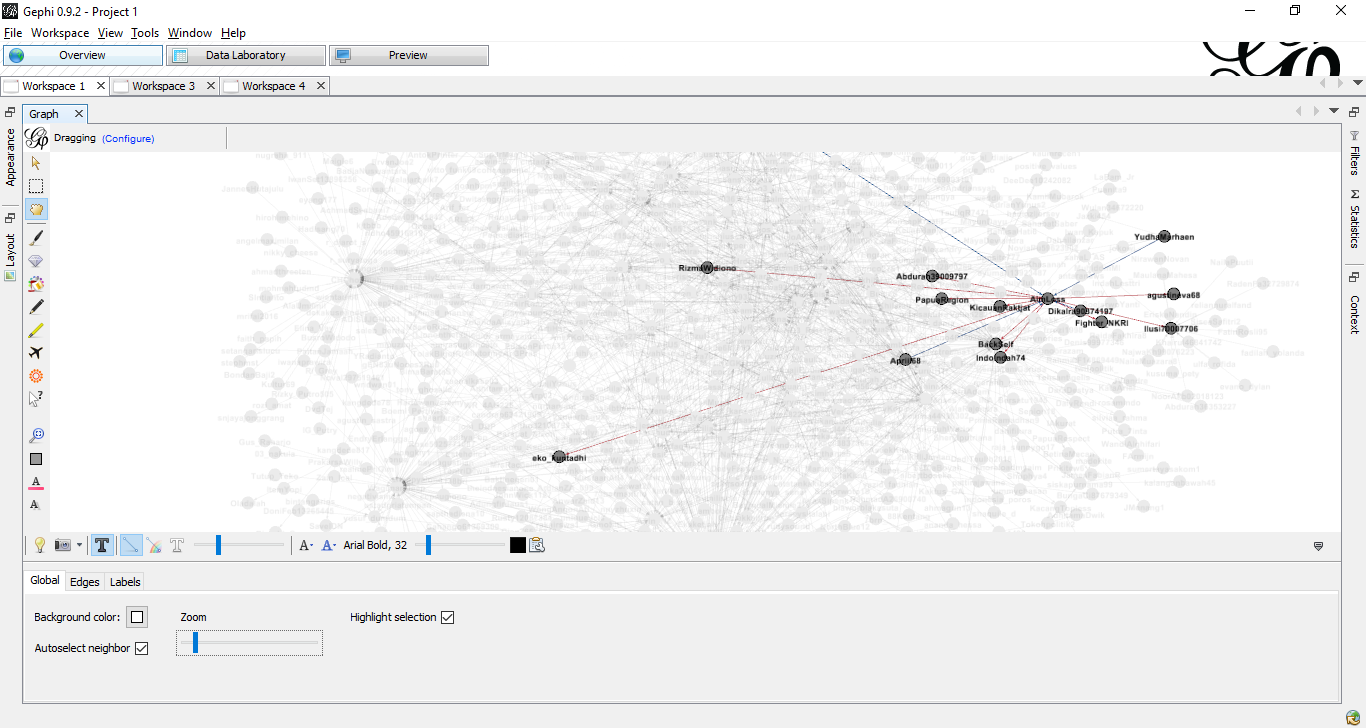
Mengatur warna dan ukuran garis edge



Menerapkan layout Label Adjust, agar label pada node lebih terbaca



Melakukan zoom tampilan graph dan memilih fokus pada node tertentu



Mengatur ukuran dan warna node berdasarkan ranking tertentu, misalnya berdasarkan indegree centrality

