import requests

import networkx as nx

import math

from time import sleep

# 山中教授がノーベル賞を受賞する理由となった論文のPubMed ID

YAMANAKA\_PMID = '16904174'

# EBIが提供する論文データベースのAPIアドレス

EBI\_URL = 'https://www.ebi.ac.uk/europepmc/webservices/rest/MED/'

# 文献検索APIのコマンド

COMMAND\_CITATION = '/citations'

COMMAND\_REFS = '/references'

# 該当論文に幾つの引用数があるのかを調べる

url = EBI\_URL + YAMANAKA\_PMID + COMMAND\_CITATION

params = {'format': 'json', 'pageSize': 1, 'page': 1}

r = requests.get(url, params=params)

result = r.json()

hit\_count = result['hitCount']

print('この論文からは' + str(hit\_count) +'件の引用があります。')

# この論文が引用されている論文の情報

def get\_refs(pmid, hit\_count):

url = EBI\_URL + pmid + COMMAND\_CITATION

max\_page = math.ceil(hit\_count/1000)

refs = []

counter = 1

while counter <= max\_page:

params = {'format': 'json', 'pageSize': 1000, 'page': counter}

r = requests.get(url, params=params)

result = r.json()

cited\_by = result['citationList']['citation']

refs += cited\_by

counter = counter + 1

sleep(0.2)

return refs

refs = get\_refs(YAMANAKA\_PMID, hit\_count)

# 全てのリファレンス情報がここに入っている。したがって長さは同一

print(len(refs))

# どんな情報が取れたか確認する。

print(refs[0])

# 被引用数が一定以上のものだけを取り出す。

CITED\_BY\_TAG = 'citedByCount'

# 引用数が、これ以上あったものだけを使う。

TH = 100

filtered\_pub = []

for ref in refs:

count = ref[CITED\_BY\_TAG]

if count >= TH:

filtered\_pub.append(ref)

print('被引用が' + str(TH) + '件以上あった論文は' + str(len(filtered\_pub)) + '報です。')

# グラフモデルにノードとして追加

# 有向グラフとして生成（引用には方向性があるので）

G = nx.MultiDiGraph()

G.add\_node(YAMANAKA\_PMID, id=YAMANAKA\_PMID)

def add\_nodes(G, entry\_list, node\_type):

new\_nodes = []

for entry in entry\_list:

keys = entry.keys()

pmid = entry['id']

G.add\_node(pmid)

new\_node = G.nodes[pmid]

new\_nodes.append(new\_node)

new\_node['nodeType'] = node\_type

for key in keys:

new\_node[key] = entry[key]

return new\_nodes

ref\_nodes = add\_nodes(G, filtered\_pub, 'refs')

# エッジを追加 （被引用の方向）

def add\_edges(G, target, edge\_type):

nodes = G.nodes()

for node in nodes:

G.add\_edge(node, target, edge\_type=edge\_type)

add\_edges(G, YAMANAKA\_PMID, 'refs')

# この論文が引用している論文を取得

url2 = EBI\_URL + YAMANAKA\_PMID + COMMAND\_REFS

params = {'format': 'json', 'pageSize': 1000, 'page': 1}

r2 = requests.get(url2, params=params)

cites = r2.json()

cite\_list = cites['referenceList']['reference']

print(len(cite\_list))

cite\_nodes = add\_nodes(G, cite\_list, 'cites')

# 逆向きの、元になる論文が引用している文献をグラフに追加する

def add\_reverse\_edges(G, nodes, source, edge\_type):

for node in nodes:

G.add\_edge(source, node['id'], edge\_type=edge\_type)

add\_reverse\_edges(G, cite\_nodes, YAMANAKA\_PMID, 'cites')

# 元の論文を引用した論文をさらに引用しているものを取得する。

new\_graph = {}

counter = 0

for pub in filtered\_pub:

pmid = pub['id']

cited\_by\_count = pub[CITED\_BY\_TAG]

new\_refs = get\_refs(pmid, cited\_by\_count)

new\_graph[pmid] = new\_refs

# ２０件ごとにデバッグメッセージ

if counter%20 == 0:

print(str(counter) + ': ID = ' + pmid + ' done (' + str(cited\_by\_count) + ')')

counter = counter+1

len(new\_graph) # 何件のデータがあったか確認。

def filter\_pubs(pubs, th):

filtered\_pubs = []

for pub in pubs:

count = pub[CITED\_BY\_TAG]

if count >= th:

filtered\_pubs.append(pub)

return filtered\_pubs

filtered\_graph = {}

# 同じフィルタをこれらの論文にも適用する

pmids = new\_graph.keys()

for pmid in pmids:

nodes = new\_graph[pmid]

filtered = filter\_pubs(nodes, TH)

filtered\_graph[pmid] = filtered

# ノードを加える

for pmid in pmids:

nodes = filtered\_graph[pmid]

if len(nodes) == 0:

continue

for node in nodes:

newId = node['id']

keys = node.keys()

if G.has\_node(newId):

continue

G.add\_node(newId)

new\_node = G.nodes[newId]

new\_node['nodeType'] = '2nd\_refs'

for key in keys:

new\_node[key] = node[key]

# 直接引用のノードに、二次引用のエッジを加える

for pmid in pmids:

nodes = filtered\_graph[pmid]

if len(nodes) == 0:

continue

for node in nodes:

newId = node['id']

G.add\_edge(newId, pmid, edge\_type='2nd\_refs')

# 直接引用のノードに、二次引用のエッジを加える

for pmid in pmids:

nodes = filtered\_graph[pmid]

if len(nodes) == 0:

continue

for node in nodes:

newId = node['id']

G.add\_edge(newId, pmid, edge\_type='2nd\_refs')

nx.write\_graphml\_lxml(G, "yamanaka\_final.graphml")