```
/home/roni/Workspace/ufu/blockchain/ClusteringBitcoinWallets/Clustering.py
Page 1 of 3
Sex 11 Out 2019 00:31:18 -03
```

```
from pymongo import MongoClient
 2
     import bson
 3
     import requests
 4
 5
     URL = "https://blockchain.info"
 6
7
     LATEST BLOCK = '/latestblock'
     RAW BLOCK = '/rawblock/
 8
 9
     client = MongoClient()
10
     database = client['bitcoin-cluster']
11
12
13
     Entity = database['Entity']
     Blocks = database['Block']
14
     Transaction = database['Transaction']
15
     AddressEntity = database['AddressEntity']
16
17
     control = database['control']
18
     AddressChange = database['AddressChange']
19
     count = 0
20
21
22
     def updateLastBlock(lastBlock):
23
         try:
24
             control.update one({"name": "lastblock"}, {"$set": {"value": lastBlock}})
25
             Blocks.insert(lastBlock)
26
         except Exception as e:
             print("Fail on update the last block ", e)
27
28
             raise e
29
30
31
     def getTheLastBlock():
32
33
34
         lastBlockControl = control.find one({
              'name": "lastblock"
         })
35
36
         if lastBlockControl is not None:
37
             lastBlock = lastBlockControl['value']
38
             control.insert({"name": "lastblock"})
39
40
             lastBlock = requests.get(URL + LATEST BLOCK).json()
41
             lastBlock = requests.get(URL + RAW BLOCK + lastBlock['hash']).json()
42
             updateLastBlock(lastBlock)
43
44
         return lastBlock
45
46
47
     def populateTransactionsDatabaseWhenNecessary(lastBlock, MAX POPULATION=1e6):
48
         print("doing count transactions")
49
         population = Transaction.count documents({})
50
         print(population, " transactions!'
51
         while True:
52
             if population >= MAX POPULATION:
53
                  return
54
             try:
                  print("Calling the last block...")
55
                 actualBlock = requests.get(URL + RAW BLOCK +
56
                                                                                              7
                  lastBlock['prev block']).json()
57
                 Transaction.insert many(actualBlock['tx'])
58
                 population += len(actualBlock['tx'])
                 updateLastBlock(actualBlock)
59
60
             except Exception as e:
                  print("fail on insert transaction ", e)
61
62
                  raise e
63
64
             lastBlock = actualBlock
             print("Actual Population: ", population)
65
66
67
68
     def executeH1Clustering():
69
         print("Proccess transactions with h1: ")
```

```
/home/roni/Workspace/ufu/blockchain/ClusteringBitcoinWallets/Clustering.py
Page 2 of 3 Sex 11 Out 2019 00:31:18 -03
```

```
70
           for transaction in Transaction.find():
71
72
               count transactions()
73
74
               addresses = get all address in transaction(transaction)
75
76
               entityToMerge = get all entity and remove address already in db(addresses)
77
78
               newEntityId = get new entity id(entityToMerge)
 79
80
               update database addressEntity(addresses, entityToMerge, newEntityId)
81
82
      def executeH2Clustering():
          change wallets = {}
83
84
           for transaction in Transaction.find():
85
               count transactions()
86
87
               addresses = get all address in transaction(transaction)
88
89
               if len(transaction['out']) > 1:
90
                   first time = 0
91
                   first address = None
92
93
                   for output in transaction['out']:
                        if output.get('addr') is not None and not (output['addr'] in
addresses or output['addr'] in change wallets):
    first address = output['addr']
94
                                                                                                   7
95
                            first time += 1
96
97
98
                   if first time == 1:
99
                        add change wallet(addresses, first address)
100
                        change wallets[first address] = True
101
102
      def add change wallet(addresses, first address):
103
104
           if len(addresses):
105
               entity = AddressEntity.find one({"address": addresses[0]})
106
               if entity is not None:
                    first time on db = AddressEntity.find one({"address": first address})
107
                   if first time on db is None or not len(first time on db):
    AddressEntity.insert one({"address": first address,"
108
109
                                                                                                    7
                        entity['entity']})
110
111
                   if AddressChange.find one({"address": first address}) is None:
                        AddressChange.insert one({"address": first address, "entity":
112
                        entity['entity']})
113
114
      def update database addressEntity(addresses, entityToMerge, newEntityId):
115
116
           # if merge, update new entityId
117
           if len(entityToMerge) > 1:
118
               AddressEntity.update many({"entity": {"$in": entityToMerge}}, {"$set":
                                                                                                    7
               {"entity": newEntityId}})
          # insert all addresses without entity
119
          newAddressesEntity = [{'address': address, 'entity': newEntityId} for address
120
                                                                                                   7
          in addresses]
121
           if len(newAddressesEntity):
122
               AddressEntity.insert many(newAddressesEntity)
123
124
125
      def count transactions():
126
           global count
127
           if not count % 10000:
128
               print(count)
129
           count += 1
130
131
132
      def get all address in transaction(transaction):
133
           addresses = []
134
           for inputAddress in transaction['inputs']:
```

## /home/roni/Workspace/ufu/blockchain/ClusteringBitcoinWallets/Clustering.py Page 3 of 3 Sex 11 Out 2019 00:31:18 -03

```
address = getAddress(inputAddress)
135
136
               if address is not None:
137
                    addresses.append(address)
138
           addresses = list(set(addresses))
139
           return addresses
140
141
142
      def get all entity and remove address already in db(addresses):
143
           addressesEntity = AddressEntity.find({"address": {"$in": addresses}})
           entityToMerge = []
144
           for addressEntity in addressesEntity:
145
               addresses.remove(addressEntity['address'])
if addressEntity['entity'] not in entityToMerge:
146
147
                    entityToMerge.append(addressEntity['entity'])
148
149
           return entityToMerge
150
151
152
      def get new entity id(entityToMerge):
153
           if len(entityToMerge) == 1:
154
               newEntityId = entityToMerge[0]
155
           else:
156
               newEntityId = bson.objectid.ObjectId()
157
           return newEntityId
158
159
      def getAddress(inputAddress):
160
161
           address = None
           if inputAddress.get('prev') is not None:
    address = inputAddress['prev'].get('addr')
162
163
           if address is None and inputAddress.get('prev out') is not None:
164
165
               address = inputAddress['prev out'].get('addr')
166
           return address
167
168
169
      def main():
170
           global count
171
           populateTransactionsDatabaseWhenNecessary(getTheLastBlock())
172
           count = 0
173
           executeH1Clustering()
174
           count = 0
175
           executeH2Clustering()
176
177
      main()
```