graph_vis.py

```
import json
import pymongo
import pandas as pd
from time import gmtime, mktime, strptime
import networkx as nx



def load_from_mongo(mongo_db, mongo_db_coll, return_cursor=False, criteria=None, projection=None, **mongo_conn_kw):

    # Can use critera and projection to limit the data that is returned
    # look at http://docs.mongodb.org/manual/reference/method/db.collection.find/

    # Can use MongoDB's aggregations framework for more detailed queries

    client = pymongo.MongoClient(**mongo_conn_kw)
    db = client[mongo_db]
    coll = db[mongo_db_coll]

    tweet_iterator = coll.find()


    tweets_data = []
    now = mktime(gmtime())
    for tweet in tweet_iterator:

        author = ""
        rtauthor = ""
        age = rtage = followers = rtfollowers = 0

    try:
        author = tweet['user']['screen_name']
        rtauthor = tweet['retweeted_status']['user']['screen_name']
        rtage = int(now - mktime(strptime(tweet['retweeted_status']['user']['created_at'], "%a %b %d %H:%M:%S +0000 %Y")) / (60 * 60 * 24))
        rtfollowers = tweet['retweeted_status']['user']['followers_count']
    except:
        try:
            author = tweet['user']['screen_name']
        except:
            continue

    reply_to = ""
    if tweet['in_reply_to_screen_name'] != 'null':
        reply_to = tweet['in_reply_to_screen_name']

    age = int(now - mktime(strptime(tweet['user']['account_created'], "%a %b %d %H:%M:%S +0000 %Y")) / (60 * 60 * 24))

    followers = tweet['user']['followers_count']
    text = tweet['text']
    dict1 = {}

    dict1.update({'author': author, 'reply_to': reply_to, 'age': age, 'followers': followers, 'retweet_of': rtauthor, 'rtfollowers': rtfollower
    tweets_data.append(dict1)

tweets = pd.DataFrame(tweets_data)


return tweets
```

```
loaded_tweets = load_from_mongo('geoWorldGraphVisDatabase', 'tweets')

J = nx.DiGraph()

for index, row in loaded_tweets.iterrows():

    this_user_id = row['author']
    author = row['reply_to']
    followers = row['followers']
    age = row['age']
    rtfollowers = row['rtfollowers']
    rtage = row['rtage']

    if not this_user_id in J:
        J.add_node(this_user_id, attr_dict={
            'followers': row['followers'],
            'age': row['age'],
        })

    if author != "" and not author in J:
        J.add_node(author, attr_dict={
            'followers': row['rtfollowers'],
            'age': row['rtage'],
        })

    if author != "":
        if J.has_edge(author, this_user_id):
            J[author][this_user_id]['weight'] += 1
        else:
            J.add_weighted_edges_from([author, this_user_id, 1.0])
```

```
nx.write_gexf(J, 'ggrtages.gexf')
```