In [ ]:
```
## created the 19.05.2020
## updated the 19.05.2020
```

In [1]:
```
import os
import pandas as pd
import numpy as np
import time
import random
import json

from multiprocessing import Pool
import operator
import functools
from multiprocessing import cpu_count
cpu_number = cpu_count() - 1
```

In [95]:
```
# Extract information on users' tweet and retweet texts from the raw Twitter data. This should result in a
# dataframe with the columns user_handle, tweet_type, tweet_text, re_tweet_text for all status updates in the data.
columns = ["user_handle", "user_id", "tweet_id", "location_full_name",
           "location_country", "location_country_code", "tweet_geo",
           "coordinates", #done
           "tweet_time", #done
           "tweet_type", #done
           "tweet_text", "reply_user",
           "qu_tweet_text", "qu_user", "qu_tweet_time",
           "retweeted_user", "retweeted_tweet_text", "retweeted_tweet_time"]
```

In [96]:
```
import datetime
```

In [106]:
```
def extract_info(tweet):
    try:
        tweet = json.loads(tweet)
        new_row ={col: [np.nan] for col in columns}
        new_row["tweet_geo"] = False

        #userhandle
        new_row["user_handle"] = tweet['user']['screen_name']

        #user_id and tweet_id
        new_row["tweet_id"] = tweet["id"]
        new_row["user_id"] = tweet["user"]["id"]

        #tweet_time
        new_row["tweet_time"] = tweet["created_at"]
        new_row["tweet_time"] = time.strftime("%Y-%m-%d %H:%M:%S", time.strptime(new_row["tweet_time"], "%a %b %d %H:%M:%S +0000 %Y"))

        #tweet type
        new_row["tweet_type"] = "Tweet"
        if "quoted_status" in tweet:
            new_row["tweet_type"] = "Quote"
        if "retweeted_status" in tweet:
            new_row["tweet_type"] = "Retweet"
            if "quoted_status" in tweet:
                new_row["tweet_type"] = "Re_Quote"

        #tweet_text for tweet
        if new_row["tweet_type"] in ["Tweet", "Quote"]:
            if "extended_tweet" in tweet:
                new_row["tweet_text"] = tweet["extended_tweet"]["full_text"]
            else:
                new_row["tweet_text"] = tweet["text"]

        #reply to user
        if new_row["tweet_type"] in ["Tweet", "Quote", "Retweet", "Re_Quote"]:
            new_row["reply_user"] = tweet["in_reply_to_screen_name"]


        #quoted tweet infromation
        if new_row["tweet_type"] in ["Quote", "Re_Quote"]:
            if "quoted_status" in tweet:
                if "extended_tweet" in tweet["quoted_status"]:
                    new_row["qu_tweet_text"] = tweet["quoted_status"]["extended_tweet"]["full_text"]
                else:
                    new_row["qu_tweet_text"] = tweet["quoted_status"]["text"]
            new_row["qu_user"] = tweet["quoted_status"]["user"]["screen_name"]
            new_row["qu_tweet_time"] = tweet["quoted_status"]["created_at"]
            new_row["qu_tweet_time"] = time.strftime("%Y-%m-%d %H:%M:%S", time.strptime(new_row["qu_tweet_time"]
                                                                                        , "%a %b %d %H:%M:%S +0000 %Y"))

        #Retweeted tweet information
        if new_row["tweet_type"] in ["Re_Quote", "Retweet"]:
            if "retweeted_status" in tweet:
                if "extended_tweet" in tweet["retweeted_status"]:
                    new_row["retweeted_tweet_text"] = tweet["retweeted_status"]["extended_tweet"]["full_text"]
                else:
                    new_row["retweeted_tweet_text"] = tweet["retweeted_status"]["text"]
            new_row["retweeted_user"] = tweet["retweeted_status"] ["user"]["screen_name"]
            new_row["retweeted_tweet_time"] = tweet["retweeted_status"]["created_at"]
            new_row["retweeted_tweet_time"] = time.strftime("%Y-%m-%d %H:%M:%S", time.strptime(new_row
```

```
                ["retweeted_tweet_time"], "%a %b %d %H:%M:%S +0000 %Y"))

        #location
        new_row["tweet_geo"] = tweet["user"]["geo_enabled"]
        if new_row["tweet_geo"]:
            try:
                new_row["location_country_code"] = tweet["place"]["country_code"]
            except:
                pass
            try:
                new_row["location_country"] = tweet["place"]["country"]
            except:
                pass
            try:
                new_row["location_full_name"] = tweet["place"]["full_name"]
            except:
                pass
            try:
                new_row["coordinates"] = tweet["place"]["bounding_box"]["coordinates"]
            except:
                pass

        new_row = pd.DataFrame.from_dict(new_row)

        return(new_row)
    except:
        pass
```

In [107]:

```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 1
##NEED TO FILL IN INFO FOR FILE PATH AND WHERE TO SAVE FEATHER FILE
files_p1 = os.listdir("period2/p1/")
#p1 is periode 1

file_paths = ["period2/p1/" + file for file in files_p1 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    df_p1 = pd.concat(list_of_dfs, ignore_index=True)

    df_p1.to_feather("period2/processed/p1/" + str(x) + ".feather")

    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took "
          + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 1 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.3805290659268697 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.3821327328681946 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.3774236003557841 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.3989825487136841 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.4044086178143819 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.3743662516276042 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.35904823541641234 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.3701878706614176 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.3681028644243876 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.3791684826215108 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.3703977306683858 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.37550516923268634 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.3627315998077393 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.3929146488507589 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.38792408307393395 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.4030084172884623 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.3897182504336039 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.38905912240346274 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.37968841393788655 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.36325846513112386 minutes.
Processing all of periode 1 Tweets took 7.608766682942709 minutes.
```

In [108]:

```
df_p1 = pd.concat([pd.read_feather("period2/processed/p1/" + file) for file in os.listdir("period2/processed/p1/") if file != ".DS_Store"], ignore_index=True)
df_p1.to_feather("period2/processed/df_p1.feather")
```

In [112]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 2
files_p2 = os.listdir("period2/p2/")
#p2 is periode 2

file_paths = ["period2/p2/" + file for file in files_p2 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p2 = pd.concat(list_of_dfs, ignore_index=True)

    df_p2.to_feather("period2/processed/p2/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked))
          + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 2 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.29411064783732094 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.27773729562759397 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.2678303837776184 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.2705013155937195 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.269218115011851 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.3010012149810791 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.37487616936365764 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.4172863006591797 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.870514182249705 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.48668224811553956 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.47710638443628944 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.3389262000719706 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.3318796475728353 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.3942950367927551 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.32723944981892905 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.32073721488316853 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.3511858979860942 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.3188134511311849 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.2929485003153483 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.2560127456982931 minutes.
Processing all of periode 2 Tweets took 7.239109234015147 minutes.
```

In [113]:
```
df_p2 = pd.concat([pd.read_feather("period2/processed/p2/" + file) for file in os.listdir("period2/processed/p2/") if file != ".DS_Store"], ignore_index=True)
df_p2.to_feather("period2/processed/df_p2.feather")
```

In [114]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 2
files_p3 = os.listdir("period2/p3/")
#p2 is periode 2

file_paths = ["period2/p3/" + file for file in files_p3 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p3 = pd.concat(list_of_dfs, ignore_index=True)

    df_p3.to_feather("period2/processed/p3/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 3 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.26318047046661375 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.2707620660463969 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.263373863697052 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.2723457852999369 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.2742803494135539 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.29086568355560305 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.2748298366864522 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.275376816590627 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.2795538862546285 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.2724348505338033 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.27433271408081056 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.28211601575215656 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.2707836667696635 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.26975741386413576 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.2741008798281352 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.2702767332394918 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.2798195997873942 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.2601745843887329 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.2677171349525452 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.2564953009287516 minutes.
Processing all of periode 3 Tweets took 5.442729886372884 minutes.
```

In [115]:
```
df_p3 = pd.concat([pd.read_feather("period2/processed/p3/" + file) for file in os.listdir("period2/processed/p3/") if file != ".DS_Store"], ignore_index=True)
df_p3.to_feather("period2/processed/df_p3.feather")
```

In [116]:

```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 2
##NEED TO FILL IN INFO FOR FILE PATH AND WHERE TO SAVE FEATHER FILE
files_p4 = os.listdir("period2/p4/")
#p4 is periode 4

file_paths = ["period2/p4/" + file for file in files_p4 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p4 = pd.concat(list_of_dfs, ignore_index=True)

    df_p4.to_feather("period2/processed/p4/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 4 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.1692593812942505 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.17615045309066774 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.16569939851760865 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.17855141560236612 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.17557881673177084 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.1749296506245931 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.17536856333414713 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.1595461328824361 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.16363495190938313 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.160619314511617 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.16064629952112833 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.1638311505317688 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.16077491442362468 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.16453123490015667 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.1600197990735372 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.16730533043543497 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.16721506516138712 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.15885743300120037 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.16528411706288657 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.16824509700139365 minutes.
Processing all of periode 4 Tweets took 3.3362359166145326 minutes.
```

In [117]:

```
df_p4 = pd.concat([pd.read_feather("period2/processed/p4/" + file) for file in os.listdir("period2/processed/p4/") if file != ".DS_Store"], ignore_index=True)
df_p4.to_feather("period2/processed/df_p4.feather")
```

In [118]:

```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 2
##NEED TO FILL IN INFO FOR FILE PATH AND WHERE TO SAVE FEATHER FILE
files_p5 = os.listdir("period2/p5/")
#p5 is periode 5

file_paths = ["period2/p5/" + file for file in files_p5 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p5 = pd.concat(list_of_dfs, ignore_index=True)

    df_p5.to_feather("period2/processed/p5/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 5 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.18372036616007487 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.18432403405507405 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.19164421558380126 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.18567768335342408 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.18732287089029948 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.1862501343091329 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.1848703344662984 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.18065814971923827 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.17132650216420492 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.175072713692983 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.1752854347229004 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.16957389911015827 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.17745131651560467 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.17858975330988566 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.17795896530151367 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.17188024918238323 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.17553898493448894 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.17411046822865803 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.16984135309855145 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.17680996656417847 minutes.
Processing all of periode 5 Tweets took 3.578013265132904 minutes.
```

In [119]:

```
df_p5 = pd.concat([pd.read_feather("period2/processed/p5/" + file) for file in os.listdir("period2/processed/p5/") if file != ".DS_Store"], ignore_index=True)
df_p5.to_feather("period2/processed/df_p5.feather")
```

In [120]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 2
##NEED TO FILL IN INFO FOR FILE PATH AND WHERE TO SAVE FEATHER FILE
files_p6 = os.listdir("period2/p6/")
#p6 is periode 6

file_paths = ["period2/p6/" + file for file in files_p6 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p6 = pd.concat(list_of_dfs, ignore_index=True)

    df_p6.to_feather("period2/processed/p6/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 6 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.2580784360567729 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.23082664807637532 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.24843322038650512 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.2314085006713867 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.24955050150553384 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.24908498128255208 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.23550386826197306 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.24094566504160564 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.24056650002797444 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.2406233310699463 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.2389382799466451 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.2488291064898173 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.23551920255025227 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.2755364179611206 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.2515408476193746 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.24894758462905883 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.2841283162434896 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.28773515224456786 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.2368334174156189 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.2375538984934489 minutes.
Processing all of periode 6 Tweets took 4.970709935824076 minutes.
```

In [121]:
```
df_p6 = pd.concat([pd.read_feather("period2/processed/p6/" + file) for file in os.listdir("period2/processed/p6/") if file != ".DS_Store"], ignore_index=True)
df_p6.to_feather("period2/processed/df_p6.feather")
```

In [122]:

```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 2
##NEED TO FILL IN INFO FOR FILE PATH AND WHERE TO SAVE FEATHER FILE
files_p7 = os.listdir("period2/p7/")
#p7 is periode 7

file_paths = ["period2/p7/" + file for file in files_p7 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p7 = pd.concat(list_of_dfs, ignore_index=True)

    df_p7.to_feather("period2/processed/p7/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 7 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.3561914841334025 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.35240312019983927 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.3527329206466675 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.38905251820882164 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.47023186683654783 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.4048752705256144 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.3936568816502889 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.43362803061803185 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.4126247008641561 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.37633511622746785 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.4584801991780599 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.4344798962275187 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.5190514524777731 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.46746379931767784 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.4219975193341573 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.37829893032709755 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.3689325213432312 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.383842666943868 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.4452372193336487 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.5560832858085633 minutes.
Processing all of periode 7 Tweets took 8.375742367903392 minutes.
```

In [123]:

```
df_p7 = pd.concat([pd.read_feather("period2/processed/p7/" + file) for file in os.listdir("period2/processed/p7/") if file != ".DS_Store"], ignore_index=True)
df_p7.to_feather("period2/processed/df_p7.feather")
```

In [124]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 8
files_p8 = os.listdir("period2/p8/")
#p8 is periode 8

file_paths = ["period2/p8/" + file for file in files_p8 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p8 = pd.concat(list_of_dfs, ignore_index=True)

    df_p8.to_feather("period2/processed/p8/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 8 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.491715681552887 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.3768309315045675 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.3778107166290283 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.3747175852457682 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.38532941341400145 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.3787106513977051 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.37660321791966755 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.3816309332847595 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.3849697470664978 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.37975003321965534 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.4074544866879781 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.47946568330128986 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.5283321460088094 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.4340378483136495 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.503952153523763 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.4967235525449117 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.41664761702219644 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.37098429997762045 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.39812846581141154 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.3769906838734945 minutes.
Processing all of periode 8 Tweets took 8.320911983648935 minutes.
```

In [125]:
```
df_p8 = pd.concat([pd.read_feather("period2/processed/p8/" + file) for file in os.listdir("period2/processed/p8/") if file != ".DS_Store"], ignore_index=True)
df_p8.to_feather("period2/processed/df_p8.feather")
```

In [126]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 9
files_p9 = os.listdir("period2/p9/")
#p9 is periode 9

file_paths = ["period2/p9/" + file for file in files_p9 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p9 = pd.concat(list_of_dfs, ignore_index=True)

    df_p9.to_feather("period2/processed/p9/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 9 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.224423352877299 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.21581885019938152 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.24724603494008382 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.21472608645757038 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.2055819511413574 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.22586426337560017 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.21636376778284708 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.20967015027999877 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.2256102720896403 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.23079158067703248 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.2197160045305888 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.20964621702829997 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.19760975042978923 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.2106182336807251 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.28022764523824056 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.2475056807200114 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.2062725027402242 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.2131189783414205 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.2195224682490031 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.20271245241165162 minutes.
Processing all of periode 9 Tweets took 4.423218814531962 minutes.
```

In [127]:
```
df_p9 = pd.concat([pd.read_feather("period2/processed/p9/" + file) for file in os.listdir("period2/processed/p9/") if file != ".DS_Store"], ignore_index=True)
df_p9.to_feather("period2/processed/df_p9.feather")
```

In [128]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 10
files_p10 = os.listdir("period2/p10/")
#p10 is periode 10

file_paths = ["period2/p10/" + file for file in files_p10 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p10 = pd.concat(list_of_dfs, ignore_index=True)

    df_p10.to_feather("period2/processed/p10/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 10 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.2238613804181417 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.20878036816914877 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.2108439803123474 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.20895096858342488 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.23723848263422648 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.2876592516899109 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.28275150060653687 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.312036136786143 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.33463843663533527 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.25893423159917195 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.26980326970418295 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.30216362873713176 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.30402016242345176 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.23099937041600546 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.27719460328420004 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.2350519259770711 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.24555398225784303 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.2873963793118795 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.2834599852561951 minutes.
```

In [129]:
```
df_p10 = pd.concat([pd.read_feather("period2/processed/p10/" + file) for file in os.listdir("period2/processed/p10/") if file != ".DS_Store"], ignore_index=True)
df_p10.to_feather("period2/processed/df_p10.feather")
```

In [130]:
```
#Create an empty Pandas dataframe to fill up with the information one-by-one for periode 11
files_p11 = os.listdir("period2/p11/")
#p11 is periode 11

file_paths = ["period2/p11/" + file for file in files_p11 if file != ".DS_Store"]
chunked = np.array_split(file_paths, 20)

# For each of these chunks of JSON files, we do the same as above.
start_all = time.time()
for x in range(0, len(chunked)):
    start = time.time()

    file_paths = chunked[x]

    with Pool(cpu_number) as pool:
        json_all = pool.map(json.loads, [open(file).read() for file in file_paths])

    json_all = functools.reduce(operator.iconcat, json_all, [])

    with Pool(cpu_number) as pool:
        list_of_dfs = pool.map(extract_info, [tweet for tweet in json_all])

    # Then turn the list of dataframes into one big one
    df_p11 = pd.concat(list_of_dfs, ignore_index=True)

    df_p11.to_feather("period2/processed/p11/" + str(x) + ".feather")
    end = time.time()
    print("Extracting Tweet information for chunk " + str(x) + " of " + str(len(chunked)) + " took " + str((end-start)/60) + " minutes.")

end_all = time.time()
print("Processing all of periode 11 Tweets took " + str((end_all-start_all)/60) + " minutes.")
```

```
Extracting Tweet information for chunk 0 of 20 took 0.19918924967447918 minutes.
Extracting Tweet information for chunk 1 of 20 took 0.17716166973114014 minutes.
Extracting Tweet information for chunk 2 of 20 took 0.17958426872889202 minutes.
Extracting Tweet information for chunk 3 of 20 took 0.18636800050735475 minutes.
Extracting Tweet information for chunk 4 of 20 took 0.1703022321065267 minutes.
Extracting Tweet information for chunk 5 of 20 took 0.17746260166168212 minutes.
Extracting Tweet information for chunk 6 of 20 took 0.1647436777750651 minutes.
Extracting Tweet information for chunk 7 of 20 took 0.17095585266749064 minutes.
Extracting Tweet information for chunk 8 of 20 took 0.1748868981997172 minutes.
Extracting Tweet information for chunk 9 of 20 took 0.23637441794077554 minutes.
Extracting Tweet information for chunk 10 of 20 took 0.20199259916941326 minutes.
Extracting Tweet information for chunk 11 of 20 took 0.19775651693344115 minutes.
Extracting Tweet information for chunk 12 of 20 took 0.17311960061391193 minutes.
Extracting Tweet information for chunk 13 of 20 took 0.22826234896977743 minutes.
Extracting Tweet information for chunk 14 of 20 took 0.2213940183321635 minutes.
Extracting Tweet information for chunk 15 of 20 took 0.18566078344980877 minutes.
Extracting Tweet information for chunk 16 of 20 took 0.22761038541793824 minutes.
Extracting Tweet information for chunk 17 of 20 took 0.16802506844202678 minutes.
Extracting Tweet information for chunk 18 of 20 took 0.17689866622289022 minutes.
Extracting Tweet information for chunk 19 of 20 took 0.1629657506942749 minutes.
Processing all of periode 11 Tweets took 3.780820949872335 minutes.
```

In [131]:
```
df_p11 = pd.concat([pd.read_feather("period2/processed/p11/" + file) for file in os.listdir("period2/processed/p11/") if file != ".DS_Store"], ignore_index=True)
df_p11.to_feather("period2/processed/df_p11.feather")
```

In [132]:
```
df_p1
```

Out[132]:

	user_handle	user_id	tweet_id	location_full_name	location_country	location_country_code	coordinates	tweet_time	tweet_type	tweet_text	reply_user	qu_tweet_text	qu_
0	PaulSpackman1	1446248749	1110201364699860994	None	None	None	None	2019-03-25 15:26:46	Retweet	None	None	None	N
1	alien_sasquatch	905937529320259584	1110193339221970947	None	None	None	None	2019-03-25 14:54:53	Re_Quote	None	None	@cactus_furious No democracy here. But we can ...	Devins
2	richardcookson	97192926	1110192714371530754	None	None	None	None	2019-03-25 14:52:24	Retweet	None	None	None	N
3	FarageFor	1068445384278769664	1110192871213281286	None	None	None	None	2019-03-25 14:53:01	Tweet	@burge2u @neskatxa @MickLivesey @Andrew_Adonis...	burge2u	None	N
4	LynAraucana	2317666772	1110190400600526848	None	None	None	None	2019-03-25 14:43:12	Retweet	None	None	None	N
...	...	...	...	...	...	...	...	...	...	...	...	...	
165240	LilianGreenwood	20148039	1111241067192557569	None	None	None	None	2019-03-28 12:18:10	Quote	Labour royalty. Pure class!	None	WATCH: Margaret Beckett on #r4Today - the Beck...	peoplesvot
165241	tnewtondunn	20598137	1111240261512900608	None	None	None	None	2019-03-28 12:14:58	Tweet	General confusion now about whether MV3 is tom...	None	None	N
165242	mkhuller	2316317535	1111242864661544961	None	None	None	None	2019-03-28 12:25:19	Retweet	None	None	None	N
165243	EmmaMarze	2938732793	1111239345283973120	None	None	None	None	2019-03-28 12:11:20	Re_Quote	None	None	While SW1 talks #Brexit, I will be on #BBCNews...	BBCMarkEa
165244	UKreality	2447664324	1111241624191930371	None	None	None	None	2019-03-28 12:20:23	Retweet	None	None	None	N

165245 rows × 18 columns

In [133]:
```
print(len(df_p1)+len(df_p2)+len(df_p3)+len(df_p4)+len(df_p5)+len(df_p6)+len(df_p7)+len(df_p8)+len(df_p9)+len(df_p10)+len(df_p11))
```

```
1197987
```

In [134]:
```
print("df_p1:" + str(len(df_p1)))
print("df_p2:" + str(len(df_p2)))
print("df_p3:" + str(len(df_p3)))
print("df_p4:" + str(len(df_p4)))
print("df_p5:" + str(len(df_p5)))
print("df_p6:" + str(len(df_p6)))
print("df_p7:" + str(len(df_p7)))
print("df_p8:" + str(len(df_p8)))
print("df_p9:" + str(len(df_p9)))
print("df_p10:" + str(len(df_p10)))
print("df_p11:" + str(len(df_p11)))
```

```
df_p1:165245
df_p2:117750
df_p3:119249
df_p4:71750
df_p5:76749
df_p6:104000
df_p7:153249
df_p8:153248
df_p9:82999
df_p10:87748
df_p11:66000
```

In [135]:
```
1 df_p4
```

Out[135]:

	user_handle	user_id	tweet_id	location_full_name	location_country	location_country_code	coordinates	tweet_time	tweet_type	tweet_text	reply_user	qu_tweet_text	qu_user	q
0	janetm50723412	1120079106027216896	1143118832942424064	None	None	None	None	2019-06-24 11:29:02	Retweet	None	None	None	None	
1	conndec	22237401	1143116936496263170	None	None	None	None	2019-06-24 11:21:30	Retweet	None	None	None	None	
2	aHumanEvolution	948446776377462785	1143099574430683136	None	None	None	None	2019-06-24 10:12:30	Re_Quote	None	None	Publicly I have always remained staunchly apol...	DoctorChristian	
3	Paulabaena26	234791423	1143115243834527746	None	None	None	None	2019-06-24 11:14:46	Tweet	Rusia no usó Facebook para influir en el refer...	None	None	None	
4	BritanniaNew	970985923746979840	1143112248312942592	None	None	None	None	2019-06-24 11:02:52	Re_Quote	None	None	A junior Gov't minister will claim on BBC Pano...	Mike_Fabricant	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
71745	kenblaber	472199345	1136011364370190336	None	None	None	None	2019-06-04 20:46:29	Retweet	None	None	None	None	
71746	TrysM	57642884	1136012139108491265	None	None	None	None	2019-06-04 20:49:34	Retweet	None	None	None	None	
71747	ireneusz18	133297270	1136013383130636290	None	None	None	None	2019-06-04 20:54:31	Retweet	None	None	None	None	
71748	jotcd	880061763684892672	1136006362339663873	None	None	None	None	2019-06-04 20:26:37	Quote	He will sell our prode and the envy of the wor...	None	Mike Greene, Brexit party candidate Peterborou...	MikeMar82888097	
71749	SpainNick	404877200	1136009366195310593	None	None	None	None	2019-06-04 20:38:33	Retweet	None	None	None	None	

71750 rows × 18 columns

In [ ]:
```
1
```

In [ ]:
```
1
```

In [2]:
```
## created the 26.05.2020
```

In [3]:
```
import os
import pandas as pd
import numpy as np

import time
import random
import json

from multiprocessing import Pool
import operator
import functools
from multiprocessing import cpu_count
cpu_number = cpu_count() - 1
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import networkx as nx
```

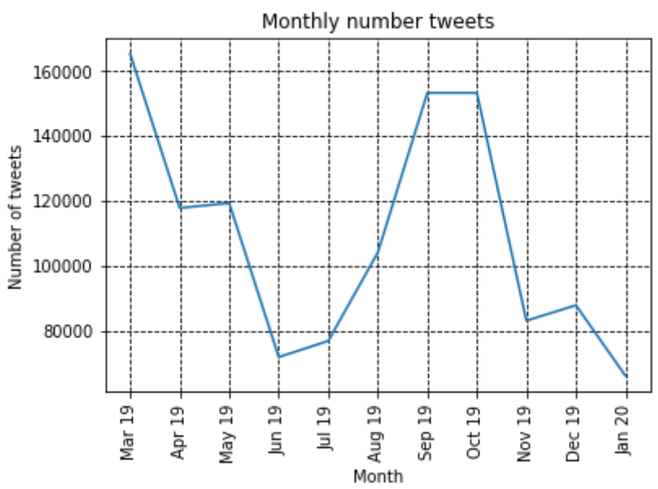
In [4]:
```
# read files - open op all periodes
df_p1 = pd.read_feather("period2/processed/df_p1.feather")
df_p2 = pd.read_feather("period2/processed/df_p2.feather")
df_p3 = pd.read_feather("period2/processed/df_p3.feather")
df_p4 = pd.read_feather("period2/processed/df_p4.feather")
df_p5 = pd.read_feather("period2/processed/df_p5.feather")
df_p6 = pd.read_feather("period2/processed/df_p6.feather")
df_p7 = pd.read_feather("period2/processed/df_p7.feather")
df_p8 = pd.read_feather("period2/processed/df_p8.feather")
df_p9 = pd.read_feather("period2/processed/df_p9.feather")
df_p10 = pd.read_feather("period2/processed/df_p10.feather")
df_p11 = pd.read_feather("period2/processed/df_p11.feather")
```

In [5]:
```
print("df_p1:" + str(len(df_p1)))
print("df_p2:" + str(len(df_p2)))
print("df_p3:" + str(len(df_p3)))
print("df_p4:" + str(len(df_p4)))
print("df_p5:" + str(len(df_p5)))
print("df_p6:" + str(len(df_p6)))
print("df_p7:" + str(len(df_p7)))
print("df_p8:" + str(len(df_p8)))
print("df_p9:" + str(len(df_p9)))
print("df_p10:" + str(len(df_p10)))
print("df_p11:" + str(len(df_p11)))
```

```
df_p1:165245
df_p2:117750
df_p3:119249
df_p4:71750
df_p5:76749
df_p6:104000
df_p7:153249
df_p8:153248
df_p9:82999
df_p10:87748
df_p11:66000
```

In [6]:
```
period = ["Mar 19", "Apr 19", "May 19", "Jun 19", "Jul 19", "Aug 19", "Sep 19", "Oct 19", "Nov 19", "Dec 19", "Jan 20"]
tweet_month = [165245, 117750, 119249, 71750, 76749, 104000, 153249, 153249, 82999, 87748, 66000]
```

In [7]:
```
plt.plot(period, tweet_month,)
plt.ylabel('Number of tweets')
plt.xlabel('Month')
plt.title("Monthly number tweets")
plt.xticks(rotation=90)


plt.grid(color="k", linestyle="--")
plt.show()
```



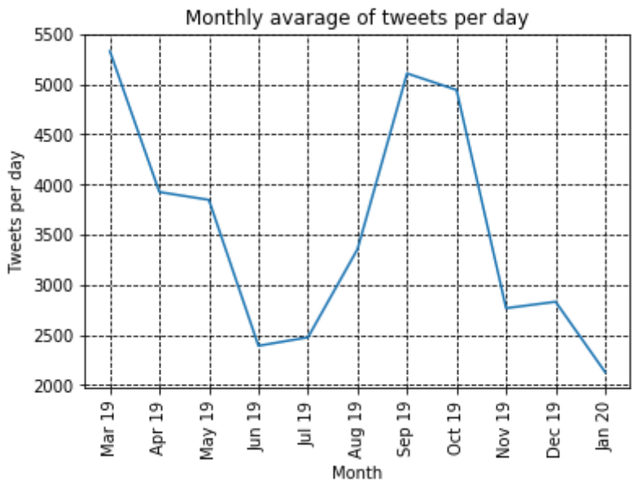
In [8]:
```
#df_p1 goes from the 1.03.2019 to 1.04.2019 (not including 1.04.2019).
#that means the p1 has 31 days

#df_p2 goes from the 1.04.2019 to 1.05.2019 (not including 1.05.2019).
#that means the p2 has 30 days

#df_p3 goes from the 1.05.2019 to 1.06.2019 (not including 1.06.2019).
#that means the p3 has 31 days

#df_p4 goes from the 1.06.2019 to 1.07.2019 (not including 1.07.2019).
#that means the p4 has 30 days

#df_p5 goes from the 1.07.2019 to 1.08.2019 (not including 1.08.2019).
#that means the p5 has 31 days

#df_p6 goes from the 1.08.2019 to 1.09.2019 (not including 1.09.2019).
#that means the p6 has 31 days

#df_p7 goes from the 1.09.2019 to 1.10.2019 (not including 1.10.2019).
#that means the p7 has 30 days

#df_p8 goes from the 1.10.2019 to 1.11.2019 (not including 1.11.2019).
#that means the p8 has 31 days.

#df_p9 goes from the 1.11.2019 to 1.12.2019 (not including 1.12.2019).
#that means the p9 has 30 days

#df_p10 goes from the 1.12.2019 to 1.01.2020 (not including 1.01.2020).
#that means the p10 has 31 days

#df_p10 goes from the 1.01.2020 to 31.01.2020 (including 31.01.2020).
#that means the p10 has 31 days
```

In [9]:
```
print("df_p1:" + str(len(df_p1)/31))
print("df_p2:" + str(len(df_p2)/30))
print("df_p3:" + str(len(df_p3)/31))
print("df_p4:" + str(len(df_p4)/30))
print("df_p5:" + str(len(df_p5)/31))
print("df_p6:" + str(len(df_p6)/31))
print("df_p7:" + str(len(df_p7)/30))
print("df_p8:" + str(len(df_p8)/31))
print("df_p9:" + str(len(df_p9)/30))
print("df_p10:" + str(len(df_p10)/31))
print("df_p11:" + str(len(df_p11)/31))
```

```
df_p1:5330.4838709677415
df_p2:3925.0
df_p3:3846.7419354838707
df_p4:2391.6666666666665
df_p5:2475.7741935483873
df_p6:3354.8387096774195
df_p7:5108.3
df_p8:4943.4838709677415
df_p9:2766.633333333333
df_p10:2830.5806451612902
df_p11:2129.032258064516
```

In [10]:
```
period = ["Mar 19", "Apr 19", "May 19", "Jun 19", "Jul 19", "Aug 19", "Sep 19", "Oct 19", "Nov 19", "Dec 19", "Jan 20"]
tweet_pday = [5330.48, 3925.0, 3846.74, 2391.67, 2475.77, 3354.84, 5108.3, 4943.48, 2766.63, 2830.58, 2129.03]
```

In [11]:
```
plt.plot(period, tweet_pday,)
plt.ylabel('Tweets per day')
plt.xlabel('Month')
plt.title("Monthly avarage of tweets per day")
plt.xticks(rotation=90)
plt.yticks([2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500], size=10)

plt.grid(color="k", linestyle="--")
plt.show()
```



In [12]:
```

print("number of different kind of tweets p1: " + str(df_p1['tweet_type'].value_counts()))
```

```
number of different kind of tweets p1: Retweet     88975
Tweet       33141
Re_Quote    30449
Quote       12680
Name: tweet_type, dtype: int64
```

In [13]:
```
p1_type=df_p1['tweet_type'].value_counts()
p2_type=df_p2['tweet_type'].value_counts()
p3_type=df_p3['tweet_type'].value_counts()
p4_type=df_p4['tweet_type'].value_counts()
p5_type=df_p5['tweet_type'].value_counts()
p6_type=df_p6['tweet_type'].value_counts()
p7_type=df_p7['tweet_type'].value_counts()
p8_type=df_p8['tweet_type'].value_counts()
p9_type=df_p9['tweet_type'].value_counts()
p10_type=df_p10['tweet_type'].value_counts()
p11_type=df_p11['tweet_type'].value_counts()
```

In [14]:
```
type(p1_type)
```

Out[14]:
```
pandas.core.series.Series
```

In [15]:
```
df_type = pd.DataFrame(columns=["p1","p2", "p3", "p4", "p5", "p6", "p7", "p8", "p9", "p10", "p11"])
df_type
```

Out[15]:

p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11

In [16]:
```
df_type["p1"] = p1_type
df_type["p2"] = p2_type
df_type["p3"] = p3_type
df_type["p4"] = p4_type
df_type["p5"] = p5_type
df_type["p6"] = p6_type
df_type["p7"] = p7_type
df_type["p8"] = p8_type
df_type["p9"] = p9_type
df_type["p10"] = p10_type
df_type["p11"] = p11_type


df_type
```

Out[16]:

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
Retweet	88975	63408	66152	38277	40671	56928	84458	85218	45197	50578	35183
Tweet	33141	20903	21964	12935	14076	18951	26002	26458	14512	16294	12090
Re_Quote	30449	24735	21967	14773	15835	19686	30948	29836	16884	14212	13253
Quote	12680	8704	9166	5765	6167	8435	11841	11736	6406	6664	5474

In [17]:
```
df_type1 = df_type.transpose()
df_type1["all_quotes"] = df_type1["Re_Quote"] + df_type1["Quote"]
df_type1["all_post"] = df_type1["all_quotes"] + df_type1["Tweet"] + df_type1["Retweet"]
```

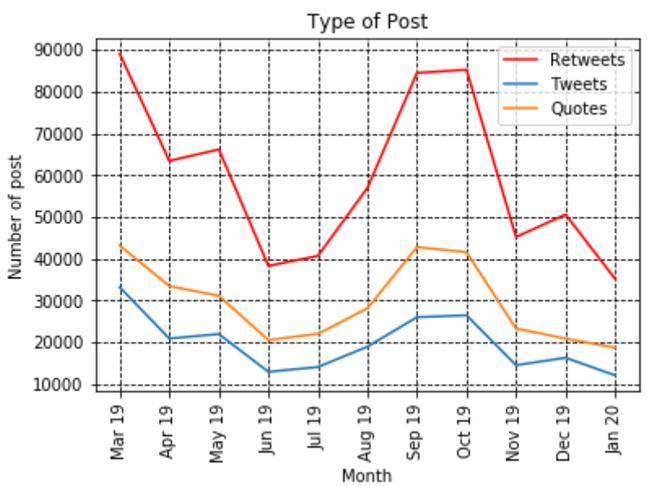
In [18]:
```
df_type1['total_post'] = sum(df_type1["all_quotes"] + df_type1["Tweet"] + df_type1["Retweet"])
df_type1['total_tweet'] = sum(df_type1["Tweet"])
df_type1['total_retweet'] = sum(df_type1["Retweet"])
df_type1['total_quotes'] = sum(df_type1["all_quotes"])
df_type1
```

Out[18]:

	Retweet	Tweet	Re_Quote	Quote	all_quotes	all_post	total_post	total_tweet	total_retweet	total_quotes
p1	88975	33141	30449	12680	43129	165245	1197987	217326	655045	325616
p2	63408	20903	24735	8704	33439	117750	1197987	217326	655045	325616
p3	66152	21964	21967	9166	31133	119249	1197987	217326	655045	325616
p4	38277	12935	14773	5765	20538	71750	1197987	217326	655045	325616
p5	40671	14076	15835	6167	22002	76749	1197987	217326	655045	325616
p6	56928	18951	19686	8435	28121	104000	1197987	217326	655045	325616
p7	84458	26002	30948	11841	42789	153249	1197987	217326	655045	325616
p8	85218	26458	29836	11736	41572	153248	1197987	217326	655045	325616
p9	45197	14512	16884	6406	23290	82999	1197987	217326	655045	325616
p10	50578	16294	14212	6664	20876	87748	1197987	217326	655045	325616
p11	35183	12090	13253	5474	18727	66000	1197987	217326	655045	325616

In [19]:
```
plt.plot(period, df_type1["Retweet"], color='red', label="Retweets")
plt.plot(period, df_type1["Tweet"], label="Tweets")
plt.plot(period, df_type1["all_quotes"], label="Quotes")
plt.grid(color="k", linestyle="--")
plt.xticks(rotation=90)
plt.xlabel('Month')
plt.ylabel('Number of post')
plt.title("Type of Post")
plt.legend()
```

Out[19]: <matplotlib.legend.Legend at 0xa41f68d50>



In [ ]:
```
1
```

In [2]:
```
#created 20.05.2020
#updated 26.05.2020
```

In [2]:
```
import os
import pandas as pd
import numpy as np

import time
import random
import json

from multiprocessing import Pool
import operator
import functools
from multiprocessing import cpu_count
cpu_number = cpu_count() - 1
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import networkx as nx
# read files - open op all periodes
df_p1 = pd.read_feather("period2/processed/df_p1.feather")
df_p2 = pd.read_feather("period2/processed/df_p2.feather")
df_p3 = pd.read_feather("period2/processed/df_p3.feather")
df_p4 = pd.read_feather("period2/processed/df_p4.feather")
df_p5 = pd.read_feather("period2/processed/df_p5.feather")
df_p6 = pd.read_feather("period2/processed/df_p6.feather")
df_p7 = pd.read_feather("period2/processed/df_p7.feather")
df_p8 = pd.read_feather("period2/processed/df_p8.feather")
df_p9 = pd.read_feather("period2/processed/df_p9.feather")
df_p10 = pd.read_feather("period2/processed/df_p10.feather")
df_p11 = pd.read_feather("period2/processed/df_p11.feather")
```

In [ ]:
```

```

In [4]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#-----------------------------period 1---------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [5]:
```
df_p1_nxall = df_p1[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p1_nxall = df_p1_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p1_nxall_nan = df_p1_nxall[df_p1_nxall['edge'].notna()]
Gp1al = nx.from_pandas_edgelist(df_p1_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp1al.add_nodes_from(df_p1_nxall["user_handle"])
Gp1al.remove_edges_from(nx.selfloop_edges(Gp1al))

print(nx.info(Gp1al))
print(len(list(nx.isolates(Gp1al))))
```

```
Name:
Type: DiGraph
Number of nodes: 111922
Number of edges: 169164
Average in degree:   1.5114
Average out degree:   1.5114
12754
```

In [6]:
```
print(len(Gp1al.nodes))
print(len(Gp1al.edges))
```

```
111922
169164
```

In [7]:
```
print(len(nx.k_core(Gp1al,k=8).nodes()))
print(len(nx.k_core(Gp1al,k=5).nodes()))
Gp1al_k8= list(nx.k_core(Gp1al,k=8).nodes())
Gp1al_k5= list(nx.k_core(Gp1al,k=5).nodes())
```

```
3029
7344
```

In [8]:
```
in_dcs = nx.in_degree_centrality(Gp1al)
Gp1_in_dcs_k8 = [in_dcs[n] for n in Gp1al_k8]
Gp1_in_dcs_k5 = [in_dcs[n] for n in Gp1al_k5]
print(sum(Gp1_in_dcs_k8))
print(sum(Gp1_in_dcs_k5))
```

```
0.9280742666702467
1.0880799849894274
```

In [9]:
```
out_dcs = nx.out_degree_centrality(Gp1al)
Gp1_out_dcs_k8 = [out_dcs[n] for n in Gp1al_k8]
Gp1_out_dcs_k5 = [out_dcs[n] for n in Gp1al_k5]
print(sum(Gp1_out_dcs_k8))
print(sum(Gp1_out_dcs_k5))
```

```
0.32671259191751373
0.5386120567185814
```

In [10]:
```
nx.density(Gp1al)
```

Out[10]:
```
1.3504574395805836e-05
```

In [11]:
```
start = time.time()
Gp1_cc = nx.closeness_centrality(Gp1al)
Gp1_cc_k8 = [Gp1_cc[n] for n in Gp1al_k8]
Gp1_cc_k5 = [Gp1_cc[n] for n in Gp1al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p1 graph closeness centrality.")
print("closeness centrality avarage for the graph 1 is: " + str(sum(Gp1_cc.values())/len(Gp1_cc)))
print("closeness centrality avarage for the graph 1 (k=8) is: " + str(sum(Gp1_cc_k8)/len(Gp1_cc_k8)))
print("closeness centrality avarage for the graph 1 (k=5) is: " + str(sum(Gp1_cc_k5)/len(Gp1_cc_k5)))
```

```
It took 4.1239738663037615 minutes to calculate p1 graph closeness centrality.
closeness centrality avarage for the graph 1 is: 0.0003137046911926254
closeness centrality avarage for the graph 1 (k=8) is: 0.004490115914100305
closeness centrality avarage for the graph 1 (k=5) is: 0.0028318158750315157
```

In [ ]:
```

```

In [12]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#-----------------------------period 2----------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [13]:
```
df_p2_nxall = df_p2[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p2_nxall = df_p2_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p2_nxall_nan = df_p2_nxall[df_p2_nxall['edge'].notna()]
Gp2al = nx.from_pandas_edgelist(df_p2_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp2al.add_nodes_from(df_p2_nxall["user_handle"])
Gp2al.remove_edges_from(nx.selfloop_edges(Gp2al))

print(nx.info(Gp2al))
print(len(list(nx.isolates(Gp2al))))
```

```
Name:
Type: DiGraph
Number of nodes: 77698
Number of edges: 127208
Average in degree:   1.6372
Average out degree:   1.6372
7025
```

In [14]:
```
#just to check that the amount iof nodes are correct, i will check how many unique twitter users there are in period 2 data.
print("number of different kind of tweet: " + str(df_p2['tweet_type'].value_counts()))
df_p2_unique = df_p2[["user_handle", "reply_user", "qu_user", "retweeted_user"]]
df_p2_unique = df_p2_unique.melt(var_name="type", value_name="edge")
df_p2_unique_user = df_p2_unique['edge'].unique()
print("the number of unqiue user in period 2 df are: " + str(len(df_p2_unique_user)))
print("it should be the same number as nodes in the "
      "graph, minus one, as nan.value is counted as an unique user, "
      "which is: " + str(len(Gp2al.nodes())))

```

```
number of different kind of tweet: Retweet     63408
Re_Quote    24735
Tweet       20903
Quote        8704
Name: tweet_type, dtype: int64
the number of unqiue user in period 2 df are: 77699
it should be the same number as nodes in the graph, minus one, as nan.value is counted as an unique user, which is: 77698
```

In [15]:
```
print(len(nx.k_core(Gp2al,k=8).nodes()))
print(len(nx.k_core(Gp2al,k=5).nodes()))
Gp2al_k8= list(nx.k_core(Gp2al,k=8).nodes())
Gp2al_k5= list(nx.k_core(Gp2al,k=5).nodes())
```

```
2455
5883
```

In [16]:
```
in_dcs = nx.in_degree_centrality(Gp2al)
Gp2_in_dcs_k8 = [in_dcs[n] for n in Gp2al_k8]
Gp2_in_dcs_k5 = [in_dcs[n] for n in Gp2al_k5]
print(sum(Gp2_in_dcs_k8))
print(sum(Gp2_in_dcs_k5))
```

```
1.0314040439141772
1.2221192581438025
```

In [17]:
```
out_dcs = nx.out_degree_centrality(Gp2al)
Gp2_out_dcs_k8 = [out_dcs[n] for n in Gp2al_k8]
Gp2_out_dcs_k5 = [out_dcs[n] for n in Gp2al_k5]
print(sum(Gp2_out_dcs_k8))
print(sum(Gp2_out_dcs_k5))
```

```
0.36816093285454926
0.6095859556996904
```

In [18]:
```
nx.density(Gp2al)
```

Out[18]:
```
2.1071736862622255e-05
```

In [19]:
```
start = time.time()
Gp2_cc = nx.closeness_centrality(Gp2al)
Gp2_cc_k8 = [Gp2_cc[n] for n in Gp2al_k8]
Gp2_cc_k5 = [Gp2_cc[n] for n in Gp2al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p2 graph closeness centrality.")
print("closeness centrality avarage for the graph 2 is: " + str(sum(Gp2_cc.values())/len(Gp2_cc)))
print("closeness centrality avarage for the graph 2 (k=8) is: " + str(sum(Gp2_cc_k8)/len(Gp2_cc_k8)))
print("closeness centrality avarage for the graph 2 (k=5) is: " + str(sum(Gp2_cc_k5)/len(Gp2_cc_k5)))
```

```
It took 2.799206801255544 minutes to calculate p2 graph closeness centrality.
closeness centrality avarage for the graph 2 is: 0.0005317720847184066
closeness centrality avarage for the graph 2 (k=8) is: 0.006020899929723154
closeness centrality avarage for the graph 2 (k=5) is: 0.0039764616169143206
```

In [20]:
```
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#------------------------------period 3--------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
```

In [21]:
```
df_p3_nxall = df_p3[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p3_nxall = df_p3_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p3_nxall_nan = df_p3_nxall[df_p3_nxall['edge'].notna()]
Gp3al = nx.from_pandas_edgelist(df_p3_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp3al.add_nodes_from(df_p3_nxall["user_handle"])
Gp3al.remove_edges_from(nx.selfloop_edges(Gp3al))

print(nx.info(Gp3al))
print(len(list(nx.isolates(Gp3al))))
```

```
Name:
Type: DiGraph
Number of nodes: 77436
Number of edges: 127547
Average in degree:   1.6471
Average out degree:   1.6471
5684
```

In [22]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp3al,k=8).nodes())))
Gp3al_k8= list(nx.k_core(Gp3al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp3al)
Gp3_in_dcs_k8 = [in_dcs[n] for n in Gp3al_k8]
print("k8 in degree centrality: " + str(sum(Gp3_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp3al)
Gp3_out_dcs_k8 = [out_dcs[n] for n in Gp3al_k8]
print("k8 out degree centrality: " + str(sum(Gp3_out_dcs_k8)))
```

```
no. of nodes, k=8,: 2352
k8 in degree centrality: 1.0354620003874258
k8 out degree centrality: 0.36413766384709834
```

In [23]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp3al,k=5).nodes())))
Gp3al_k5= list(nx.k_core(Gp3al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp3al)
Gp3_in_dcs_k5 = [in_dcs[n] for n in Gp3al_k5]
print("k5 in degree centrality: " + str(sum(Gp3_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp3al)
Gp3_out_dcs_k5 = [out_dcs[n] for n in Gp3al_k5]
print("k5 out degree centrality: " + str(sum(Gp3_out_dcs_k5)))
```

```
no. of nodes, k=5,: 5719
k5 in degree centrality: 1.2193194291986942
k5 out degree centrality: 0.6056047007167459
```

In [24]:
```
nx.density(Gp3al)
```

Out[24]:
```
2.1271104162509283e-05
```

In [25]:
```
start = time.time()
Gp3_cc = nx.closeness_centrality(Gp3al)
Gp3_cc_k8 = [Gp3_cc[n] for n in Gp3al_k8]
Gp3_cc_k5 = [Gp3_cc[n] for n in Gp3al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p3 graph closeness centrality.")
print("closeness centrality avarage for the graph 3 is: " + str(sum(Gp3_cc.values())/len(Gp3_cc)))
print("closeness centrality avarage for the graph 3 (k=8) is: " + str(sum(Gp3_cc_k8)/len(Gp3_cc_k8)))
print("closeness centrality avarage for the graph 3 (k=5) is: " + str(sum(Gp3_cc_k5)/len(Gp3_cc_k5)))
```

```
It took 3.670165248711904 minutes to calculate p3 graph closeness centrality.
closeness centrality avarage for the graph 3 is: 0.0007141188322691499
closeness centrality avarage for the graph 3 (k=8) is: 0.007428644281034676
closeness centrality avarage for the graph 3 (k=5) is: 0.005034785730120513
```

In [26]:
```
##EXTRA
```

In [27]:
```
#just to check that the amount of nodes are correct, i will check how many unique twitter users there are in period 3 data.
print("number of different kind of tweet: " + str(df_p3['tweet_type'].value_counts()))
df_p3_unique = df_p3[["user_handle", "reply_user", "qu_user", "retweeted_user"]]
df_p3_unique = df_p3_unique.melt(var_name="type", value_name="edge")
df_p3_unique_user = df_p3_unique['edge'].unique()
print("the number of unqiue user in period 3 df are: " + str(len(df_p3_unique_user)))
print("it should be the same number as nodes in the "
      "graph, minus one, as nan.value is counted as an unique user, "
      "which is: " + str(len(Gp3al.nodes())))
```

```
number of different kind of tweet: Retweet     66152
Re_Quote    21967
Tweet       21964
Quote        9166
Name: tweet_type, dtype: int64
the number of unqiue user in period 3 df are: 77437
it should be the same number as nodes in the graph, minus one, as nan.value is counted as an unique user, which is: 77436
```

In [28]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#-----------------------------period 4----------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [29]:
```
df_p4_nxall = df_p4[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p4_nxall = df_p4_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p4_nxall_nan = df_p4_nxall[df_p4_nxall['edge'].notna()]
Gp4al = nx.from_pandas_edgelist(df_p4_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp4al.add_nodes_from(df_p4_nxall["user_handle"])
Gp4al.remove_edges_from(nx.selfloop_edges(Gp4al))

print(nx.info(Gp4al))
print(len(list(nx.isolates(Gp4al))))
```

```
Name:
Type: DiGraph
Number of nodes: 52304
Number of edges: 78970
Average in degree:   1.5098
Average out degree:   1.5098
3344
```

In [30]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp4al,k=8).nodes())))
Gp4al_k8= list(nx.k_core(Gp4al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp4al)
Gp4_in_dcs_k8 = [in_dcs[n] for n in Gp4al_k8]
print("k8 in degree centrality: " + str(sum(Gp4_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp4al)
Gp4_out_dcs_k8 = [out_dcs[n] for n in Gp4al_k8]
print("k8 out degree centrality: " + str(sum(Gp4_out_dcs_k8)))
```

```
no. of nodes, k=8,: 617
k8 in degree centrality: 0.6648567003804763
k8 out degree centrality: 0.13016461770835308
```

In [31]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp4al,k=5).nodes())))
Gp4al_k5= list(nx.k_core(Gp4al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp4al)
Gp4_in_dcs_k5 = [in_dcs[n] for n in Gp4al_k5]
print("k5 in degree centrality: " + str(sum(Gp4_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp4al)
Gp4_out_dcs_k5 = [out_dcs[n] for n in Gp4al_k5]
print("k5 out degree centrality: " + str(sum(Gp4_out_dcs_k5)))
```

```
no. of nodes, k=5,: 3112
k5 in degree centrality: 1.0412404642181303
k5 out degree centrality: 0.4136091619983469
```

In [32]:
```
nx.density(Gp4al)
```

Out[32]:
```
2.8866932379986218e-05
```

In [33]:
```
start = time.time()
Gp4_cc = nx.closeness_centrality(Gp4al)
Gp4_cc_k8 = [Gp4_cc[n] for n in Gp4al_k8]
Gp4_cc_k5 = [Gp4_cc[n] for n in Gp4al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p4 graph closeness centrality.")
print("closeness centrality avarage for the graph 4 is: " + str(sum(Gp4_cc.values())/len(Gp4_cc)))
print("closeness centrality avarage for the graph 4 (k=8) is: " + str(sum(Gp4_cc_k8)/len(Gp4_cc_k8)))
print("closeness centrality avarage for the graph 4 (k=5) is: " + str(sum(Gp4_cc_k5)/len(Gp4_cc_k5)))

```

```
It took 0.4367289980252584 minutes to calculate p4 graph closeness centrality.
closeness centrality avarage for the graph 4 is: 0.00023345793270927677
closeness centrality avarage for the graph 4 (k=8) is: 0.0043346277776593525
closeness centrality avarage for the graph 4 (k=5) is: 0.0021240241352922184
```

In [34]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#------------------------------period 5--------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [35]:
```
df_p5_nxall = df_p5[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p5_nxall = df_p5_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p5_nxall_nan = df_p5_nxall[df_p5_nxall['edge'].notna()]
Gp5al = nx.from_pandas_edgelist(df_p5_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp5al.add_nodes_from(df_p5_nxall["user_handle"])
Gp5al.remove_edges_from(nx.selfloop_edges(Gp5al))

print(nx.info(Gp5al))
print(len(list(nx.isolates(Gp5al))))
```

```
Name:
Type: DiGraph
Number of nodes: 55953
Number of edges: 84031
Average in degree:   1.5018
Average out degree:   1.5018
3872
```

In [36]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp5al,k=8).nodes())))
Gp5al_k8= list(nx.k_core(Gp5al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp5al)
Gp5_in_dcs_k8 = [in_dcs[n] for n in Gp5al_k8]
print("k8 in degree centrality: " + str(sum(Gp5_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp5al)
Gp5_out_dcs_k8 = [out_dcs[n] for n in Gp5al_k8]
print("k8 out degree centrality: " + str(sum(Gp5_out_dcs_k8)))
```

```
no. of nodes, k=8,: 740
k8 in degree centrality: 0.6825850729196442
k8 out degree centrality: 0.15166571346868676
```

In [37]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp5al,k=5).nodes())))
Gp5al_k5= list(nx.k_core(Gp5al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp5al)
Gp5_in_dcs_k5 = [in_dcs[n] for n in Gp5al_k5]
print("k5 in degree centrality: " + str(sum(Gp5_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp5al)
Gp5_out_dcs_k5 = [out_dcs[n] for n in Gp5al_k5]
print("k5 out degree centrality: " + str(sum(Gp5_out_dcs_k5)))
```

```
no. of nodes, k=5,: 3369
k5 in degree centrality: 1.0431619959965621
k5 out degree centrality: 0.4256326851586987
```

In [38]:
```
nx.density(Gp5al)
```

Out[38]:
```
2.6841114213668064e-05
```

In [39]:
```
start = time.time()
Gp5_cc = nx.closeness_centrality(Gp5al)
Gp5_cc_k8 = [Gp5_cc[n] for n in Gp5al_k8]
Gp5_cc_k5 = [Gp5_cc[n] for n in Gp5al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p5 graph closeness centrality.")
print("closeness centrality avarage for the graph 5 is: " + str(sum(Gp5_cc.values())/len(Gp5_cc)))
print("closeness centrality avarage for the graph 5 (k=8) is: " + str(sum(Gp5_cc_k8)/len(Gp5_cc_k8)))
print("closeness centrality avarage for the graph 5 (k=5) is: " + str(sum(Gp5_cc_k5)/len(Gp5_cc_k5)))


```

```
It took 0.649120815594991 minutes to calculate p5 graph closeness centrality.
closeness centrality avarage for the graph 5 is: 0.0003012079458801734
closeness centrality avarage for the graph 5 (k=8) is: 0.005654028937749656
closeness centrality avarage for the graph 5 (k=5) is: 0.0027525467033317754
```

In [40]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#------------------------------period 6--------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [41]:
```
df_p6_nxall = df_p6[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p6_nxall = df_p6_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p6_nxall_nan = df_p6_nxall[df_p6_nxall['edge'].notna()]
Gp6al = nx.from_pandas_edgelist(df_p6_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp6al.add_nodes_from(df_p6_nxall["user_handle"])
Gp6al.remove_edges_from(nx.selfloop_edges(Gp6al))

print(nx.info(Gp6al))
print(len(list(nx.isolates(Gp6al))))
```

```
Name:
Type: DiGraph
Number of nodes: 71976
Number of edges: 111785
Average in degree:   1.5531
Average out degree:   1.5531
5399
```

In [42]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp6al,k=8).nodes())))
Gp6al_k8= list(nx.k_core(Gp6al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp6al)
Gp6_in_dcs_k8 = [in_dcs[n] for n in Gp6al_k8]
print("k8 in degree centrality: " + str(sum(Gp6_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp6al)
Gp6_out_dcs_k8 = [out_dcs[n] for n in Gp6al_k8]
print("k8 out degree centrality: " + str(sum(Gp6_out_dcs_k8)))
```

```
no. of nodes, k=8,: 1415
k8 in degree centrality: 0.8348176450156315
k8 out degree centrality: 0.22716220910038204
```

In [43]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp6al,k=5).nodes())))
Gp6al_k5= list(nx.k_core(Gp6al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp6al)
Gp6_in_dcs_k5 = [in_dcs[n] for n in Gp6al_k5]
print("k5 in degree centrality: " + str(sum(Gp6_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp6al)
Gp6_out_dcs_k5 = [out_dcs[n] for n in Gp6al_k5]
print("k5 out degree centrality: " + str(sum(Gp6_out_dcs_k5)))
```

```
no. of nodes, k=5,: 4829
k5 in degree centrality: 1.1090934352205646
k5 out degree centrality: 0.49544980896144514
```

In [44]:
```
nx.density(Gp6al)
```

Out[44]:
```
2.1578147136614523e-05
```

In [45]:
```
start = time.time()
Gp6_cc = nx.closeness_centrality(Gp6al)
Gp6_cc_k8 = [Gp6_cc[n] for n in Gp6al_k8]
Gp6_cc_k5 = [Gp6_cc[n] for n in Gp6al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p6 graph closeness centrality.")
print("closeness centrality avarage for the graph 6 is: " + str(sum(Gp6_cc.values())/len(Gp6_cc)))
print("closeness centrality avarage for the graph 6 (k=8) is: " + str(sum(Gp6_cc_k8)/len(Gp6_cc_k8)))
print("closeness centrality avarage for the graph 6 (k=5) is: " + str(sum(Gp6_cc_k5)/len(Gp6_cc_k5)))


```

```
It took 2.839759111404419 minutes to calculate p6 graph closeness centrality.
closeness centrality avarage for the graph 6 is: 0.0006261118661366462
closeness centrality avarage for the graph 6 (k=8) is: 0.008310565835184676
closeness centrality avarage for the graph 6 (k=5) is: 0.004930793374764635
```

In [46]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#-----------------------------period 7---------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [47]:
```
df_p7_nxall = df_p7[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p7_nxall = df_p7_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p7_nxall_nan = df_p7_nxall[df_p7_nxall['edge'].notna()]
Gp7al = nx.from_pandas_edgelist(df_p7_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp7al.add_nodes_from(df_p7_nxall["user_handle"])
Gp7al.remove_edges_from(nx.selfloop_edges(Gp7al))

print(nx.info(Gp7al))
print(len(list(nx.isolates(Gp7al))))
```

```
Name:
Type: DiGraph
Number of nodes: 98739
Number of edges: 166755
Average in degree:   1.6888
Average out degree:   1.6888
7506
```

In [48]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp7al,k=8).nodes())))
Gp7al_k8= list(nx.k_core(Gp7al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp7al)
Gp7_in_dcs_k8 = [in_dcs[n] for n in Gp7al_k8]
print("k8 in degree centrality: " + str(sum(Gp7_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp7al)
Gp7_out_dcs_k8 = [out_dcs[n] for n in Gp7al_k8]
print("k8 out degree centrality: " + str(sum(Gp7_out_dcs_k8)))
```

```
no. of nodes, k=8,: 3147
k8 in degree centrality: 1.0948672243715676
k8 out degree centrality: 0.3616338187931686
```

In [49]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp7al,k=5).nodes())))
Gp7al_k5= list(nx.k_core(Gp7al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp7al)
Gp7_in_dcs_k5 = [in_dcs[n] for n in Gp7al_k5]
print("k5 in degree centrality: " + str(sum(Gp7_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp7al)
Gp7_out_dcs_k5 = [out_dcs[n] for n in Gp7al_k5]
print("k5 out degree centrality: " + str(sum(Gp7_out_dcs_k5)))
```

```
no. of nodes, k=5,: 8140
k5 in degree centrality: 1.303793878749807
k5 out degree centrality: 0.6399258644088444
```

In [50]:
```
nx.density(Gp7al)
```

Out[50]:
```
1.710432004400625e-05
```

In [51]:
```
start = time.time()
Gp7_cc = nx.closeness_centrality(Gp7al)
Gp7_cc_k8 = [Gp7_cc[n] for n in Gp7al_k8]
Gp7_cc_k5 = [Gp7_cc[n] for n in Gp7al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p7 graph closeness centrality.")
print("closeness centrality avarage for the graph 7 is: " + str(sum(Gp7_cc.values())/len(Gp7_cc)))
print("closeness centrality avarage for the graph 7 (k=8) is: " + str(sum(Gp7_cc_k8)/len(Gp7_cc_k8)))
print("closeness centrality avarage for the graph 7 (k=5) is: " + str(sum(Gp7_cc_k5)/len(Gp7_cc_k5)))

```

```
It took 5.927479720115661 minutes to calculate p7 graph closeness centrality.
closeness centrality avarage for the graph 7 is: 0.0007004523004910116
closeness centrality avarage for the graph 7 (k=8) is: 0.007452709274069453
closeness centrality avarage for the graph 7 (k=5) is: 0.004772107456720052
```

In [52]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#-----------------------------period 8---------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [53]:
```
df_p8_nxall = df_p8[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p8_nxall = df_p8_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p8_nxall_nan = df_p8_nxall[df_p8_nxall['edge'].notna()]
Gp8al = nx.from_pandas_edgelist(df_p8_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp8al.add_nodes_from(df_p8_nxall["user_handle"])
Gp8al.remove_edges_from(nx.selfloop_edges(Gp8al))

print(nx.info(Gp8al))
print(len(list(nx.isolates(Gp8al))))
```

```
Name:
Type: DiGraph
Number of nodes: 97494
Number of edges: 164258
Average in degree:   1.6848
Average out degree:   1.6848
8418
```

In [54]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp8al,k=8).nodes())))
Gp8al_k8= list(nx.k_core(Gp8al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp8al)
Gp8_in_dcs_k8 = [in_dcs[n] for n in Gp8al_k8]
print("k8 in degree centrality: " + str(sum(Gp8_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp8al)
Gp8_out_dcs_k8 = [out_dcs[n] for n in Gp8al_k8]
print("k8 out degree centrality: " + str(sum(Gp8_out_dcs_k8)))
```

```
no. of nodes, k=8,: 3186
k8 in degree centrality: 1.06594319592175
k8 out degree centrality: 0.37500128214333295
```

In [55]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp8al,k=5).nodes())))
Gp8al_k5= list(nx.k_core(Gp8al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp8al)
Gp8_in_dcs_k5 = [in_dcs[n] for n in Gp8al_k5]
print("k5 in degree centrality: " + str(sum(Gp8_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp8al)
Gp8_out_dcs_k5 = [out_dcs[n] for n in Gp8al_k5]
print("k5 out degree centrality: " + str(sum(Gp8_out_dcs_k5)))
```

```
no. of nodes, k=5,: 8131
k5 in degree centrality: 1.2932723375011268
k5 out degree centrality: 0.6503030986839937
```

In [56]:
```
nx.density(Gp8al)
```

Out[56]:
```
1.7281252151088908e-05
```

In [57]:
```
start = time.time()
Gp8_cc = nx.closeness_centrality(Gp8al)
Gp8_cc_k8 = [Gp8_cc[n] for n in Gp8al_k8]
Gp8_cc_k5 = [Gp8_cc[n] for n in Gp8al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p8 graph closeness centrality.")
print("closeness centrality avarage for the graph 8 is: " + str(sum(Gp8_cc.values())/len(Gp8_cc)))
print("closeness centrality avarage for the graph 8 (k=8) is: " + str(sum(Gp8_cc_k8)/len(Gp8_cc_k8)))
print("closeness centrality avarage for the graph 8 (k=5) is: " + str(sum(Gp8_cc_k5)/len(Gp8_cc_k5)))


```

```
It took 5.451728165149689 minutes to calculate p8 graph closeness centrality.
closeness centrality avarage for the graph 8 is: 0.0006941140153086925
closeness centrality avarage for the graph 8 (k=8) is: 0.007029553696790752
closeness centrality avarage for the graph 8 (k=5) is: 0.004718905503252526
```

In [58]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#------------------------------period 9--------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [59]:
```
df_p9_nxall = df_p9[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p9_nxall = df_p9_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p9_nxall_nan = df_p9_nxall[df_p9_nxall['edge'].notna()]
Gp9al = nx.from_pandas_edgelist(df_p9_nxall_nan, source="user_handle", target="edge",
                                create_using=nx.DiGraph())
Gp9al.add_nodes_from(df_p9_nxall["user_handle"])
Gp9al.remove_edges_from(nx.selfloop_edges(Gp9al))

print(nx.info(Gp9al))
print(len(list(nx.isolates(Gp9al))))
```

```
Name:
Type: DiGraph
Number of nodes: 61336
Number of edges: 91540
Average in degree:   1.4924
Average out degree:   1.4924
3664
```

In [60]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp9al,k=8).nodes())))
Gp9al_k8= list(nx.k_core(Gp9al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp9al)
Gp9_in_dcs_k8 = [in_dcs[n] for n in Gp9al_k8]
print("k8 in degree centrality: " + str(sum(Gp9_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp9al)
Gp9_out_dcs_k8 = [out_dcs[n] for n in Gp9al_k8]
print("k8 out degree centrality: " + str(sum(Gp9_out_dcs_k8)))
```

```
no. of nodes, k=8,: 0
k8 in degree centrality: 0
k8 out degree centrality: 0
```

In [61]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp9al,k=5).nodes())))
Gp9al_k5= list(nx.k_core(Gp9al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp9al)
Gp9_in_dcs_k5 = [in_dcs[n] for n in Gp9al_k5]
print("k5 in degree centrality: " + str(sum(Gp9_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp9al)
Gp9_out_dcs_k5 = [out_dcs[n] for n in Gp9al_k5]
print("k5 out degree centrality: " + str(sum(Gp9_out_dcs_k5)))
```

```
no. of nodes, k=5,: 3291
k5 in degree centrality: 0.9951577402788041
k5 out degree centrality: 0.3663976522377103
```

In [62]:
```
nx.density(Gp9al)
```

Out[62]:
```
2.433251995625484e-05
```

In [63]:
```
start = time.time()
Gp9_cc = nx.closeness_centrality(Gp9al)
#closeness centrality mean for k-8 cant be calculated as k-8=0
Gp9_cc_k5 = [Gp9_cc[n] for n in Gp9al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p9 graph closeness centrality.")
print("closeness centrality avarage for the graph 9 is: " + str(sum(Gp9_cc.values())/len(Gp9_cc)))
print("closeness centrality avarage for the graph 9 (k=5) is: " + str(sum(Gp9_cc_k5)/len(Gp9_cc_k5)))

```

```
It took 1.5154205481211345 minutes to calculate p9 graph closeness centrality.
closeness centrality avarage for the graph 9 is: 0.00040238920062419473
closeness centrality avarage for the graph 9 (k=5) is: 0.003814708426131662
```

In [64]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#------------------------------period 10--------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [65]:
```
df_p10_nxall = df_p10[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p10_nxall = df_p10_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p10_nxall_nan = df_p10_nxall[df_p10_nxall['edge'].notna()]
Gp10al = nx.from_pandas_edgelist(df_p10_nxall_nan, source="user_handle", target="edge",
                                 create_using=nx.DiGraph())
Gp10al.add_nodes_from(df_p10_nxall["user_handle"])
Gp10al.remove_edges_from(nx.selfloop_edges(Gp10al))

print(nx.info(Gp10al))
print(len(list(nx.isolates(Gp10al))))
```

```
Name:
Type: DiGraph
Number of nodes: 76205
Number of edges: 92472
Average in degree:   1.2135
Average out degree:   1.2135
5224
```

In [66]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp10al,k=8).nodes())))
Gp10al_k8= list(nx.k_core(Gp10al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp10al)
Gp10_in_dcs_k8 = [in_dcs[n] for n in Gp10al_k8]
print("k8 in degree centrality: " + str(sum(Gp10_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp10al)
Gp10_out_dcs_k8 = [out_dcs[n] for n in Gp10al_k8]
print("k8 out degree centrality: " + str(sum(Gp10_out_dcs_k8)))
```

```
no. of nodes, k=8,: 0
k8 in degree centrality: 0
k8 out degree centrality: 0
```

In [67]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp10al,k=5).nodes())))
Gp10al_k5= list(nx.k_core(Gp10al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp10al)
Gp10_in_dcs_k5 = [in_dcs[n] for n in Gp10al_k5]
print("k5 in degree centrality: " + str(sum(Gp10_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp10al)
Gp10_out_dcs_k5 = [out_dcs[n] for n in Gp10al_k5]
print("k5 out degree centrality: " + str(sum(Gp10_out_dcs_k5)))
```

```
no. of nodes, k=5,: 1910
k5 in degree centrality: 0.6121594666946627
k5 out degree centrality: 0.1626292583066516
```

In [68]:
```
nx.density(Gp10al)
```

Out[68]:
```
1.5923884356271793e-05
```

In [69]:
```
start = time.time()
Gp10_cc = nx.closeness_centrality(Gp10al)
#closeness centrality mean for k-8 cant be calculated as k-8=0
Gp10_cc_k5 = [Gp10_cc[n] for n in Gp10al_k5]
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p10 graph closeness centrality.")
print("closeness centrality avarage for the graph 10 is: " + str(sum(Gp10_cc.values())/len(Gp10_cc)))
print("closeness centrality avarage for the graph 10 (k=5) is: " + str(sum(Gp10_cc_k5)/len(Gp10_cc_k5)))

```

```
It took 0.25149667263031006 minutes to calculate p10 graph closeness centrality.
closeness centrality avarage for the graph 10 is: 5.173298517616693e-05
closeness centrality avarage for the graph 10 (k=5) is: 0.0008995130179805902
```

In [70]:
```
sum(Gp10_cc.values())
```

Out[70]:
```
3.942312135349801
```

In [71]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#-----------------------------period 11-----------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [72]:
```
df_p11_nxall = df_p11[["user_handle", "tweet_type", "tweet_id", "tweet_time", "reply_user", "qu_user", "retweeted_user"]]
df_p11_nxall = df_p11_nxall.melt(["user_handle", "tweet_type", "tweet_id", "tweet_time"], var_name="type", value_name="edge")
df_p11_nxall_nan = df_p11_nxall[df_p11_nxall['edge'].notna()]
Gp11al = nx.from_pandas_edgelist(df_p11_nxall_nan, source="user_handle", target="edge",
                                 create_using=nx.DiGraph())
Gp11al.add_nodes_from(df_p11_nxall["user_handle"])
Gp11al.remove_edges_from(nx.selfloop_edges(Gp11al))

print(nx.info(Gp11al))
print(len(list(nx.isolates(Gp11al))))
```

```
Name:
Type: DiGraph
Number of nodes: 56053
Number of edges: 71359
Average in degree:   1.2731
Average out degree:   1.2731
4477
```

In [73]:
```
print("no. of nodes, k=8,: "+ str(len(nx.k_core(Gp11al,k=8).nodes())))
Gp11al_k8= list(nx.k_core(Gp11al,k=8).nodes())
in_dcs = nx.in_degree_centrality(Gp11al)
Gp11_in_dcs_k8 = [in_dcs[n] for n in Gp11al_k8]
print("k8 in degree centrality: " + str(sum(Gp11_in_dcs_k8)))
out_dcs = nx.out_degree_centrality(Gp11al)
Gp11_out_dcs_k8 = [out_dcs[n] for n in Gp11al_k8]
print("k8 out degree centrality: " + str(sum(Gp11_out_dcs_k8)))
```

```
no. of nodes, k=8,: 0
k8 in degree centrality: 0
k8 out degree centrality: 0
```

In [74]:
```
print("no. of nodes, k=5,: "+ str(len(nx.k_core(Gp11al,k=5).nodes())))
Gp11al_k5= list(nx.k_core(Gp11al,k=5).nodes())
in_dcs = nx.in_degree_centrality(Gp11al)
Gp11_in_dcs_k5 = [in_dcs[n] for n in Gp11al_k5]
print("k5 in degree centrality: " + str(sum(Gp11_in_dcs_k5)))
out_dcs = nx.out_degree_centrality(Gp11al)
Gp11_out_dcs_k5 = [out_dcs[n] for n in Gp11al_k5]
print("k5 out degree centrality: " + str(sum(Gp11_out_dcs_k5)))
```

```
no. of nodes, k=5,: 1956
k5 in degree centrality: 0.6936416184971097
k5 out degree centrality: 0.23954542210804397
```

In [75]:
```
nx.density(Gp11al)
```

Out[75]:
```
2.2712177869695147e-05
```

In [76]:
```
start = time.time()
Gp11_cc = nx.closeness_centrality(Gp11al)
Gp11_cc_k5 = [Gp11_cc[n] for n in Gp11al_k5]
#closeness centrality mean for k-8 cant be calculated as k-8=0
end = time.time()
print("It took " + str((end-start)/60) + " minutes to calculate p11 graph closeness centrality.")
print("closeness centrality avarage for the graph 11 is: " + str(sum(Gp11_cc.values())/len(Gp11_cc)))
print("closeness centrality avarage for the graph 11 (k=5) is: " + str(sum(Gp11_cc_k5)/len(Gp11_cc_k5)))

```

```
It took 0.32306398153305055 minutes to calculate p11 graph closeness centrality.
closeness centrality avarage for the graph 11 is: 0.0001798023928626082
closeness centrality avarage for the graph 11 (k=5) is: 0.002412870827455767
```

In [ ]:
```

```

In [77]:
```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#------------------------------nodes and edges--------------------------
#-----------------------------------------------------------------
#-----------------------------------------------------------------
```

In [78]:
```
print(nx.info(Gp1al))
print(type(nx.info(Gp1al)))
(len(list(nx.isolates(Gp1al))))
```

```
Name:
Type: DiGraph
Number of nodes: 111922
Number of edges: 169164
Average in degree:   1.5114
Average out degree:   1.5114
<class 'str'>
```

Out[78]:
```
12754
```

In [79]:
```
gp1_info = {"period": "p1", "nodes": 111922, "edges":169164, "isolated nodes": 12754}
```

In [80]:
```
print(gp1_info)
```

```
{'period': 'p1', 'nodes': 111922, 'edges': 169164, 'isolated nodes': 12754}
```

In [81]:
```
print(nx.info(Gp2al))
print(type(nx.info(Gp2al)))
(len(list(nx.isolates(Gp2al))))
```

```
Name:
Type: DiGraph
Number of nodes: 77698
Number of edges: 127208
Average in degree:   1.6372
Average out degree:   1.6372
<class 'str'>
```

Out[81]:
```
7025
```

In [82]:
```
gp2_info = {"period": "p2", "nodes": 77698, "edges":127208, "isolated nodes": 7025}
```

In [83]:
```
print(nx.info(Gp3al))
print(type(nx.info(Gp3al)))
(len(list(nx.isolates(Gp3al))))
```

```
Name:
Type: DiGraph
Number of nodes: 77436
Number of edges: 127547
Average in degree:   1.6471
Average out degree:   1.6471
<class 'str'>
```

Out[83]:
```
5684
```

In [84]:
```
gp3_info = {"period": "p3", "nodes": 77436, "edges":127547, "isolated nodes": 5684}
```

In [85]:
```
print(nx.info(Gp4al))
print(type(nx.info(Gp4al)))
(len(list(nx.isolates(Gp4al))))
```

```
Name:
Type: DiGraph
Number of nodes: 52304
Number of edges: 78970
Average in degree:   1.5098
Average out degree:   1.5098
<class 'str'>
```

Out[85]:
```
3344
```

In [86]:
```
gp4_info = {"period": "p4", "nodes": 52304, "edges":78970, "isolated nodes": 3344}
```

In [87]:
```
print(nx.info(Gp5al))
print(type(nx.info(Gp5al)))
(len(list(nx.isolates(Gp5al))))
```

```
Name:
Type: DiGraph
Number of nodes: 55953
Number of edges: 84031
Average in degree:   1.5018
Average out degree:   1.5018
<class 'str'>
```

Out[87]: 3872

In [88]:
```
gp5_info = {"period": "p5", "nodes": 55953, "edges":84031, "isolated nodes": 3872}
```

In [89]:
```
print(nx.info(Gp6al))
print(type(nx.info(Gp6al)))
(len(list(nx.isolates(Gp6al))))
```

```
Name:
Type: DiGraph
Number of nodes: 71976
Number of edges: 111785
Average in degree:   1.5531
Average out degree:   1.5531
<class 'str'>
```

Out[89]: 5399

In [90]:
```
gp6_info = {"period": "p6", "nodes": 71976, "edges":111785, "isolated nodes": 5399}
```

In [91]:
```
print(nx.info(Gp7al))
print(type(nx.info(Gp7al)))
(len(list(nx.isolates(Gp7al))))
```

```
Name:
Type: DiGraph
Number of nodes: 98739
Number of edges: 166755
Average in degree:   1.6888
Average out degree:   1.6888
<class 'str'>
```

Out[91]: 7506

In [92]:
```
gp7_info = {"period": "p7", "nodes": 98739, "edges":166755, "isolated nodes": 7506}
```

In [93]:
```
print(nx.info(Gp8al))
print(type(nx.info(Gp8al)))
(len(list(nx.isolates(Gp8al))))
```

```
Name:
Type: DiGraph
Number of nodes: 97494
Number of edges: 164258
Average in degree:   1.6848
Average out degree:   1.6848
<class 'str'>
```

Out[93]: 8418

In [94]:
```
gp8_info = {"period": "p8", "nodes": 97494, "edges":164258, "isolated nodes": 8418}
```

In [95]:
```
print(nx.info(Gp9al))
print(type(nx.info(Gp9al)))
(len(list(nx.isolates(Gp9al))))
```

```
Name:
Type: DiGraph
Number of nodes: 61336
Number of edges: 91540
Average in degree:   1.4924
Average out degree:   1.4924
<class 'str'>
```

Out[95]: 3664

In [96]:
```
gp9_info = {"period": "p9", "nodes": 61336, "edges":91540, "isolated nodes": 3664}
```

In [97]:
```
print(nx.info(Gp10al))
print(type(nx.info(Gp10al)))
(len(list(nx.isolates(Gp10al))))
```

```
Name:
Type: DiGraph
Number of nodes: 76205
Number of edges: 92472
Average in degree:   1.2135
Average out degree:   1.2135
<class 'str'>
```

Out[97]: 5224

In [98]:
```
gp10_info = {"period": "p10", "nodes": 76205, "edges":92472, "isolated nodes": 5224}
```

In [99]:
```
print(nx.info(Gp11al))
print(type(nx.info(Gp11al)))
(len(list(nx.isolates(Gp11al))))
```

```
Name:
Type: DiGraph
Number of nodes: 56053
Number of edges: 71359
Average in degree:   1.2731
Average out degree:   1.2731
<class 'str'>
```

Out[99]: 4477

In [100]:
```
gp11_info = {"period": "p11", "nodes": 56053, "edges":71359, "isolated nodes": 4477}
```

In [101]:
```
df_info1 = pd.DataFrame(gp1_info, index=[0])
df_info2 = pd.DataFrame(gp2_info, index=[0])
df_info = df_info1.append(df_info2, ignore_index = True)
df_info3 = pd.DataFrame(gp3_info, index=[0])
df_info = df_info.append(df_info3, ignore_index = True)
df_info4 = pd.DataFrame(gp4_info, index=[0])
df_info = df_info.append(df_info4, ignore_index = True)
df_info5 = pd.DataFrame(gp6_info, index=[0])
df_info = df_info.append(df_info5, ignore_index = True)
df_info6 = pd.DataFrame(gp6_info, index=[0])
df_info = df_info.append(df_info6, ignore_index = True)
df_info7 = pd.DataFrame(gp7_info, index=[0])
df_info = df_info.append(df_info7, ignore_index = True)
df_info8 = pd.DataFrame(gp8_info, index=[0])
df_info = df_info.append(df_info8, ignore_index = True)
df_info9 = pd.DataFrame(gp9_info, index=[0])
df_info = df_info.append(df_info9, ignore_index = True)
df_info10 = pd.DataFrame(gp10_info, index=[0])
df_info = df_info.append(df_info10, ignore_index = True)
df_info11 = pd.DataFrame(gp11_info, index=[0])
df_info = df_info.append(df_info11, ignore_index = True)
```

In [102]:
```
df_info
```

Out[102]:

	period	nodes	edges	isolated nodes
0	p1	111922	169164	12754
1	p2	77698	127208	7025
2	p3	77436	127547	5684
3	p4	52304	78970	3344
4	p6	71976	111785	5399
5	p6	71976	111785	5399
6	p7	98739	166755	7506
7	p8	97494	164258	8418
8	p9	61336	91540	3664
9	p10	76205	92472	5224
10	p11	56053	71359	4477

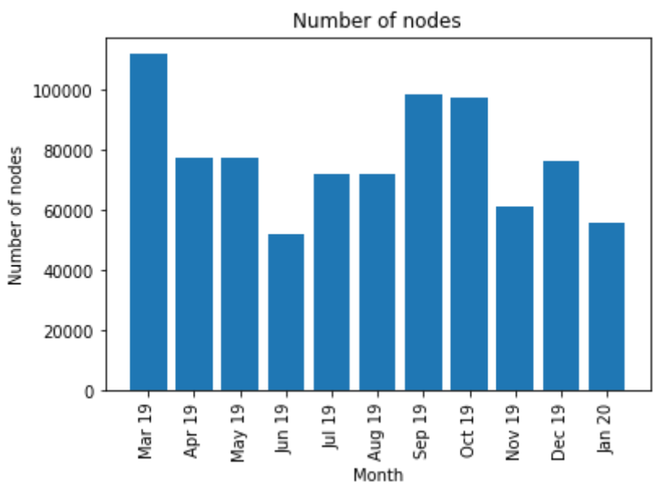
In [103]:
```
df_info["nodes pr edges"] = df_info["edges"] / df_info["nodes"]
df_info["no isolated nodes pr edges"] = df_info["edges"] / (df_info["nodes"] - df_info["isolated nodes"])
df_info
```

Out[103]:

	period	nodes	edges	isolated nodes	nodes pr edges	no isolated nodes pr edges
0	p1	111922	169164	12754	1.511445	1.705833
1	p2	77698	127208	7025	1.637211	1.799952
2	p3	77436	127547	5684	1.647128	1.777609
3	p4	52304	78970	3344	1.509827	1.612949
4	p6	71976	111785	5399	1.553087	1.679033
5	p6	71976	111785	5399	1.553087	1.679033
6	p7	98739	166755	7506	1.688846	1.827793
7	p8	97494	164258	8418	1.684801	1.844021
8	p9	61336	91540	3664	1.492435	1.587252
9	p10	76205	92472	5224	1.213464	1.302771
10	p11	56053	71359	4477	1.273063	1.383570

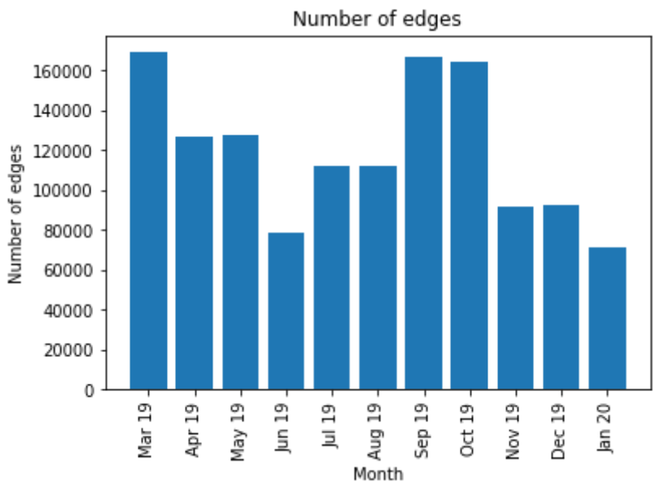
In [104]:
```
period = ["Mar 19", "Apr 19", "May 19", "Jun 19", "Jul 19", "Aug 19", "Sep 19", "Oct 19", "Nov 19", "Dec 19", "Jan 20"]
plt.bar(period, df_info["nodes"], label="nodes")
plt.xticks(rotation=90)
plt.xlabel('Month')
plt.ylabel('Number of nodes')
plt.title("Number of nodes")

```

Out[104]: Text(0.5, 1.0, 'Number of nodes')



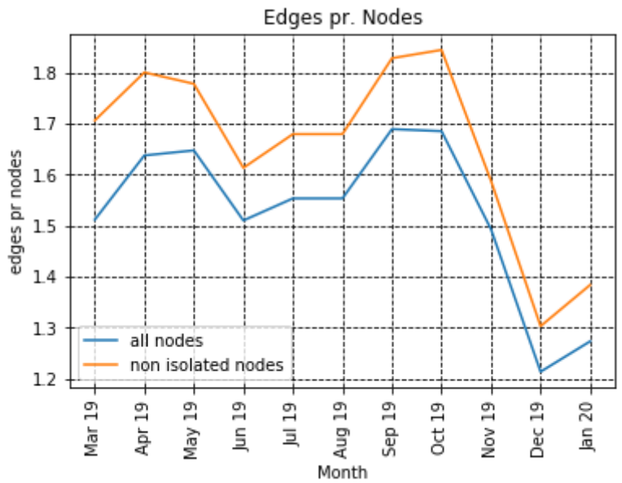
In [105]:
```
plt.bar(period, df_info["edges"], label="edges")
plt.xticks(rotation=90)
plt.xlabel('Month')
plt.ylabel('Number of edges')
plt.title("Number of edges")
```

Out[105]: Text(0.5, 1.0, 'Number of edges')



In [106]:
```
plt.plot(period, df_info["nodes pr edges"], label="all nodes")
plt.plot(period, df_info["no isolated nodes pr edges"], label="non isolated nodes")
plt.grid(color="k", linestyle="--")
plt.xticks(rotation=90)
plt.xlabel('Month')
plt.ylabel('edges pr nodes')
plt.title("Edges pr. Nodes")
plt.legend()

```

Out[106]: <matplotlib.legend.Legend at 0xa9177c810>



In [107]:
```
Gp1_k5nodes = len(nx.k_core(Gp1al,k=5).nodes())
Gp1_nodes = len(Gp1al.nodes())
Gp2_k5nodes = len(nx.k_core(Gp2al,k=5).nodes())
Gp2_nodes = len(Gp2al.nodes())
Gp3_k5nodes = len(nx.k_core(Gp3al,k=5).nodes())
Gp3_nodes = len(Gp3al.nodes())
Gp4_k5nodes = len(nx.k_core(Gp4al,k=5).nodes())
Gp4_nodes = len(Gp4al.nodes())
Gp5_k5nodes = len(nx.k_core(Gp5al,k=5).nodes())
Gp5_nodes = len(Gp5al.nodes())
Gp6_k5nodes = len(nx.k_core(Gp6al,k=5).nodes())
Gp6_nodes = len(Gp6al.nodes())
Gp7_k5nodes = len(nx.k_core(Gp7al,k=5).nodes())
Gp7_nodes = len(Gp7al.nodes())
Gp8_k5nodes = len(nx.k_core(Gp8al,k=5).nodes())
Gp8_nodes = len(Gp8al.nodes())
Gp9_k5nodes = len(nx.k_core(Gp9al,k=5).nodes())
Gp9_nodes = len(Gp9al.nodes())
Gp10_k5nodes = len(nx.k_core(Gp10al,k=5).nodes())
Gp10_nodes = len(Gp10al.nodes())
Gp11_k5nodes = len(nx.k_core(Gp11al,k=5).nodes())
Gp11_nodes = len(Gp11al.nodes())
```

In [108]:
```
type(Gp1_k5nodes)
```

Out[108]: int

In [109]:
```
Gp_k5nodes = [Gp1_k5nodes, Gp2_k5nodes, Gp3_k5nodes, Gp4_k5nodes, Gp5_k5nodes, Gp6_k5nodes, Gp7_k5nodes, Gp8_k5nodes, Gp9_k5nodes, Gp10_k5nodes, Gp11_k5nodes]

```

In [110]:
```
Gp_k5nodes
```

Out[110]:
```
[7344, 5883, 5719, 3112, 3369, 4829, 8140, 8131, 3291, 1910, 1956]
```

In [111]:
```
Gp_nodes = [Gp1_nodes, Gp2_nodes, Gp3_nodes, Gp4_nodes, Gp5_nodes, Gp6_nodes, Gp7_nodes, Gp8_nodes, Gp9_nodes, Gp10_nodes, Gp11_nodes]
Gp_nodes
```

Out[111]:
```
[111922, 77698, 77436, 52304, 55953, 71976, 98739, 97494, 61336, 76205, 56053]
```

In [112]:
```
df_info_k5 = pd.DataFrame(columns=["period", "nodes", "k-5 nodes"])
```

In [113]:
```
df_info_k5
```

Out[113]:

period	nodes	k-5 nodes

In [114]:
```
Gp_k5nodes = pd.Series(Gp_k5nodes)
```

In [115]:
```
df_info_k5["nodes"] = Gp_nodes
df_info_k5["k-5 nodes"] = Gp_k5nodes
df_info_k5["k-5 nodes/nodes"] = (df_info_k5["k-5 nodes"]/df_info_k5["nodes"])*100
df_info_k5["period"] = period

```

In [116]:
```
df_info_k5
```

Out[116]:

	period	nodes	k-5 nodes	k-5 nodes/nodes
0	Mar 19	111922	7344	6.561713
1	Apr 19	77698	5883	7.571623
2	May 19	77436	5719	7.385454
3	Jun 19	52304	3112	5.949832
4	Jul 19	55953	3369	6.021125
5	Aug 19	71976	4829	6.709181
6	Sep 19	98739	8140	8.243956
7	Oct 19	97494	8131	8.340000
8	Nov 19	61336	3291	5.365528
9	Dec 19	76205	1910	2.506397
10	Jan 20	56053	1956	3.489555

In [117]:
```
Gp1_k5_odsum = sum(Gp1_out_dcs_k5)
Gp2_k5_odsum = sum(Gp2_out_dcs_k5)
Gp3_k5_odsum = sum(Gp3_out_dcs_k5)
Gp4_k5_odsum = sum(Gp4_out_dcs_k5)
Gp5_k5_odsum = sum(Gp5_out_dcs_k5)
Gp6_k5_odsum = sum(Gp6_out_dcs_k5)
Gp7_k5_odsum = sum(Gp7_out_dcs_k5)
Gp8_k5_odsum = sum(Gp8_out_dcs_k5)
Gp9_k5_odsum = sum(Gp9_out_dcs_k5)
Gp10_k5_odsum = sum(Gp10_out_dcs_k5)
Gp11_k5_odsum = sum(Gp11_out_dcs_k5)

Gp1_k5_idsum = sum(Gp1_in_dcs_k5)
Gp2_k5_idsum = sum(Gp2_in_dcs_k5)
Gp3_k5_idsum = sum(Gp3_in_dcs_k5)
Gp4_k5_idsum = sum(Gp4_in_dcs_k5)
Gp5_k5_idsum = sum(Gp5_in_dcs_k5)
Gp6_k5_idsum = sum(Gp6_in_dcs_k5)
Gp7_k5_idsum = sum(Gp7_in_dcs_k5)
Gp8_k5_idsum = sum(Gp8_in_dcs_k5)
Gp9_k5_idsum = sum(Gp9_in_dcs_k5)
Gp10_k5_idsum = sum(Gp10_in_dcs_k5)
Gp11_k5_idsum = sum(Gp11_in_dcs_k5)
```

In [118]:
```
Gp_k5_id = [Gp1_k5_idsum, Gp2_k5_idsum, Gp3_k5_idsum, Gp4_k5_idsum, Gp5_k5_idsum, Gp6_k5_idsum, Gp7_k5_idsum, Gp8_k5_idsum, Gp9_k5_idsum, Gp10_k5_idsum, Gp11_k5_id
Gp_k5_od = [Gp1_k5_odsum, Gp2_k5_odsum, Gp3_k5_odsum, Gp4_k5_odsum, Gp5_k5_odsum, Gp6_k5_odsum, Gp7_k5_odsum, Gp8_k5_odsum, Gp9_k5_odsum, Gp10_k5_odsum, Gp11_k5_od
```

In [119]:
```
df_info_k5["in_degree mean (nor)"] = Gp_k5_id/df_info_k5["k-5 nodes"]
df_info_k5["out_degree mean (nor)"] = Gp_k5_od/df_info_k5["k-5 nodes"]
df_info_k5
```

Out[119]:

	period	nodes	k-5 nodes	k-5 nodes/nodes	in_degree mean (nor)	out_degree mean (nor)
0	Mar 19	111922	7344	6.561713	0.000148	0.000073
1	Apr 19	77698	5883	7.571623	0.000208	0.000104
2	May 19	77436	5719	7.385454	0.000213	0.000106
3	Jun 19	52304	3112	5.949832	0.000335	0.000133
4	Jul 19	55953	3369	6.021125	0.000310	0.000126
5	Aug 19	71976	4829	6.709181	0.000230	0.000103
6	Sep 19	98739	8140	8.243956	0.000160	0.000079
7	Oct 19	97494	8131	8.340000	0.000159	0.000080
8	Nov 19	61336	3291	5.365528	0.000302	0.000111
9	Dec 19	76205	1910	2.506397	0.000321	0.000085
10	Jan 20	56053	1956	3.489555	0.000355	0.000122

In [120]:
```
df_info_k5["in_degree mean"] = df_info_k5["in_degree mean (nor)"] * (df_info_k5["nodes"] - 1)
df_info_k5["out_degree mean"] = df_info_k5["out_degree mean (nor)"] * (df_info_k5["nodes"] - 1)
```

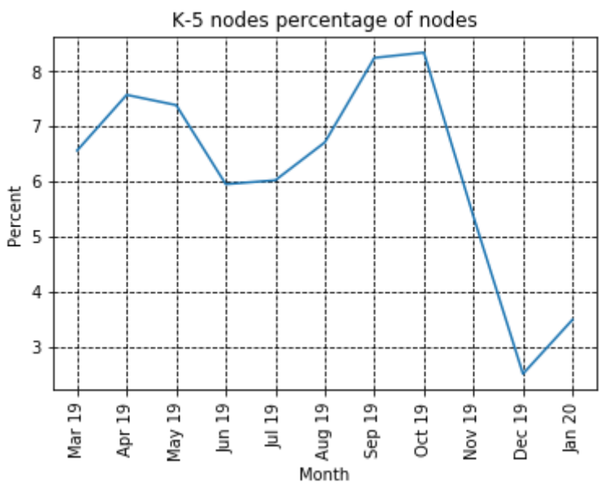
In [121]:
```
df_info_k5
```

Out[121]:

	period	nodes	k-5 nodes	k-5 nodes/nodes	in_degree mean (nor)	out_degree mean (nor)	in_degree mean	out_degree mean
0	Mar 19	111922	7344	6.561713	0.000148	0.000073	16.582108	8.208333
1	Apr 19	77698	5883	7.571623	0.000208	0.000104	16.140575	8.050824
2	May 19	77436	5719	7.385454	0.000213	0.000106	16.509530	8.199860
3	Jun 19	52304	3112	5.949832	0.000335	0.000133	17.500000	6.951478
4	Jul 19	55953	3369	6.021125	0.000310	0.000126	17.324725	7.068863
5	Aug 19	71976	4829	6.709181	0.000230	0.000103	16.530752	7.384552
6	Sep 19	98739	8140	8.243956	0.000160	0.000079	15.814988	7.762285
7	Oct 19	97494	8131	8.340000	0.000159	0.000080	15.506703	7.797319
8	Nov 19	61336	3291	5.365528	0.000302	0.000111	18.546946	6.828624
9	Dec 19	76205	1910	2.506397	0.000321	0.000085	24.423560	6.488482
10	Jan 20	56053	1956	3.489555	0.000355	0.000122	19.877301	6.864519

In [122]:
```
plt.plot(period, df_info_k5["k-5 nodes/nodes"])
plt.grid(color="k", linestyle="--")
plt.xticks(rotation=90)
plt.xlabel('Month')
plt.ylabel('Percent')
plt.title("K-5 nodes percentage of nodes")

```

Out[122]: Text(0.5, 1.0, 'K-5 nodes percentage of nodes')



In [123]:
```
plt.plot(period, df_info_k5["in_degree mean"], label = "in_degree mean")
plt.plot(period, df_info_k5["out_degree mean"], label = "out_degree mean")
plt.grid(color="k", linestyle="--")
plt.xticks(rotation=90)
plt.xlabel('Month')
plt.ylabel('Degrees (not normalized)')
plt.title("K-5 core not normalized degrees")
plt.legend()
```

Out[123]: <matplotlib.legend.Legend at 0xa9e27c4d0>