VOLATILE WORLD : BRIEF ANALYSIS OF ETHNIC CONFLICTS IN 2001

INTRODUCTION

Social dynamics is the study of collective human behaviour. How humans behave and interact in a group and how those individual interactions form the group behaviour. In simple words social dynamics is study of mob behaviour and behaviour and interactions of its constituents.

Humans have collectively achieved great milestones in history and continue to do so. Corporates, institutions, governments, organizations all tend to work and make decisions by involving multiple peoples, who are good in their field. Most sports involve multiple players, and that's what make it more interesting, we see how admirably players interact and coordinate to achieve their goal. Then there is scientific community, whose constituents contribute, create and share new ideas and do research to prove their hypothesis and these researches then make foundation for new research and ideas, it's kind of group interaction of curious and intelligent peoples.

These are some of the examples of tasks that humans do as a group and are productive. But this is not the case all the time. What if people use this collective or group behaviour to do something destructive, something negative. This is what ethnic conflict is about. Mob behaviour and decisions, in general are not very logical. And this become specifically true if mob is big and constituted by volatile minded peoples. Its widely accepted idea that in volatile mobs the collective intelligence is significantly lower than the individual intelligence of constituents. And that's what make this collective human behaviour so dangerous.

This report is about analysis of network of ethnic conflicts among different communities. And how these ethnic conflicts can be viewed to get better understanding.

What is an ETHNIC CONFLICT ?

ETHNICITY

According to Norwegian anthropologist Thomas Hylland Eriksen:

The term ethnicity refers to the enduring relationship between more or less bounded groups or social categories that perceive themselves as being culturally different from each other. Ethnic identities are, furthermore, based on widely held notions of shared origin and shared culture, and must be recognized as such by outsiders as well as by the proclaimed members of an ethnic group or category. Ethnicity, it could be said, is a means of ren-dering cultural differences comparable. There has evolved a global discourse, or "grammar," about ethnicity which entails a great num-ber of formal commonalities between ethnic groups struggling for recognition everywhere. The emphasis on cultural heritage, shared customs, and a history of oppression is shared by ethnic minorities everywhere. To a great extent, they have learned from each other, even if their battles are in a substantial way local and unique.[1]

HOMOPHILY

Homophily literally means 'love of similar kind'. Homophily is the tendency of people to bond and relate more closely with peoples that are similar to them. Homophily theory predicts that people are more likely to interact with individuals similar to themselves in respect to a variety of qualities and characteristics.

There are two leading theories about homophily[2]:

1. The similarity-attraction hypothesis

Similarity-attraction hypothesis postulates that people are more likely to interact with those who have similar traits.

2. Theory of self-categorization

   Self-categorization theory argues that people tend to self-categorize with regard to race, gender, socio-economic status, etc., and they differentiate between similar and dissimilar others based on these attributes.

HETEROPHILY

Heterophily literally means 'love of different kind'. Heterophily is the tendency of people to bond and relate more closely with peoples that are different to them.

So according to the theory of self-categorization, humans in the society tends to follow homophily. In simple words, humans like people who are similar to them with respect to ethnical attributes – ancestry, religion, culture, nation, language, traditions.

Hence, we can argue that the peoples can relate to each other more closely if they belong to same ethnicity then as compared to the peoples belonging to different ethnicity. This can be the one of the prime factor contributing to ethnic conflicts.

DATASET

This study is based on the global ethnic conflict data of year 2001. The dataset contains 10 columns and 1176 rows and have missing values.

Brief explanation about columns

```
 Actor1Code             : The short form of Actor1Name.

Actor1Name             : One of the two actors involved in the Ethnic conflict.
Can be termed as a source with certain assumptions.

Actor1Geo_ADM1Code     : The geographical location of the Actor 1.

Actor2Code             : The short form of Actor2Name.

Actor2Name             : Second actor of the two actors involved in the Ethnic
conflict. Can be termed as a target with certain assumptions.

Actor2Geo_ADM1Code     : The geographical location of the Actor 2.

ActionGeo_FullName     : The geographical location of the ethnic conflict.

ActionGeo_ADM1Code     : The short form of ActionGeo_FullName.

Year                   : The year when the ethnic conflict happens. This is 2001
in this dataset.

SQLDATE                : The accurate date of the ethnic conflict.
```

In [83]:
```
# Importing relevant libraries
import networkx as nx
import numpy as np
import pandas as pd
import matplotlib as plt
from scipy.optimize import curve_fit
import wordcloud as wc

import seaborn as sns
sns.set_style('whitegrid')

import matplotlib.pyplot as plt
%matplotlib inline
```

In [84]:
```
# Loading dataset to dataframe
ec = pd.read_csv('E:\Documents\excel data\EC.csv')
```

Preliminary Analysis

In [85]:
```
# size of our dataset
print('There are {} rows and {} columns in the dataset'.format(ec.shape[0],ec.shape[1]))
```

```
There are 1176 rows and 10 columns in the dataset
```

In [86]:
```
# Top 5 values from dataframe to get information about structure of dataset
ec.head()
```

Out[86]:

	Actor1Code	Actor1Name	Actor1Geo_ADM1Code	Actor2Code	Actor2Name	Actor2Geo_ADM1Code	Action
0	ALB	ALBANIAN	AL	MKD	MACEDONIAN	AL	
1	RWA	RWANDA	RW	NaN	NaN	NaN	
2	MIL	PARAMILITARY	US	NaN	NaN	NaN	
3	CVL	VILLAGE	AL46	NaN	NaN	NaN	Varvar
4	CVL	POPULATION	AL	ALB	ALBANIAN	AL	B

In [87]:
```
# General information about columns in dataframe
print('There are total {} columns and name of columns names are \n{}'.format(len(list(ec.columr
```

```
There are total 10 columns and name of columns names are
['Actor1Code', 'Actor1Name', 'Actor1Geo_ADM1Code', 'Actor2Code', 'Actor2Name', 'Actor2Geo_ADM1C
ode', 'ActionGeo_FullName', 'ActionGeo_ADM1Code', 'Year', 'SQLDATE']
```

In [88]:
```
# Basic information about dataframe: number of non null values, data type of columns
ec.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1176 entries, 0 to 1175
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Actor1Code          1145 non-null   object
 1   Actor1Name          1145 non-null   object
 2   Actor1Geo_ADM1Code  1143 non-null   object
```

```
 3   Actor2Code          690 non-null    object
 4   Actor2Name          690 non-null    object
 5   Actor2Geo_ADM1Code  690 non-null    object
 6   ActionGeo_FullName  1093 non-null   object
 7   ActionGeo_ADM1Code  1174 non-null   object
 8   Year                1176 non-null   int64
 9   SQLDATE             1176 non-null   int64
dtypes: int64(2), object(8)
memory usage: 92.0+ KB
```

In [89]:
```
# Percentage of null values columnwise
nulls_percent = (ec.isnull().sum().sort_values()*100 / len(ec)).to_frame().style.background_gra
nulls_percent
```

Out[89]:

	0
Year	0.000000
SQLDATE	0.000000
ActionGeo_ADM1Code	0.170068
Actor1Code	2.636054
Actor1Name	2.636054
Actor1Geo_ADM1Code	2.806122
ActionGeo_FullName	7.057823
Actor2Code	41.326531
Actor2Name	41.326531
Actor2Geo_ADM1Code	41.326531

CONCLUSION

- There are lots of null values in columns Actor2Code, Actor2Name, Actor2Geo_ADM1Code. These all columns relate to Actor2, so most of the null values in dataset comes from missing information about Actor2 (target in case of directed graph).

Plotting relevant columns depicting dynamics

Geolocation of Ethnic conflicts - ActionGeo_FullName

In [90]:
```
# total number of ethnic conflicts happening at a particular location
number_loc = ec.ActionGeo_FullName.value_counts()
number_loc = number_loc.to_frame().reset_index().rename(columns={'index':'location','ActionGeo_
number_loc
```

Out[90]:

	location	frequency
0	Albania	133
1	Belgrade, Serbia (general),	72
2	Skopje, Macedonia (general), Macedonia	47
3	Croatia	37
4	Israel	36
...	...	...
256	Bitola, Bitola, Macedonia	1
257	Rome, Lazio, Italy	1

	location	frequency
258	Celebici, Federation of Bosnia and Herzegovina...	1
259	Neprosteno, Macedonia (general), Macedonia	1
260	Paris, RhôAlpes, France	1

261 rows × 2 columns

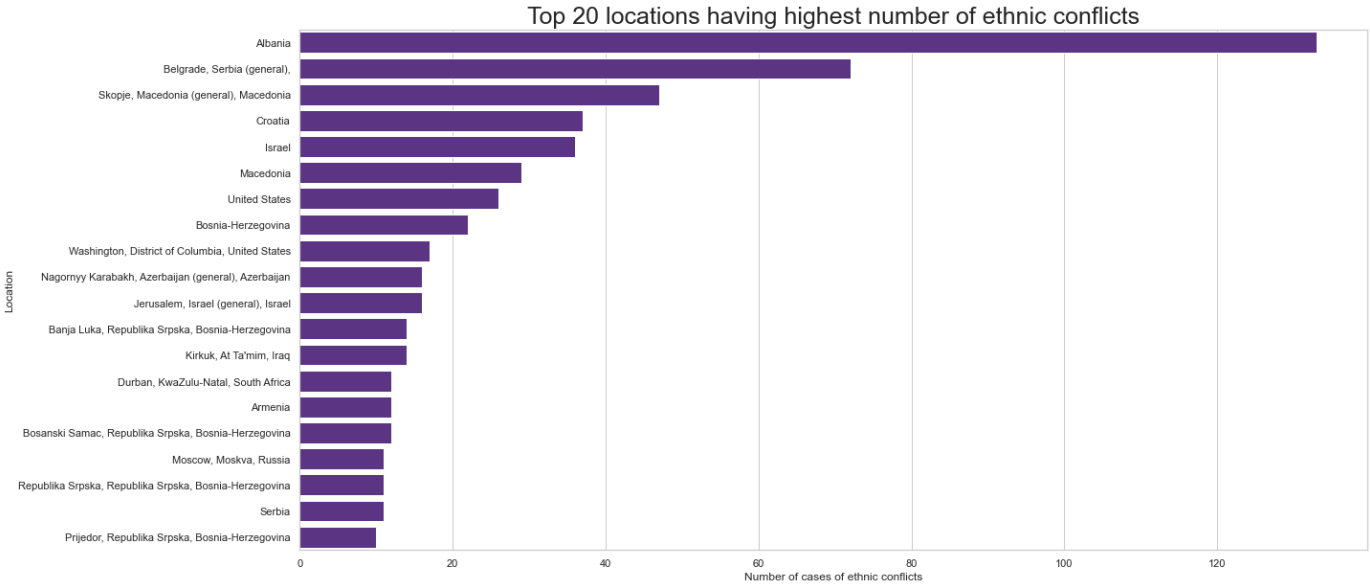
In [91]:
```
# wordplot for locations (based on number of incidents of ethnic conflicts)
d = dict(number_loc.values)
wordcloud = wc.WordCloud(width = 3000, height = 1500)
wordcloud.generate_from_frequencies(frequencies=d)
plt.figure(figsize=(25,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('Wordcloud for number of ethnic conflicts reported at Locations')
plt.show()
```

CONCLUSION

- Albania, belgrade serbia, skopje macedonia are some of the most volatile regions in 2001.

In [92]:
```
# top 20 countries having most incidents of ethnic conflicts
sns.set_theme(style="whitegrid")
plt.figure(figsize=(20,10))
plt.title('Top 20 locations having highest number of ethnic conflicts', fontsize=25)
ax = sns.barplot(x='frequency', y='location',data=number_loc.head(20), orient='h', color='#5c27
ax.set(xlabel='Number of cases of ethnic conflicts', ylabel='Location')
```

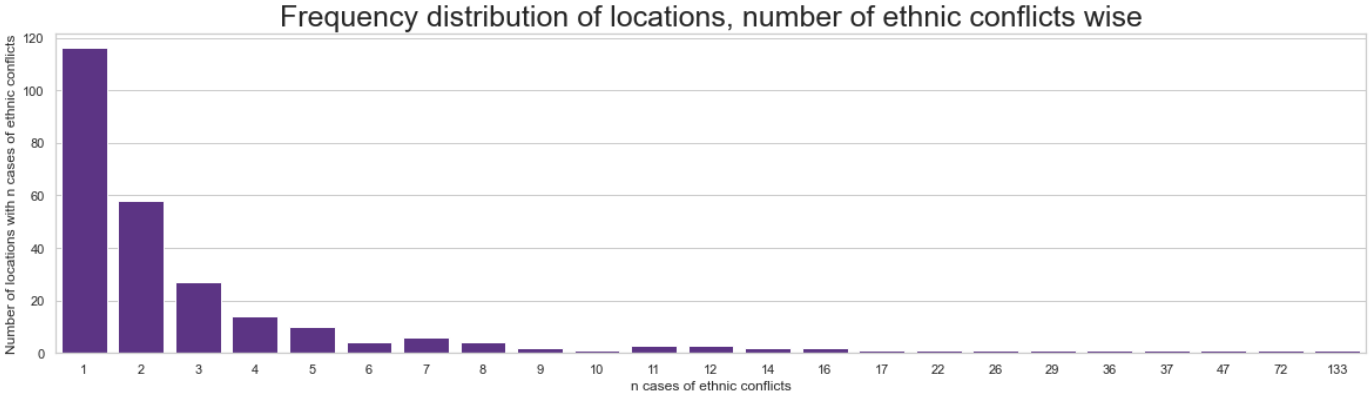
Out[92]:
```
[Text(0.5, 0, 'Number of cases of ethnic conflicts'), Text(0, 0.5, 'Location')]
```



- Albania is at top with 133 ethnic conflict in year 2001, followed by belgrade serbia with 72 conflicts. One thing to note is that the number of conflicts at a particular location decreases as we go down in graph follows a kind of exponential pattern.

In [93]:
```
# count of countries having n ethnic conflicts in year 2001
plt.figure(figsize=(20,5))
plt.title('Frequency distribution of locations, number of ethnic conflicts wise', fontsize=25)
ax = sns.countplot(x='frequency', data=number_loc, orient='h', color='#5c2791')
ax.set(xlabel='n cases of ethnic conflicts', ylabel='Number of locations with n cases of ethnic
```

Out[93]:
```
[Text(0.5, 0, 'n cases of ethnic conflicts'),
 Text(0, 0.5, 'Number of locations with n cases of ethnic conflicts')]
```



CONCLUSION

- from here it is seen that most of the locations have reported only 1 instance of ethnic conflict in year 2001 (around 115 locations have only 1 case of ethnic conflict), then there is steep drop in number of locations having 2 ethnic conflicts and the trend follows for subsequent location and then it attains constant behaviour.

Analysis of actors involved

Analysis considering actors as:

- source
- target
- actor (no differentiation between actors as source and target but take them as entity involved in ethnic conflict)

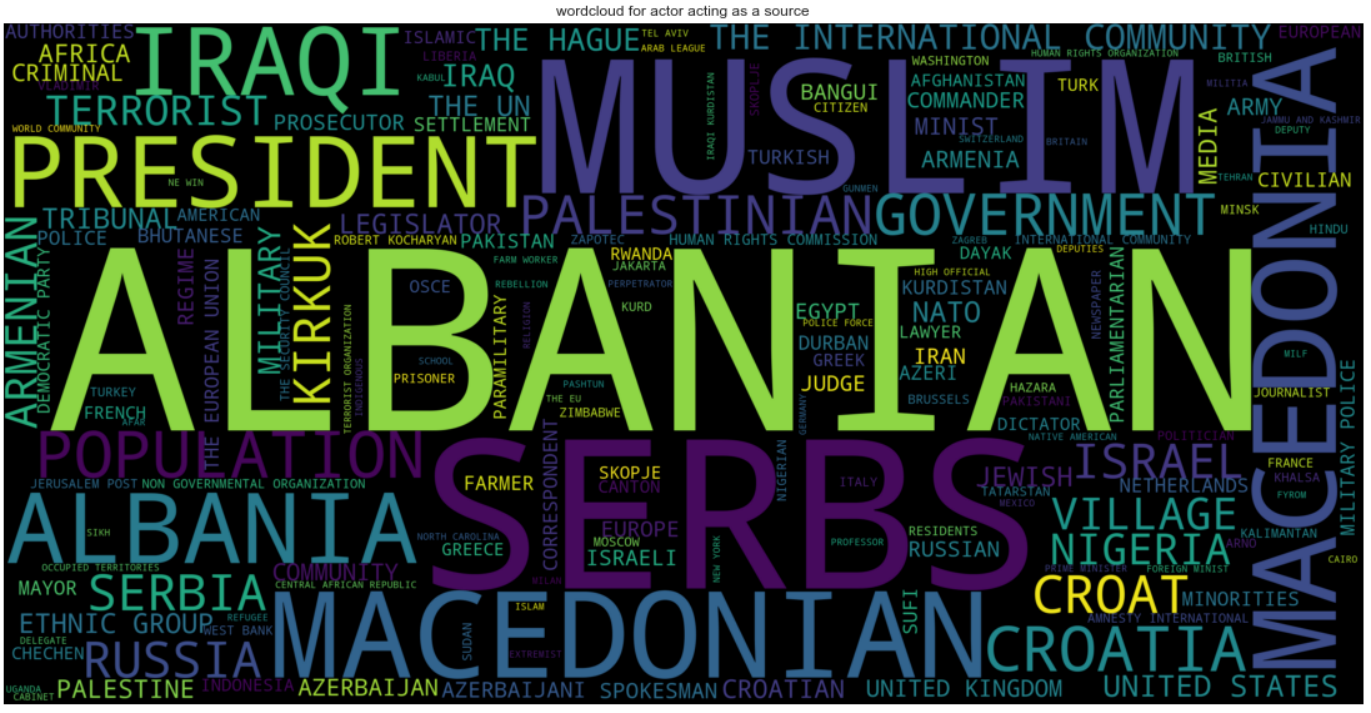
In [94]:
```
# Actor1Name as source
src = ec.Actor1Name.value_counts()
```

```
src = src.to_frame().reset_index().rename(columns={'index':'source','Actor1Name':'frequency'})

# Actor2Name as target
tgt = ec.Actor2Name.value_counts()
tgt = tgt.to_frame().reset_index().rename(columns={'index':'target','Actor2Name':'frequency'})

# Actor1Name and Actor2Name as common actors
comn = {}
for i,j in zip(list(src.source),list(src.frequency)):
    if i not in comn:
        comn[i] = int(j)

for i,j in zip(list(tgt.target),list(tgt.frequency)):
    if i not in comn:
        comn[i] = int(j)
    else:
        comn[i] = int(j) + comn.get(i)

comn = {'actors':list(comn.keys()),'frequency':list(comn.values())}
comn = pd.DataFrame.from_dict(comn)
```

Actor1Name as a source

In [95]:

```
# wordplot for actors involved in ethnic conflicts as source
d = dict(src.values)
wordcloud = wc.WordCloud(width = 3000, height = 1500)
wordcloud.generate_from_frequencies(frequencies=d)
plt.figure(figsize=(25,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('wordcloud for actor acting as a source')
plt.show()
```

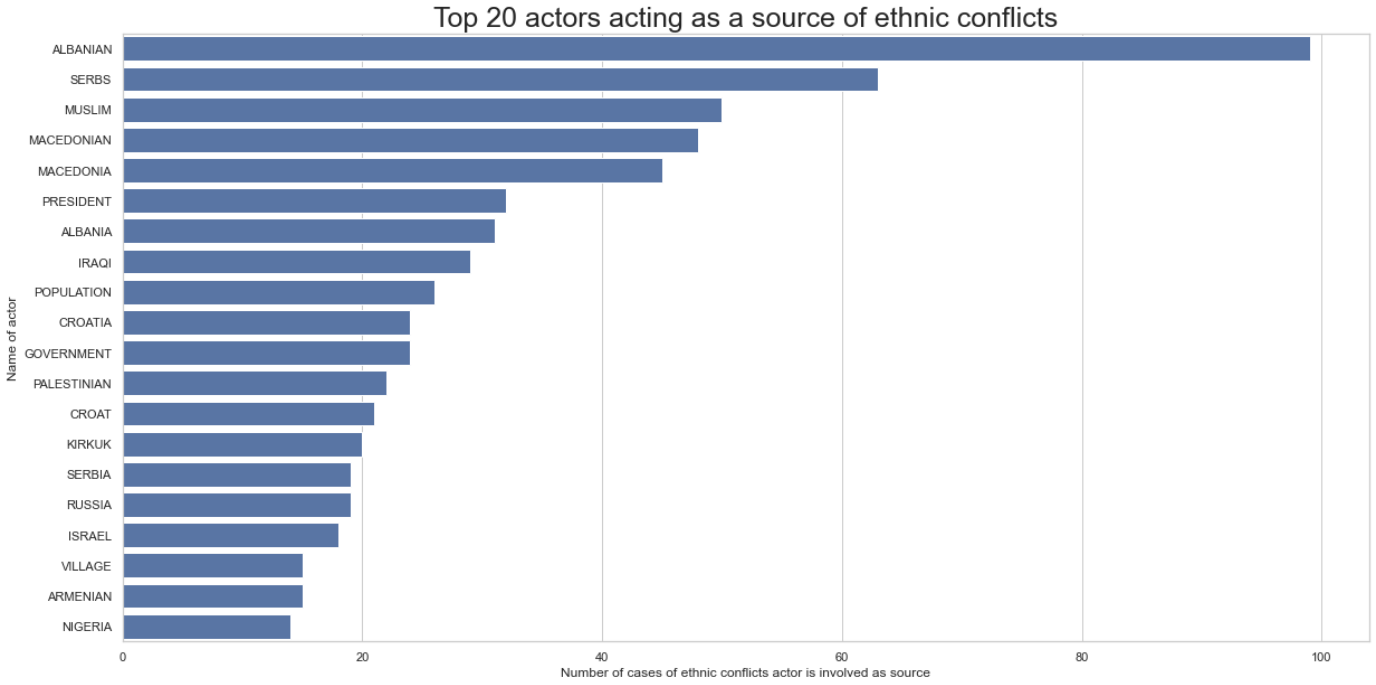


In [96]:

```
# Top 20 actors acting as a source of ethnic conflicts
sns.set_theme(style="whitegrid")
plt.figure(figsize=(20,10))
plt.title('Top 20 actors acting as a source of ethnic conflicts', fontsize=25)
ax = sns.barplot(x='frequency', y='source',data=src.head(20), orient='h', color='b')
ax.set(xlabel='Number of cases of ethnic conflicts actor is involved as source', ylabel='Name c
```

Out[96]:

```
[Text(0.5, 0, 'Number of cases of ethnic conflicts actor is involved as source'),
 Text(0, 0.5, 'Name of actor')]
```

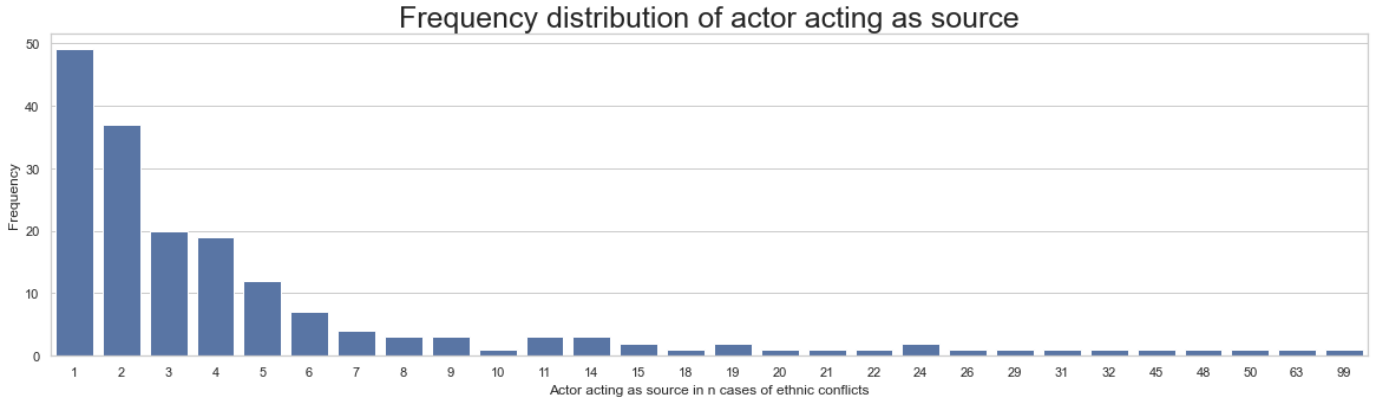


In [97]:

```
# Frequency distribution of actor acting as source
plt.figure(figsize=(20,5))
plt.title('Frequency distribution of actor acting as source', fontsize=25)
ax = sns.countplot(x='frequency', data=src, orient='h', color='b')
ax.set(xlabel='Actor acting as source in n cases of ethnic conflicts', ylabel='Frequency')
```

Out[97]:

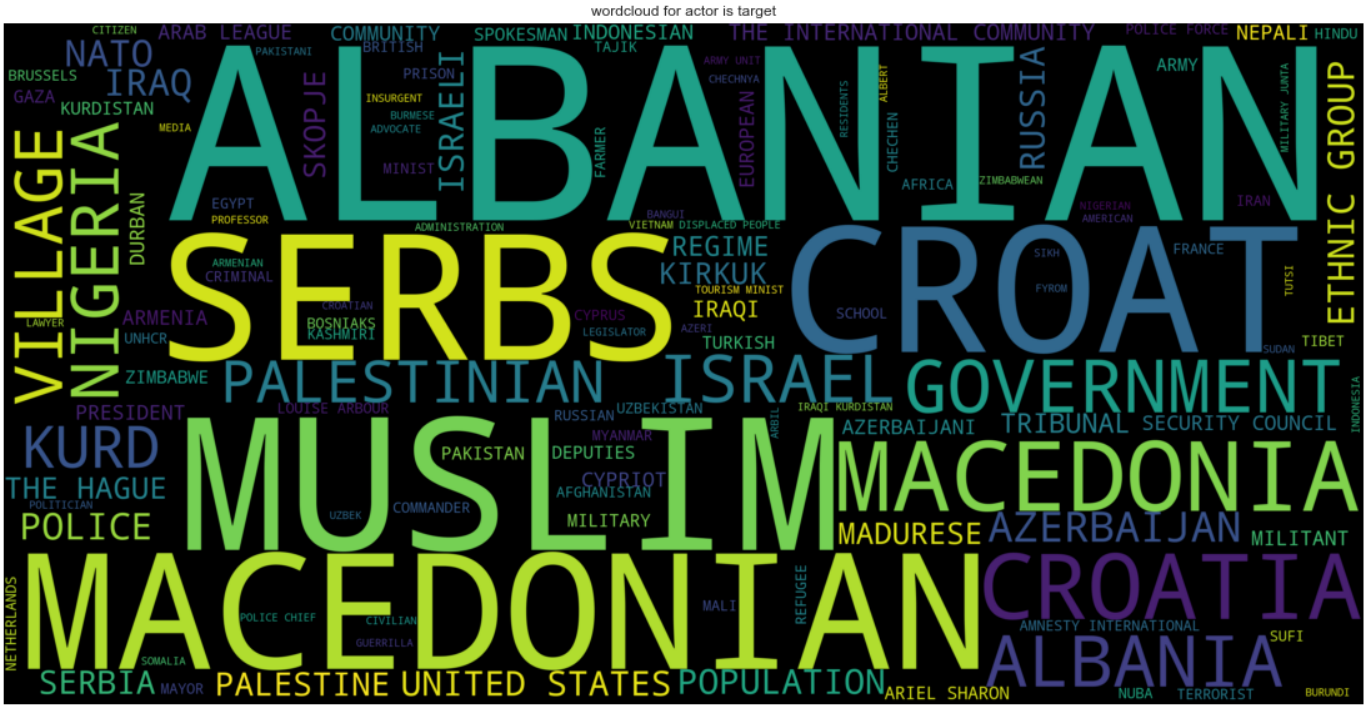
```
[Text(0.5, 0, 'Actor acting as source in n cases of ethnic conflicts'),
 Text(0, 0.5, 'Frequency')]
```



Actor2Name as a target

In [98]:

```
# wordplot for actors involved in ethnic conflicts as target
d = dict(tgt.values)
wordcloud = wc.WordCloud(width = 3000, height = 1500)
wordcloud.generate_from_frequencies(frequencies=d)
plt.figure(figsize=(25,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('wordcloud for actor is target')
plt.show()
```

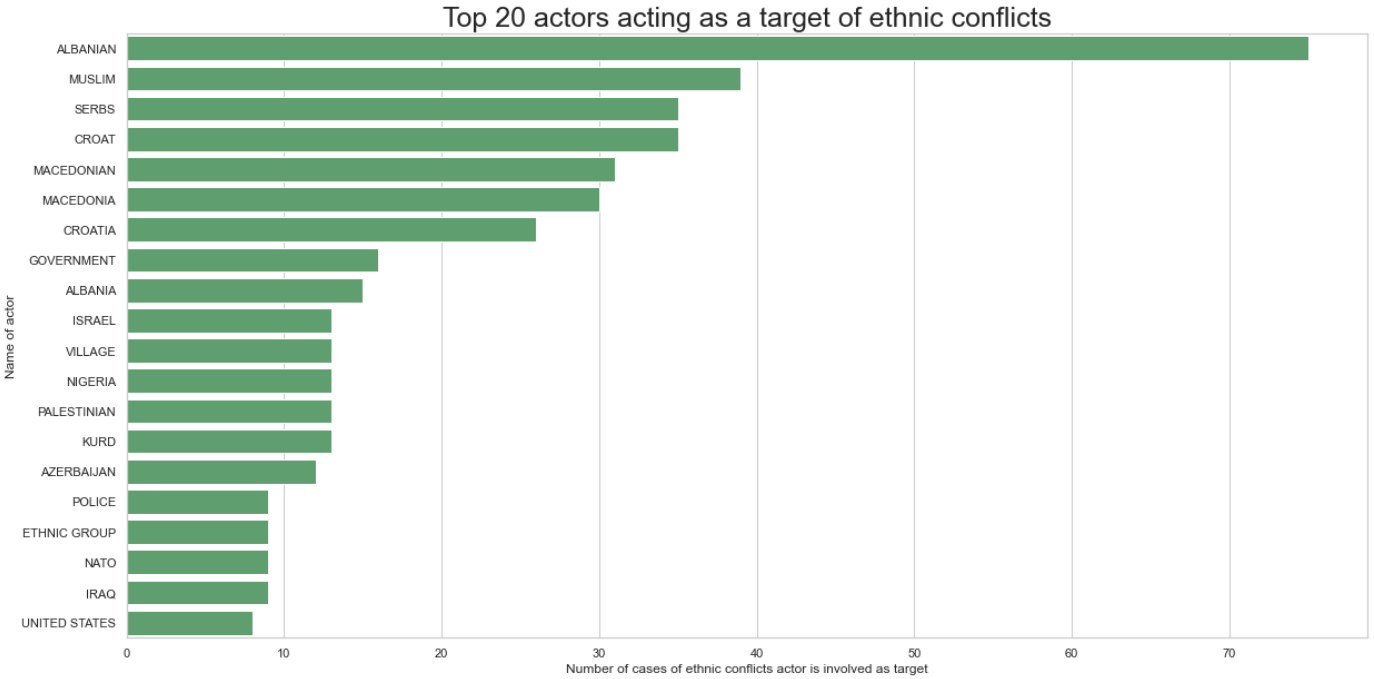


In [99]:

```
# Top 20 actors acting as a target of ethnic conflicts
sns.set_theme(style="whitegrid")
plt.figure(figsize=(20,10))
plt.title('Top 20 actors acting as a target of ethnic conflicts', fontsize=25)
ax = sns.barplot(x='frequency', y='target',data=tgt.head(20), orient='h', color='g')
ax.set(xlabel='Number of cases of ethnic conflicts actor is involved as target', ylabel='Name c
```

Out[99]:

```
[Text(0.5, 0, 'Number of cases of ethnic conflicts actor is involved as target'),
 Text(0, 0.5, 'Name of actor')]
```

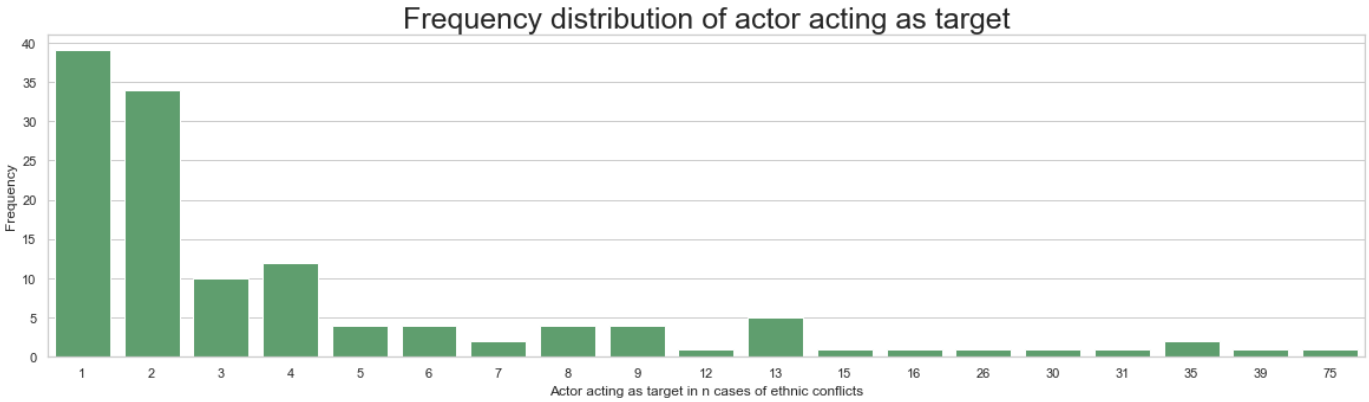


In [100…

```
# Frequency distribution of actor acting as target
plt.figure(figsize=(20,5))
plt.title('Frequency distribution of actor acting as target', fontsize=25)
ax = sns.countplot(x='frequency', data=tgt, orient='h', color='g')
ax.set(xlabel='Actor acting as target in n cases of ethnic conflicts', ylabel='Frequency')
```

Out[100…

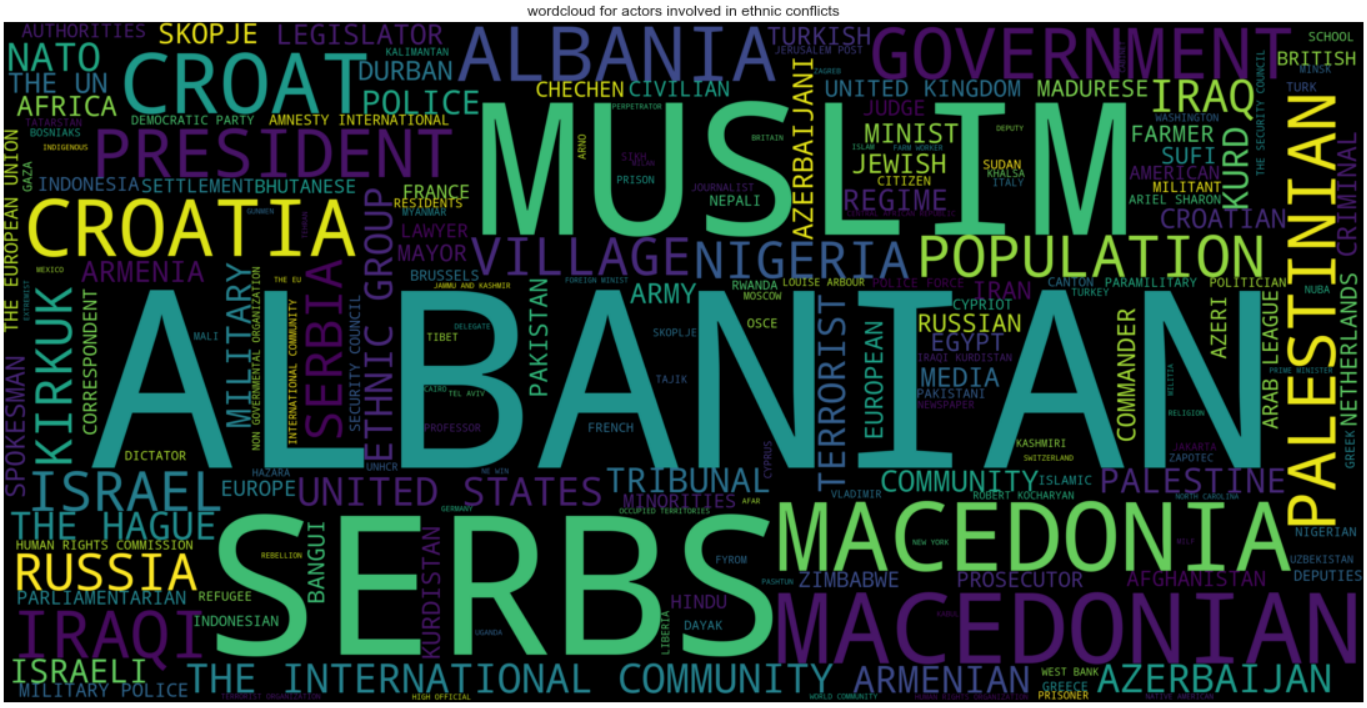
```
[Text(0.5, 0, 'Actor acting as target in n cases of ethnic conflicts'),
 Text(0, 0.5, 'Frequency')]
```



Actor1Name and Actor2Name as an Actors

In [101…

```
# wordplot for actors involved in ethnic conflicts
d = dict(comn.values)
wordcloud = wc.WordCloud(width = 3000, height = 1500)
wordcloud.generate_from_frequencies(frequencies=d)
plt.figure(figsize=(25,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('wordcloud for actors involved in ethnic conflicts')
plt.show()
```

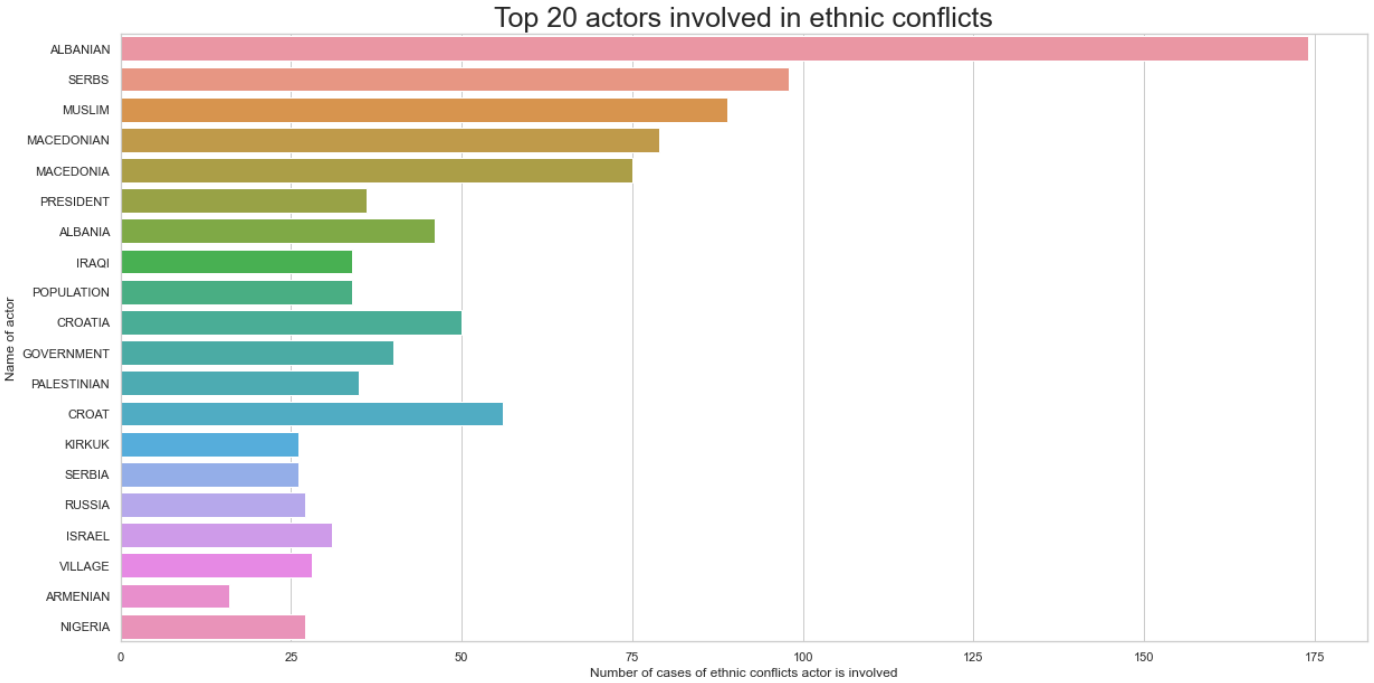


In [102…

```
# Top 20 actors involved in ethnic conflicts
sns.set_theme(style="whitegrid")
plt.figure(figsize=(20,10))
plt.title('Top 20 actors involved in ethnic conflicts', fontsize=25)
ax = sns.barplot(x='frequency', y='actors',data=comn.head(20), orient='h')
ax.set(xlabel='Number of cases of ethnic conflicts actor is involved', ylabel='Name of actor')
```

Out[102…

```
[Text(0.5, 0, 'Number of cases of ethnic conflicts actor is involved'),
 Text(0, 0.5, 'Name of actor')]
```

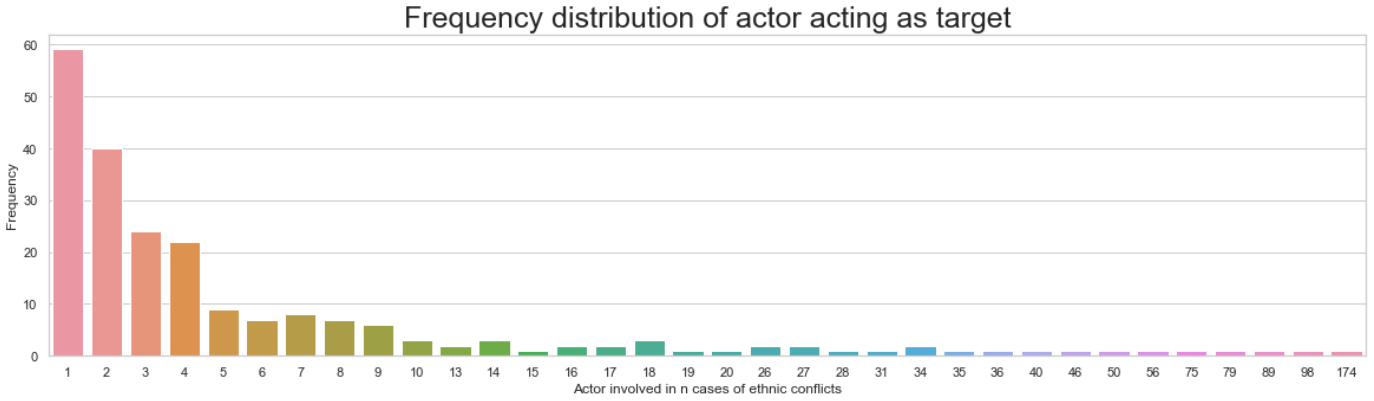


In [103…

```
# Frequency distribution of actor acting as target
plt.figure(figsize=(20,5))
plt.title('Frequency distribution of actor acting as target', fontsize=25)
ax = sns.countplot(x='frequency', data=comn, orient='h')
ax.set(xlabel='Actor involved in n cases of ethnic conflicts', ylabel='Frequency')
```

Out[103…

```
[Text(0.5, 0, 'Actor involved in n cases of ethnic conflicts'),
 Text(0, 0.5, 'Frequency')]
```



Constructing a Network

In [104…

```
# creating list of unique nodes to add to graph
nodes = list(set(list(ec.Actor1Name) + list(ec.Actor2Name)))[1:]
# creating dataframe with edges between nodes
edge_df = ec[['Actor1Name', 'Actor2Name']]
# droping edges having null as Actor
edge_df.dropna(inplace=True)
edge_df['edges_col'] = list(zip(edge_df.Actor1Name, edge_df.Actor2Name))
edges = list(edge_df.edges_col)
```

```
C:\Users\HIIO\anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
ndexing.html#returning-a-view-versus-a-copy

C:\Users\HIIO\anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
```

```
ndexing.html#returning-a-view-versus-a-copy
  import sys
```

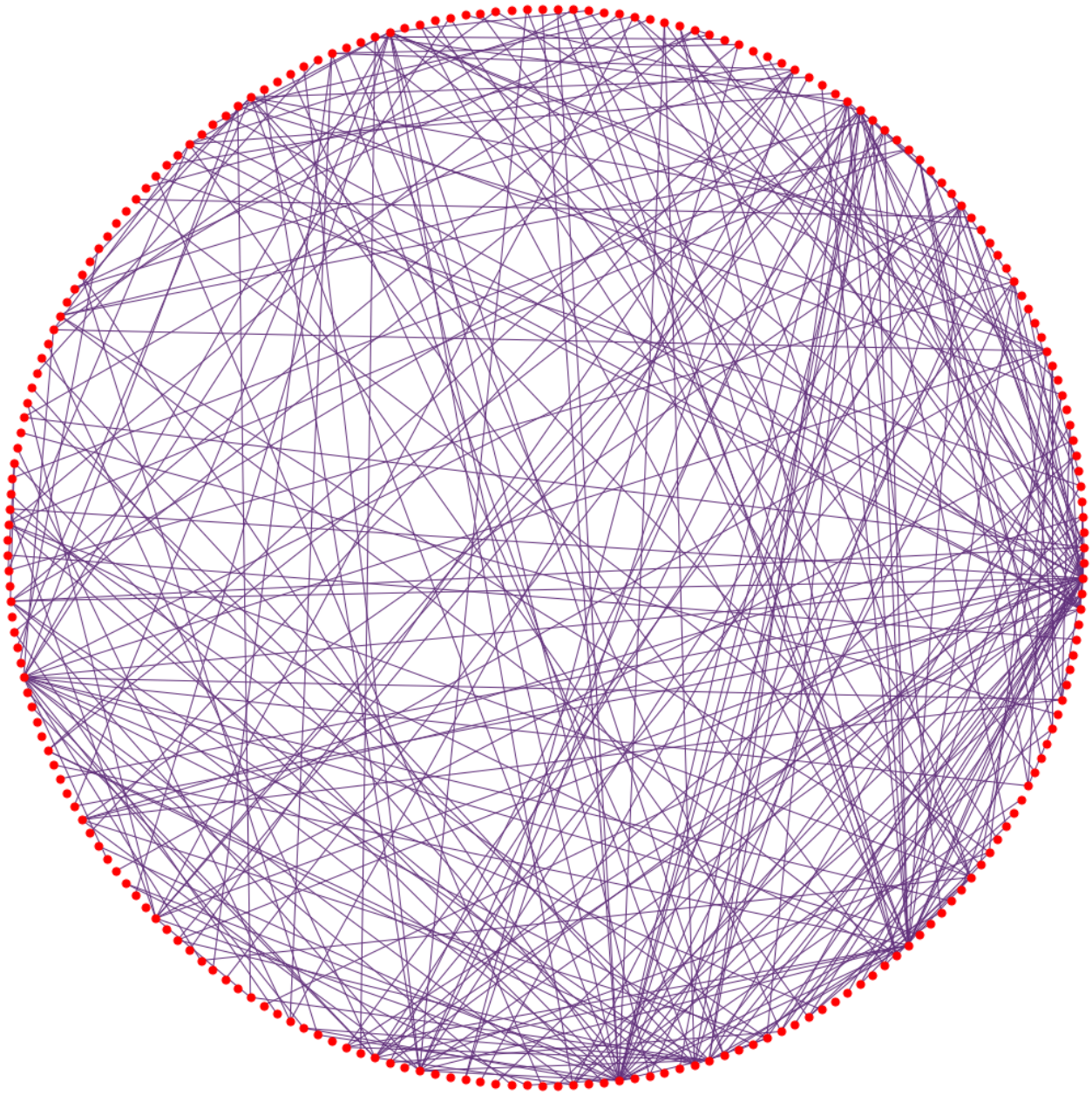
Creating undirected network taking Actor1Name and Actor2Name as actors in ethnic conflicts

In [105…

```
# creating undirected graph G and adding nodes and edges to it
G = nx.Graph()
G.add_nodes_from(nodes)
G.add_edges_from(edges)
```

In [106…

```
# plotting undirected network in a circular layout
plt.figure(figsize=(15,15))
nx.draw(G, node_color='red', edge_color='#63327d', node_size=50, with_labels=False, pos=nx.circ
```



CONCLUSION

- Added all the possible nodes from Actor1Name and Actor2Name to network. So there may be some nodes for which the degree can be 0, ie not participating in network but relevant because they are part of 2001 ethnic conflicts

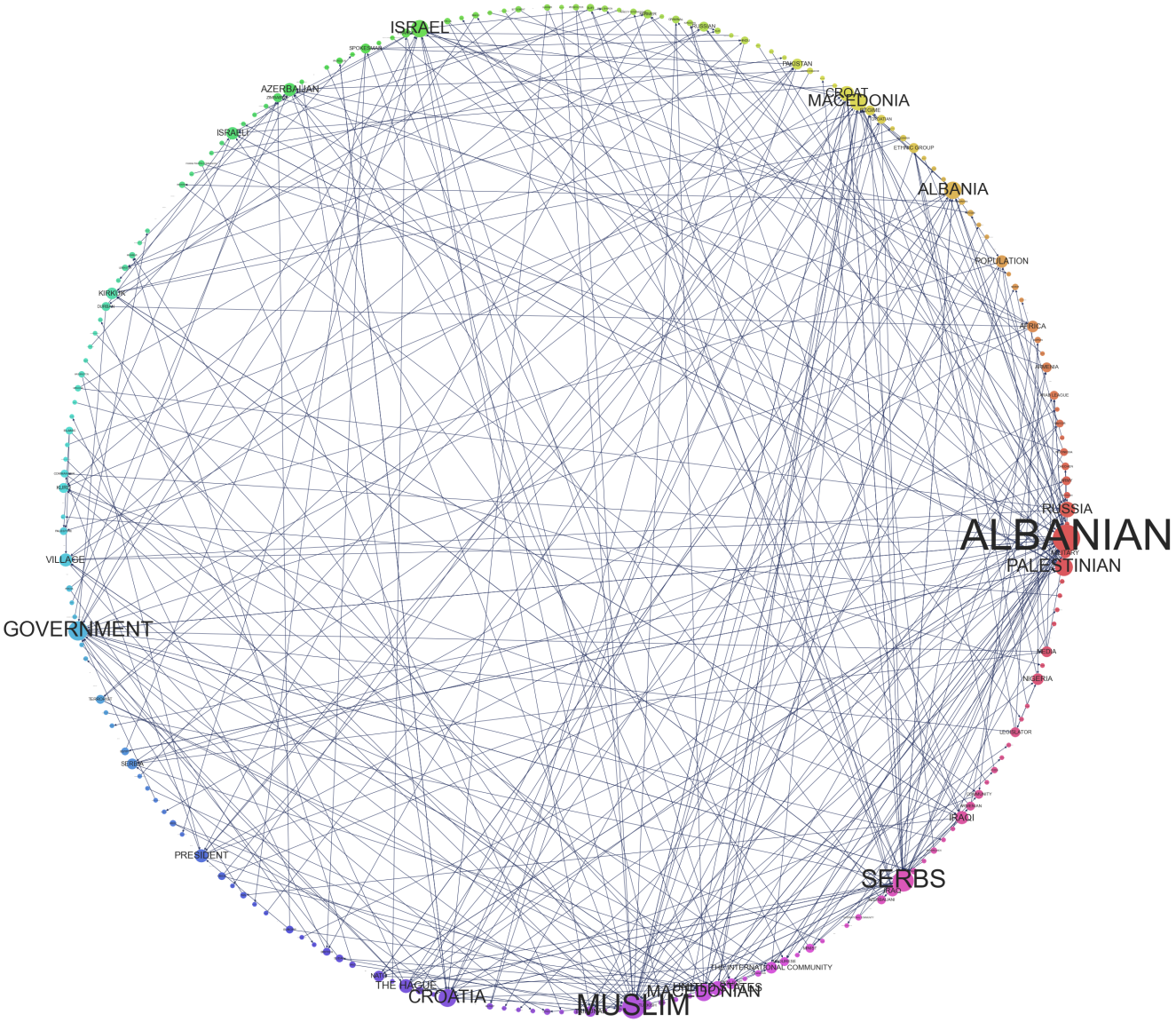
Creating directed network taking Actor1Name and Actor2Name as actors in ethnic conflicts

In [107…

```
# creating directed graph G and adding nodes and edges to it, here edges have directionality ie
DG = nx.DiGraph()
DG.add_nodes_from(nodes)
DG.add_edges_from(edges)
```

In [108…

```
# plotting directed network in a circular layout with different color for each node and font si
plt.figure(figsize=(40,40))
pos = nx.circular_layout(DG)
nx.draw(DG, pos, node_color=sns.color_palette('hls',219) ,edge_color='#293563' , node_size=[val
for node, (x, y) in pos.items():
    plt.text(x, y, node, fontsize=DG.degree[node]*2.5, ha='center', va='center')
```



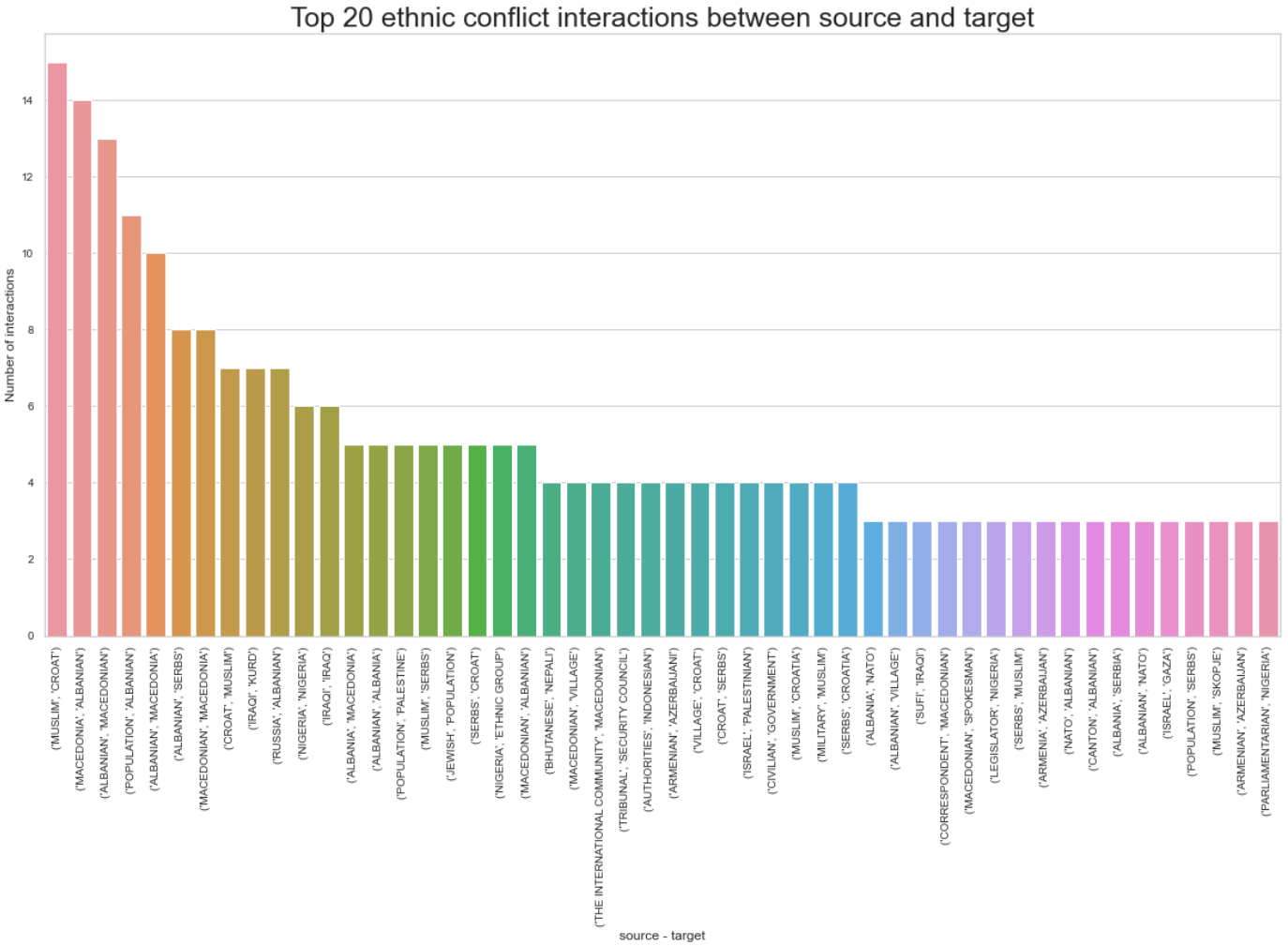
CONCLUSION

- Here the directed graph is formed, with nodes size and label proportional to their degrees. From here observe that there are some nodes which are participating very actively in network.

Source-Target Interactions

In [109…
```
st = edge_df['edges_col'].value_counts().reset_index()
# Top 20 actors involved in ethnic conflicts
sns.set_theme(style="whitegrid")
plt.figure(figsize=(20,10))
plt.title('Top 20 ethnic conflict interactions between source and target', fontsize=25)
ax = sns.barplot(x='index', y='edges_col', data=st.head(50), orient='v')
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)
ax.set(xlabel='source - target', ylabel='Number of interactions')
```

Out[109…
```
[Text(0.5, 0, 'source - target'), Text(0, 0.5, 'Number of interactions')]
```



CONCLUSION

- some communities are vary involved in ethnic conflicts with each others, and have multiple clash in year 2001

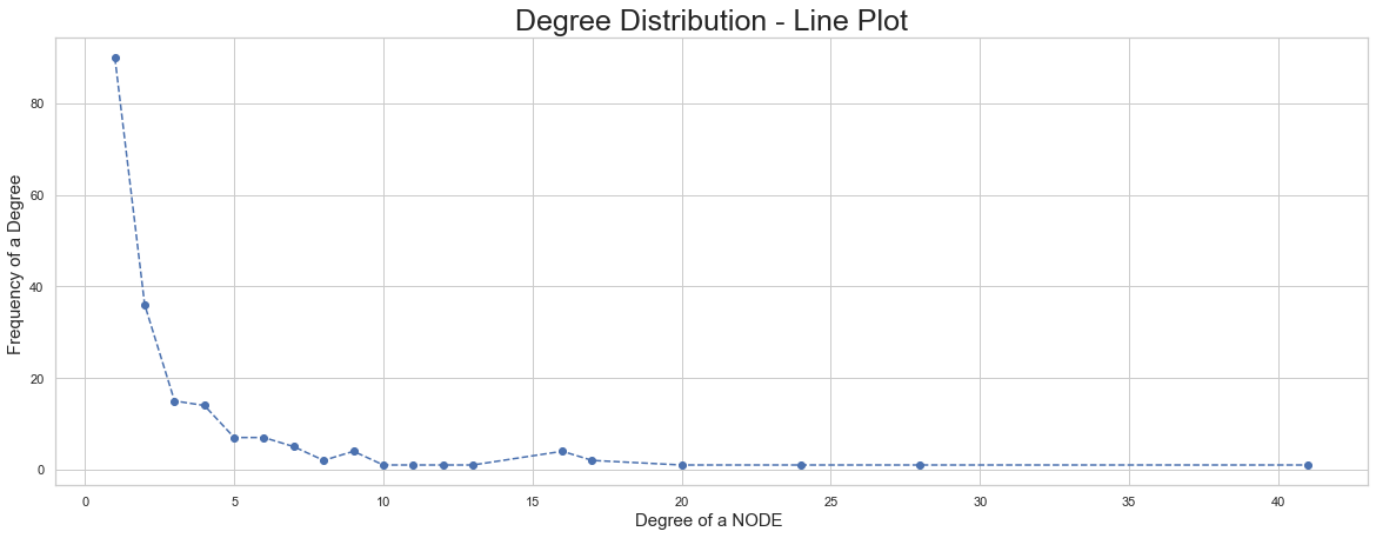
Degree Distribution and Power Law

In [110…
```
# getting degrees of each node to plot graph
degrees = list(dict(DG.degree()).values())
degree_dist = []
for i in set(degrees):
    degrees.count(i)
    degree_dist.append([i, degrees.count(i)])

# values of x and y and dropping the 0 degree nodes
x , y = [x[0] for x in degree_dist[1:]] , [x[1] for x in degree_dist[1:]]
```

In [111…
```
plt.figure(figsize=(20,7))
plt.plot(x, y, 'o--b')
plt.title('Degree Distribution - Line Plot', size = 25)
```

```
plt.xlabel('Degree of a NODE ', size = 15)
plt.ylabel('Frequency of a Degree ', size = 15)
```

Out[111… `Text(0, 0.5, 'Frequency of a Degree ')`



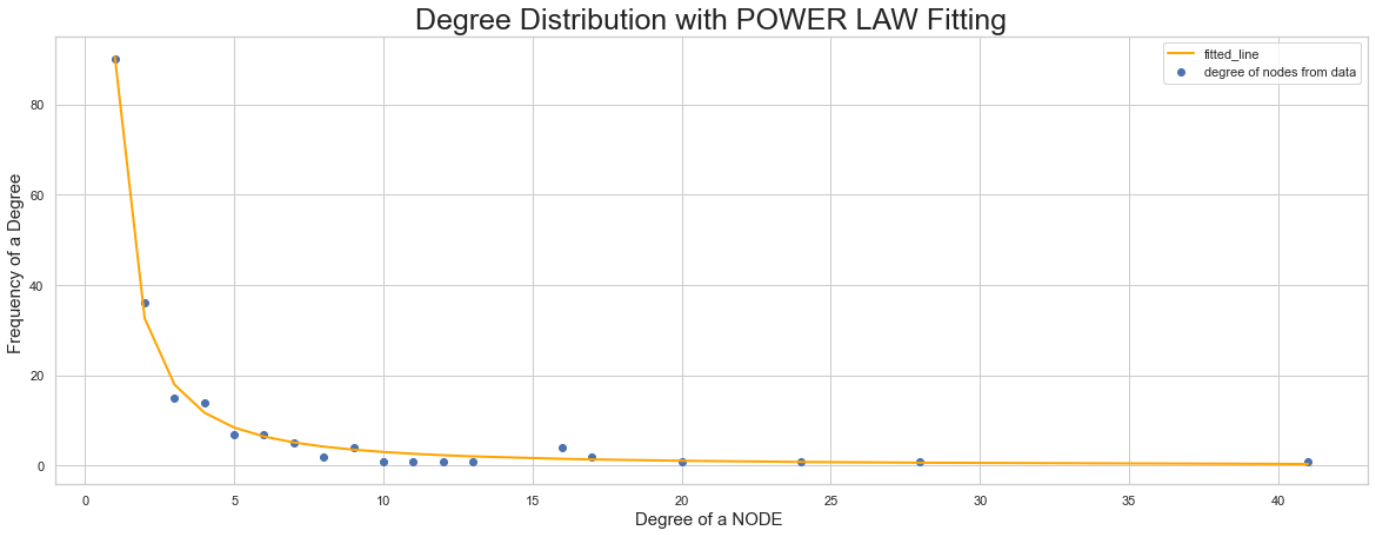
In [112…

```
# Fit the power law
def power_law(x, a, b):
    return a*np.power(x, b)

# Fit the power-law data, pars give the values of parameters ie a and b in this case, and cov g
pars, cov = curve_fit(f=power_law, xdata=x, ydata=y, p0=[0, 0], bounds=(-np.inf, np.inf))

# Get the standard deviations of the parameters (square roots of the # diagonal of the covarian
stdevs = np.sqrt(np.diag(cov))

# Calculate the residuals
res = y - power_law(x, *pars)
```

In [113…

```
# Plot the fit data as an overlay on the scatter data
plt.figure(figsize=(20,7))
plt.scatter(x,y, label='degree of nodes from data')
plt.title('Degree Distribution with POWER LAW Fitting', size = 25)
plt.xlabel('Degree of a NODE ', size = 15)
plt.ylabel('Frequency of a Degree ', size = 15)
plt.plot(x, power_law(x, *pars), linestyle='-', linewidth=2, color='orange', label='fitted_line
plt.legend()
```

Out[113… `<matplotlib.legend.Legend at 0x12d846d6308>`



In [114…

```
pars, stdevs
```

Out[114… (array([90.61051444, -1.47363555]), array([1.68557955, 0.04217985]))

CONCLUSION

- For power law of the form

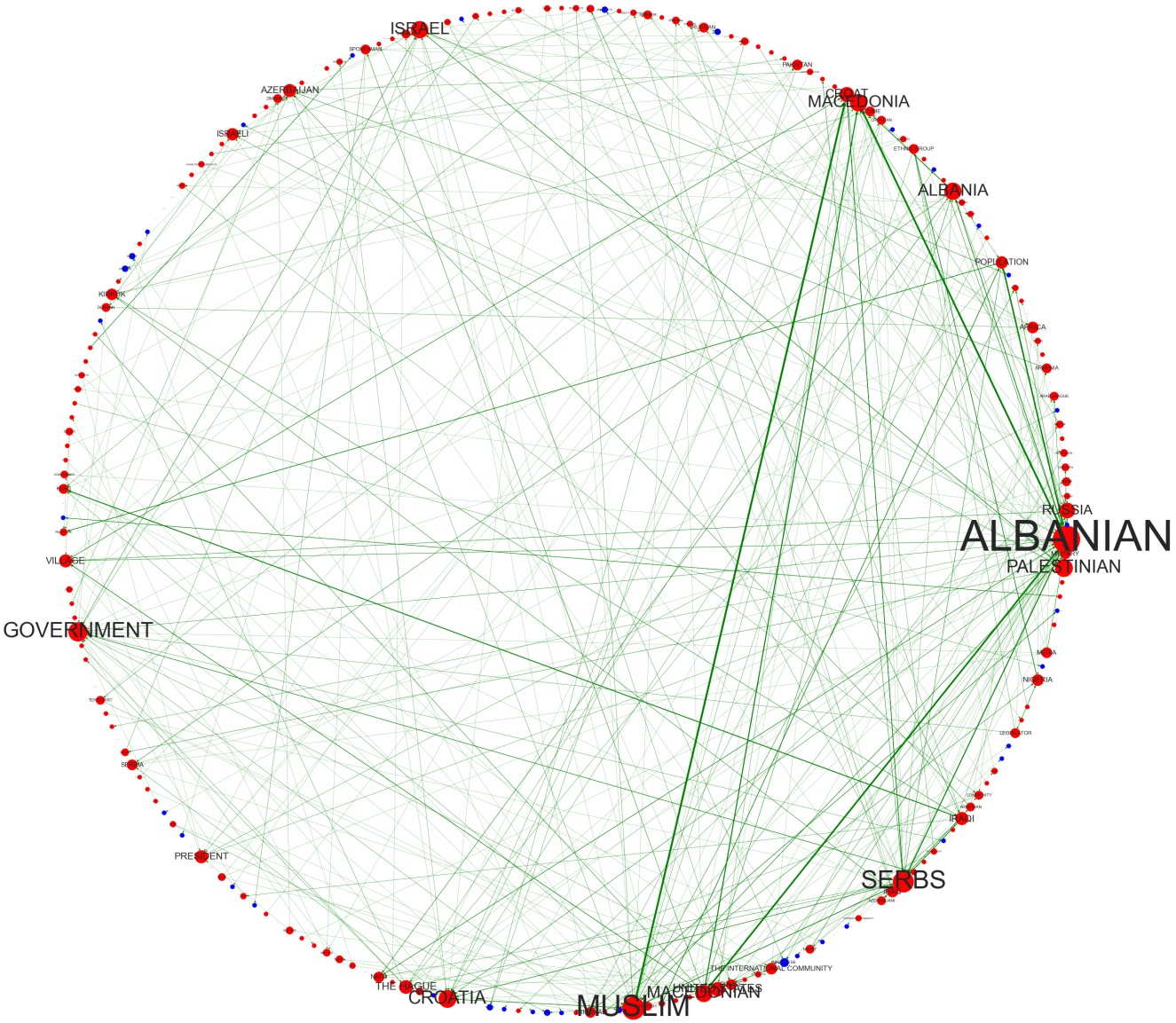
$$y = mx^b$$

- m = 90.61051444 +- 1.68557955
- b = -1.47363555 +- 0.04217985

Different color source and target with edge attribute

In [115…

```
# here nodes present in both source and targets are only represented in source color but indegr
weighted_edges = [(i[0],i[1],j) for i,j in zip(list(st['index']),list(st['edges_col']))]

DGW = nx.DiGraph()
DGW.add_nodes_from(nodes)
DGW.add_weighted_edges_from(weighted_edges)

weights = [DGW[u][v]['weight']*0.3 for u,v in DGW.edges]

plt.figure(figsize=(40,40))
pos = nx.circular_layout(DGW)
# source nodes in red  and target node is blue
nx.draw(DGW, pos, node_color=["red" if node in list(ec.Actor1Name) else 'blue' for node in DGW.
for node, (x, y) in pos.items():
    plt.text(x, y, node, fontsize=DGW.degree[node]*2.5, ha='center', va='center')
```



CONCLUSION

- In above network the actors which are more active (ie higher degree) are bigger in size and interaction between actors is represented by edge thickness which is proportional to edge weight

Degree centrality, Closeness centrality and Normalized betweenness centrality

Properties of Network

In [116…

```
# take largest connected component of network to calculate diameter, average shortest path leng
G_connected_nodes = sorted(nx.connected_components(G), key=len, reverse=True)
G_connected = G.subgraph(G_connected_nodes[0])
```

In [117…

```
print(nx.info(G))
```

```
Name:
Type: Graph
Number of nodes: 219
Number of edges: 312
Average degree:   2.8493
```

In [118…

```
print('For network of ethnic conflicts following are the network properties \n')
print('ORDER                          :  {}'.format(G.order()))
print('SIZE                           :  {}'.format(G.size()))
print('DENSITY                        :  {}'.format(nx.density(G)))
print('AVERAGE DEGREE                 :  {}'.format(2.8493))
print('DIAMETER                       :  {}'.format(nx.algorithms.distance_measures.diameter(G_c
print('TRANSITIVITY                   :  {}'.format(nx.transitivity(G)))
print('Average shortest path length   :  {}'.format(nx.average_shortest_path_length(G_connected)
print('Average clustering coefficient:  {}'.format(nx.average_clustering(G)))
print('Average node connectivity      :  {}'.format(nx.average_node_connectivity(G)))
```

```
For network of ethnic conflicts following are the network properties

ORDER                          :  219
SIZE                           :  312
DENSITY                        :  0.013070252607766746
AVERAGE DEGREE                 :  2.8493
DIAMETER                       :  10
TRANSITIVITY                   :  0.12585499316005472
Average shortest path length   :  4.028441558441559
Average clustering coefficient:  0.1429386528064548
Average node connectivity      :  0.893510954714926
```

In [119…

```
# Total attacks
Number_of_attacks = 0
for i in weighted_edges:
    Number_of_attacks = Number_of_attacks + int(i[2])
Number_of_attacks
```

Out[119… 659

CONCLUSION

- Number of organizations = number of nodes = 219
- Number of Attacks = sum of weights of edges = 659

For Node 15

In [120…

```
# The 15th node in graph is
count = 0
for i in G_connected.nodes():
    count += 1
    if count == 15:
        node_15 = i
        break
node_15
```

Out[120… 'POPULATION'

In [121…

```
# degree centrality of node 15
nx.degree_centrality(G)[node_15]
```

Out[121… 0.03669724770642202

In [122…

```
# closeness centrality of node 15
nx.closeness_centrality(G)[node_15]
```

Out[122… 0.26556077764867064

In [123…

```
# normalized betweenness centrality of node 15
nx.betweenness_centrality(G, k=None, normalized=True, weight=None, endpoints=False, seed=None)[
```

Out[123… 0.015729444722409162

CONCLUSION

- For node 15
- degree centrality : 0.03669724770642202
- closeness centrality : 0.26556077764867064
- normalized betweenness centrality : 0.015729444722409162

Subgroups and Community Detection

In [124…
```
# getting the connected nodes and removing nodes with degree 0, to get proper subgroups
G_connected_nodes = sorted(nx.connected_components(G), key=len, reverse=True)
G_connected_nodes = [x for x in G_connected_nodes if len(x)>1]
```

In [125…
```
print('Number of Subgroups in network are {}'.format(len(G_connected_nodes)))
```

```
Number of Subgroups in network are 10
```

In [126…
```
# size of subgroups
for i in G_connected_nodes:
    print(len(i))
```

```
176
2
2
2
2
2
2
2
2
2
```

In [127…
```
girvan_newman = nx.algorithms.community.centrality.girvan_newman(G)
girvan_newman_communities = list(c for c in next(girvan_newman))
girvan_newman_communities = [x for x in girvan_newman_communities if len(x)>1]
```

In [128…
```
print('Number of Girvan Newman subcommunities in network are {}'.format(len(girvan_newman_commu
```

```
Number of Girvan Newman subcommunities in network are 11
```

In [129…
```
# size of girvan newman communities
for i in girvan_newman_communities:
    print(len(i))
```

```
148
28
2
2
2
2
2
2
2
2
2
```

CONCLUSION

- There are 10 subgroups in network with one having 176 nodes and rest have 2 nodes each, means one subgroup is more involved in conflicts with one another while 2 nodes subnetworks are involved in conflicts among themselves

- There are 11 subcommunities according to girvan-newman algorithm and two are big sized with having 148 and 28 nodes, others have 2 nodes each

Strongly Connected Component and Weakly Connected Component

In [130…
```
strongly_connected = nx.algorithms.components.strongly_connected_components(DG)
strongly_connected_components = list(c for c in next(strongly_connected))
len(strongly_connected_components)
```

Out[130… 1

In [131…
```
weakly_connected = nx.algorithms.components.weakly_connected_components(DG)
weakly_connected_components = list(c for c in next(weakly_connected))
len(weakly_connected_components)
```

Out[131… 176

CONCLUSION

- there is only 1 strongly connected component in network
- there are 176 weakly connected component in network

SUMMARY

The ethnic conflicts are complex matters and important field of social dynamics. But through detailed analysis of ethnic conflicts in 2001 we can say that in year 2001:

- There are 219 nodes in network, means in total there are 219 actors
- The total edges weight in network is 659, means there are in total 659 reported ethnic conflicts in 2001 where both participants are known (in data).
- Most of occurrences of ethnic conflicts are reported from few location and occurrences of ethnic conflicts location wise follows power law.
- Some of the actors participating in ethnic conflicts are more active than others and have very high degrees, so essentially following power law.
- There are some pair of communities which have very high incidents ethnic conflicts between them.
- The average degree for network is 2.849, means each actor on average is participating in 3 conflicts.
- The degree distribution follows the power law, and values of parameters are m = 90.61051444 +- 1.68557955 and b = -1.47363555 +- 0.04217985
- There are 10 subgroups in network, means there are 10 sub networks were there is high reporting of ethnic conflicts between nodes.
- There are 11 subcommunities according to Girvan Newman algorithm.
- There is only 1 strongly connected component in network.
- There are 176 weakly connected components in network.

REFERENCES

1. Eriksen, T. H. (2012). Ethnicity. The Wiley-Blackwell Encyclopedia of Globalization.

2. P. R. Monge and N. S. Contractor, Theories of communication networks. New York: Oxford University Press, 2003.
3. Network Science Book by Albert-László Barabási