Preliminary Report mini-project

VENT

Express your feelings and connect with people who care

February 2020

Data-Driven Social Analytics

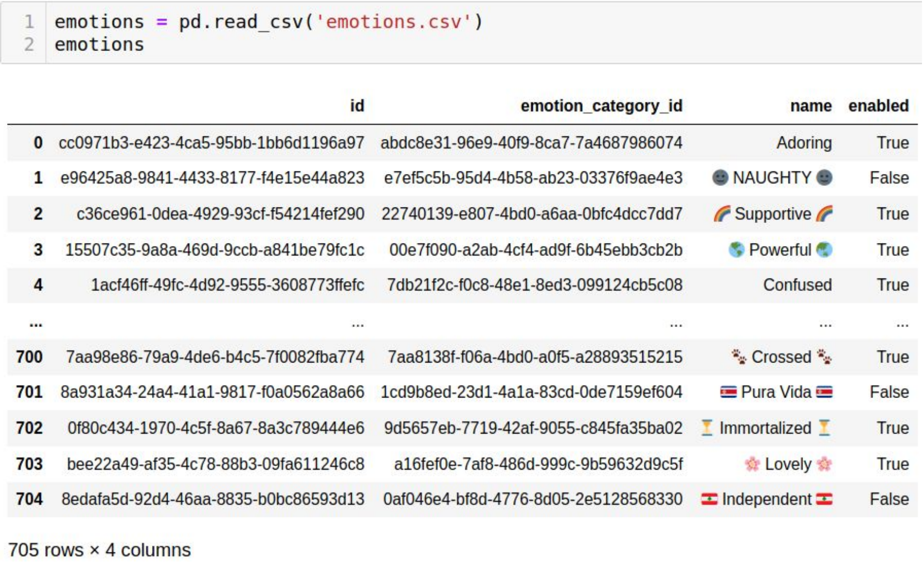
Jonatan Koren

The data

the data contains 4 CSV files:

- Emotions
- Emotions categories
- Metadata
- Edge list

Emotions

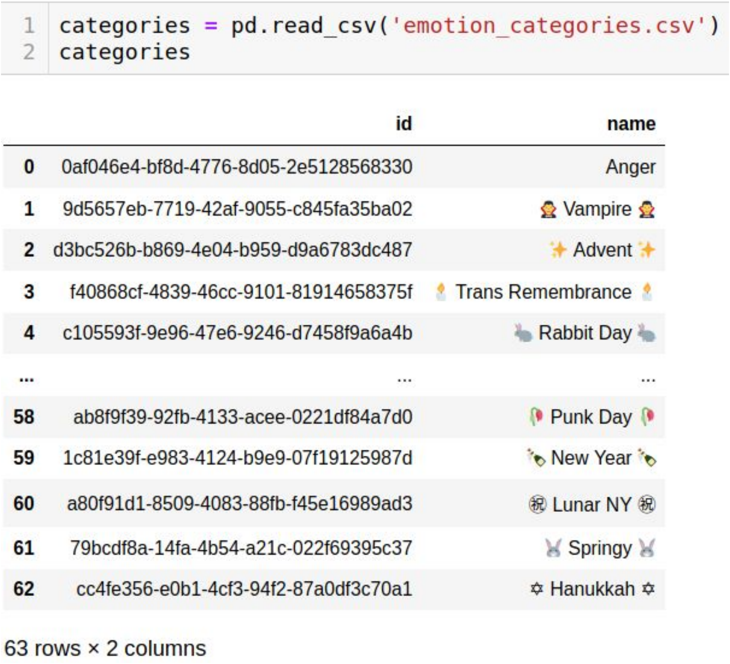


```
emotions = pd.read_csv('emotions.csv')
emotions
```

	id	emotion_category_id	name	enabled
0	cc0971b3-e423-4ca5-95bb-1bb6d1196a97	abdc8e31-96e9-40f9-8ca7-7a4687986074	Adoring	True
1	e96425a8-9841-4433-8177-f4e15e44a823	e7ef5c5b-95d4-4b58-ab23-03376f9ae4e3	🌚 NAUGHTY 🌚	False
2	c36ce961-0dea-4929-93cf-f54214fef290	22740139-e807-4bd0-a6aa-0bfc4dcc7dd7	🌈 Supportive 🌈	True
3	15507c35-9a8a-469d-9ccb-a841be79fc1c	00e7f090-a2ab-4cf4-ad9f-6b45ebb3cb2b	🌎 Powerful 🌏	True
4	1acf46ff-49fc-4d92-9555-3608773ffefc	7db21f2c-f0c8-48e1-8ed3-099124cb5c08	Confused	True
...	...	...	...	...
700	7aa98e86-79a9-4de6-b4c5-7f0082fba774	7aa8138f-f06a-4bd0-a0f5-a28893515215	🐾 Crossed 🐾	True
701	8a931a34-24a4-41a1-9817-f0a0562a8a66	1cd9b8ed-23d1-4a1a-83cd-0de7159ef604	🇨🇷 Pura Vida 🇨🇷	False
702	0f80c434-1970-4c5f-8a67-8a3c789444e6	9d5657eb-7719-42af-9055-c845fa35ba02	⏳ Immortalized ⏳	True
703	bee22a49-af35-4c78-88b3-09fa611246c8	a16fef0e-7af8-486d-999c-9b59632d9c5f	🌸 Lovely 🌸	True
704	8edafa5d-92d4-46aa-8835-b0bc86593d13	0af046e4-bf8d-4776-8d05-2e5128568330	🇱🇧 Independent 🇱🇧	False

705 rows × 4 columns

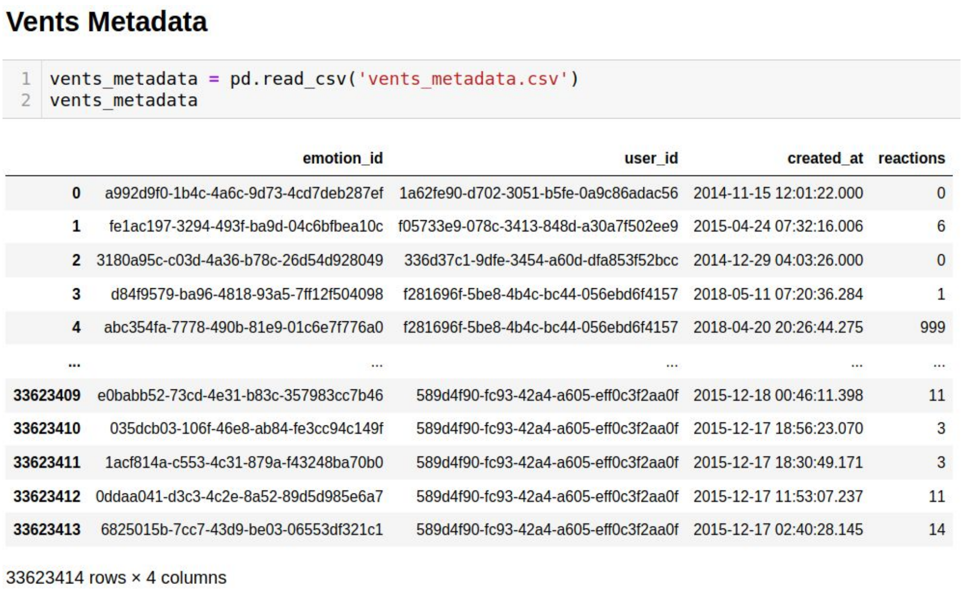
Emotions Categories



```
categories = pd.read_csv('emotion_categories.csv')
categories
```

	id	name
0	0af046e4-bf8d-4776-8d05-2e5128568330	Anger
1	9d5657eb-7719-42af-9055-c845fa35ba02	🧛 Vampire 🧛
2	d3bc526b-b869-4e04-b959-d9a6783dc487	✨ Advent ✨
3	f40868cf-4839-46cc-9101-81914658375f	🕯️ Trans Remembrance 🕯️
4	c105593f-9e96-47e6-9246-d7458f9a6a4b	🐇 Rabbit Day 🐇
...	...	...
58	ab8f9f39-92fb-4133-acee-0221df84a7d0	🥀 Punk Day 🥀
59	1c81e39f-e983-4124-b9e9-07f19125987d	🍾 New Year 🍾
60	a80f91d1-8509-4083-88fb-f45e16989ad3	㊗️ Lunar NY ㊗️
61	79bcdf8a-14fa-4b54-a21c-022f69395c37	🐰 Springy 🐰
62	cc4fe356-e0b1-4cf3-94f2-87a0df3c70a1	✡️ Hanukkah ✡️

63 rows × 2 columns



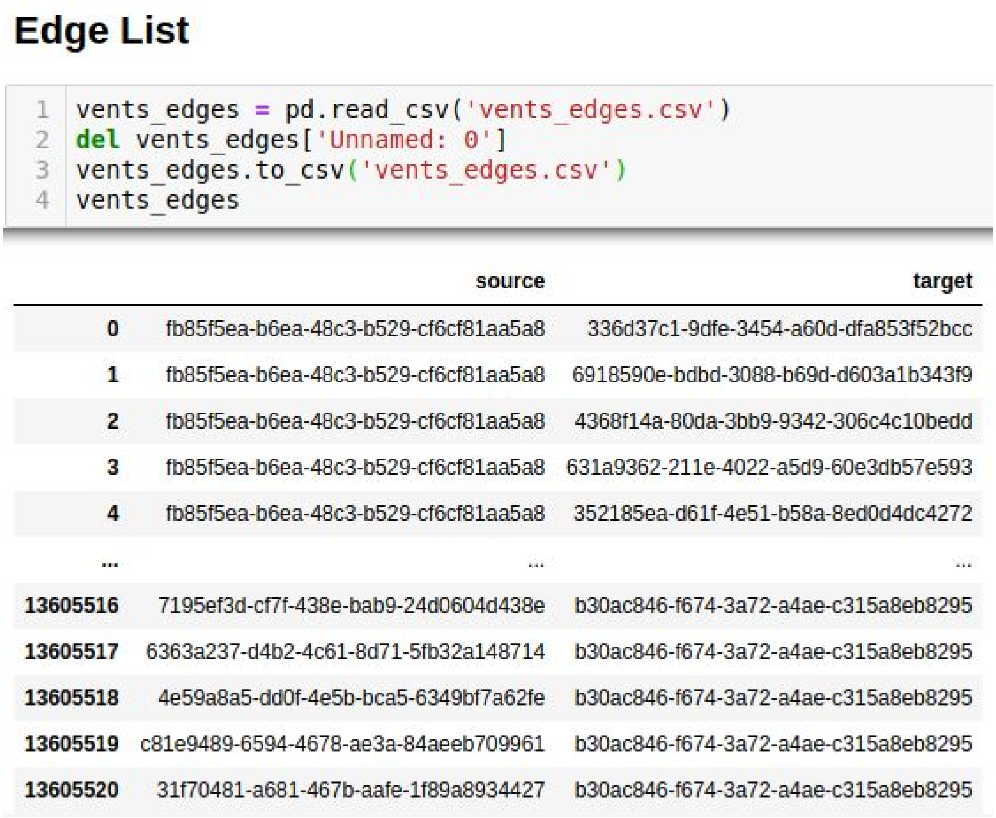
Vents Metadata

```
vents_metadata = pd.read_csv('vents_metadata.csv')
vents_metadata
```

	emotion_id	user_id	created_at	reactions
0	a992d9f0-1b4c-4a6c-9d73-4cd7deb287ef	1a62fe90-d702-3051-b5fe-0a9c86adac56	2014-11-15 12:01:22.000	0
1	fe1ac197-3294-493f-ba9d-04c6bfbea10c	f05733e9-078c-3413-848d-a30a7f502ee9	2015-04-24 07:32:16.006	6
2	3180a95c-c03d-4a36-b78c-26d54d928049	336d37c1-9dfe-3454-a60d-dfa853f52bcc	2014-12-29 04:03:26.000	0
3	d84f9579-ba96-4818-93a5-7ff12f504098	f281696f-5be8-4b4c-bc44-056ebd6f4157	2018-05-11 07:20:36.284	1
4	abc354fa-7778-490b-81e9-01c6e7f776a0	f281696f-5be8-4b4c-bc44-056ebd6f4157	2018-04-20 20:26:44.275	999
...	...	...	...	...
33623409	e0babb52-73cd-4e31-b83c-357983cc7b46	589d4f90-fc93-42a4-a605-eff0c3f2aa0f	2015-12-18 00:46:11.398	11
33623410	035dcb03-106f-46e8-ab84-fe3cc94c149f	589d4f90-fc93-42a4-a605-eff0c3f2aa0f	2015-12-17 18:56:23.070	3
33623411	1acf814a-c553-4c31-879a-f43248ba70b0	589d4f90-fc93-42a4-a605-eff0c3f2aa0f	2015-12-17 18:30:49.171	3
33623412	0ddaa041-d3c3-4c2e-8a52-89d5d985e6a7	589d4f90-fc93-42a4-a605-eff0c3f2aa0f	2015-12-17 11:53:07.237	11
33623413	6825015b-7cc7-43d9-be03-06553df321c1	589d4f90-fc93-42a4-a605-eff0c3f2aa0f	2015-12-17 02:40:28.145	14

33623414 rows × 4 columns

- In order to work with the edge list dataset it was necessary to convert the edges file into CSV file, then separate the connections between the edges by a comma instead of whitespace. I added a title column for source and target.



Edge List

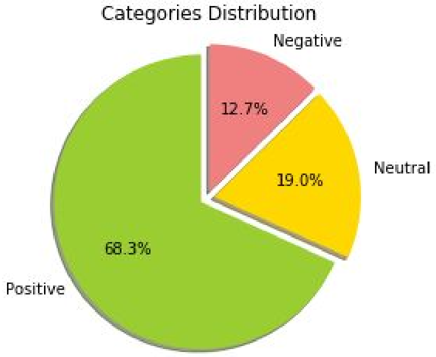
```
vents_edges = pd.read_csv('vents_edges.csv')
del vents_edges['Unnamed: 0']
vents_edges.to_csv('vents_edges.csv')
vents_edges
```

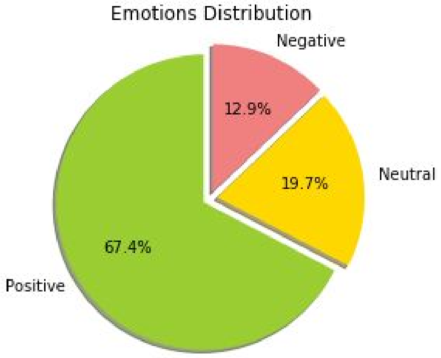
	source	target
0	fb85f5ea-b6ea-48c3-b529-cf6cf81aa5a8	336d37c1-9dfe-3454-a60d-dfa853f52bcc
1	fb85f5ea-b6ea-48c3-b529-cf6cf81aa5a8	6918590e-bdbd-3088-b69d-d603a1b343f9
2	fb85f5ea-b6ea-48c3-b529-cf6cf81aa5a8	4368f14a-80da-3bb9-9342-306c4c10bedd
3	fb85f5ea-b6ea-48c3-b529-cf6cf81aa5a8	631a9362-211e-4022-a5d9-60e3db57e593
4	fb85f5ea-b6ea-48c3-b529-cf6cf81aa5a8	352185ea-d61f-4e51-b58a-8ed0d4dc4272
...	...	...
13605516	7195ef3d-cf7f-438e-bab9-24d0604d438e	b30ac846-f674-3a72-a4ae-c315a8eb8295
13605517	6363a237-d4b2-4c61-8d71-5fb32a148714	b30ac846-f674-3a72-a4ae-c315a8eb8295
13605518	4e59a8a5-dd0f-4e5b-bca5-6349bf7a62fe	b30ac846-f674-3a72-a4ae-c315a8eb8295
13605519	c81e9489-6594-4678-ae3a-84aeeb709961	b30ac846-f674-3a72-a4ae-c315a8eb8295
13605520	31f70481-a681-467b-aafe-1f89a8934427	b30ac846-f674-3a72-a4ae-c315a8eb8295

After understanding the data structure i thought to label each sentiment category into different category sentiments types: positive, negative and neutral. This allows me to map the different sentiments also to these different sentiments types.

The following pie charts explain the properties of the data.

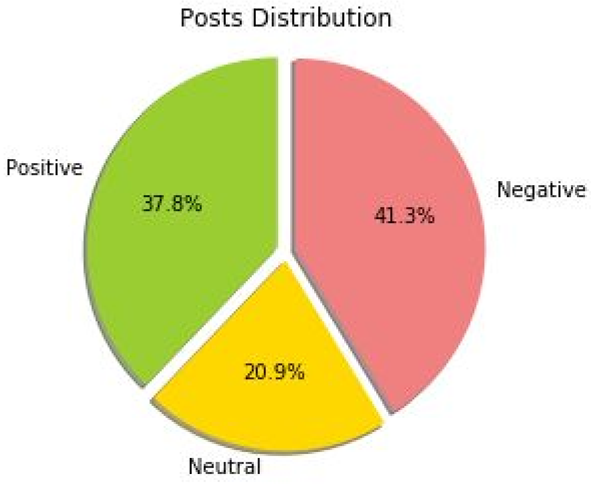
The given emotions distribution by the Vent App:





the given categories and emotions, the sentiment types are distributing pretty the same way. A little bit more than $2/3$ of the emotions and categories are positive, almost $1/5$ are neutral sentiments and the rest are negative sentiments.

Emotions distribution of users posts:



In contrary to the distribution offered by the Vent App, user posts distribution are completely different.

Overall, most posts that are posted are either positive or negative, but slightly more posts with negative emotion (about $4/5$), but only about $1/5$ of the posts are of a neutral emotion.

Thinking:

If i can map sentiments into categories, according to metadata dataset i can create a little profiling to the users according to their posts. I calculated the mean sentiment score to each user, using grouping by and aggregation. By giving Threshold i can define if a user is generally:

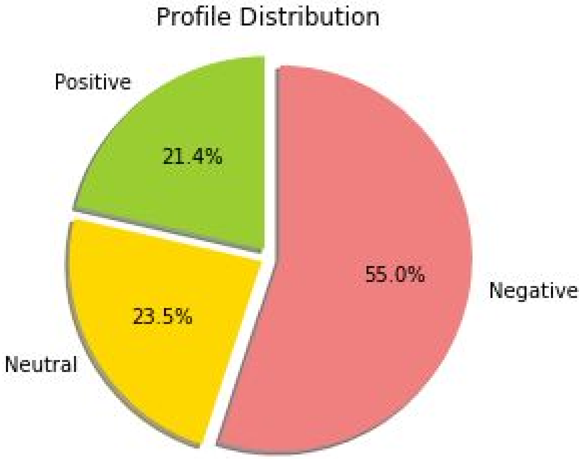
positive: x > 0.2
neutral: -0.2 < x < 0.2
negative: x < -0.2

Furthermore, for the given reactions for user posts i created parameter that determines the user popularity by his posts. Thus it determines if a user has a low impact, soft impact, high impact or he is a “top user”:

Top user: x > 330
high: 110 < x < 330
Neutral: 15 < x < 110
Low: x < 15

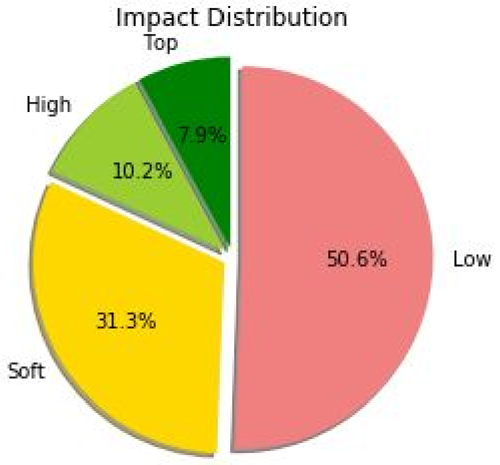
the numbers were chosen according to:
Mean(x) = 220
Median(x) = 15

General user profiling according to their posts:



More than half of the users are determined as negative users. In one hand surprising because of the emotions distribution that are given by the application and the actual situation, but in the other hand we have already seen in the last pie chart that people are posting more negative posts rather than positive ones despite the numerical closeness between them (~40% each).

General user’s post popularity / impact:



About half of the users have low impact / posts popularity. Almost 1/5 of the users have big impact by their posts.

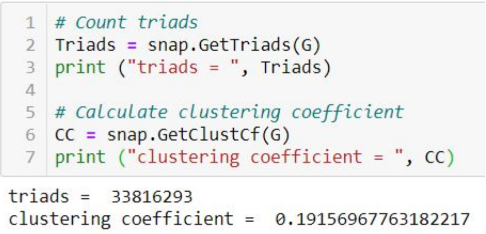
In order to analyze the graph with SNAP, i had to convert the strings of the user_id in the nodes dataset into ‘int’ types by giving them the index number as according to their id.

This allowed me to map the users id in the edges dataset into the corresponding numbers in the new nodes id’s.

Graph Analysis using SNAP:

Before removing low impact and low degree nodes i am dealing with. 700,052 nodes and 13.6M edges.

The number of triads and clustering coefficient:

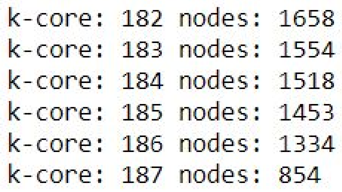


```
# Count triads
Triads = snap.GetTriads(G)
print ("triads = ", Triads)

# Calculate clustering coefficient
CC = snap.GetClustCf(G)
print ("clustering coefficient = ", CC)
```

```
triads =  33816293
clustering coefficient =  0.19156967763182217
```

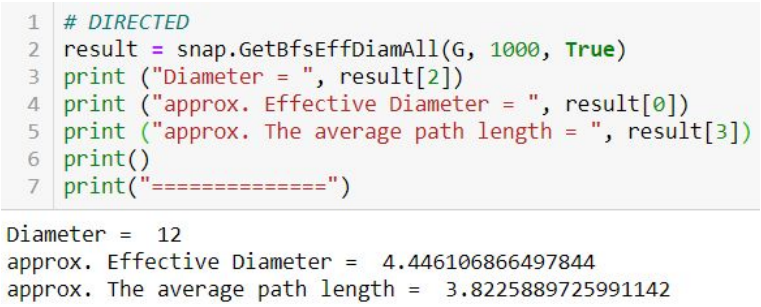
187 k-cores for the innermost k-shell calculation:



```
k-core: 182 nodes: 1658
k-core: 183 nodes: 1554
k-core: 184 nodes: 1518
k-core: 185 nodes: 1453
k-core: 186 nodes: 1334
k-core: 187 nodes: 854
```

Average path length, diameter and effective diameter:

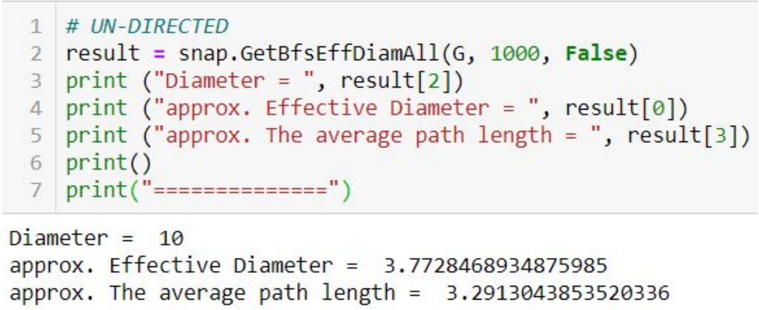
Directed:



```
# DIRECTED
result = snap.GetBfsEffDiamAll(G, 1000, True)
print ("Diameter = ", result[2])
print ("approx. Effective Diameter = ", result[0])
print ("approx. The average path length = ", result[3])
print()
print("==============")
```

```
Diameter =  12
approx. Effective Diameter =  4.446106866497844
approx. The average path length =  3.8225889725991142
```

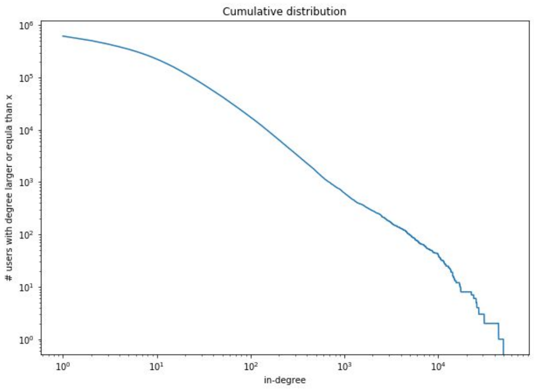
Undirected:

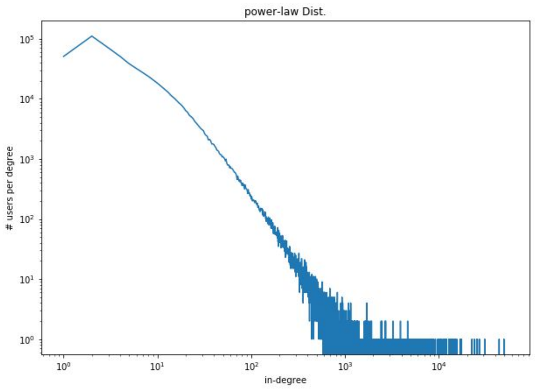


```
# UN-DIRECTED
result = snap.GetBfsEffDiamAll(G, 1000, False)
print ("Diameter = ", result[2])
print ("approx. Effective Diameter = ", result[0])
print ("approx. The average path length = ", result[3])
print()
print("==============")
```

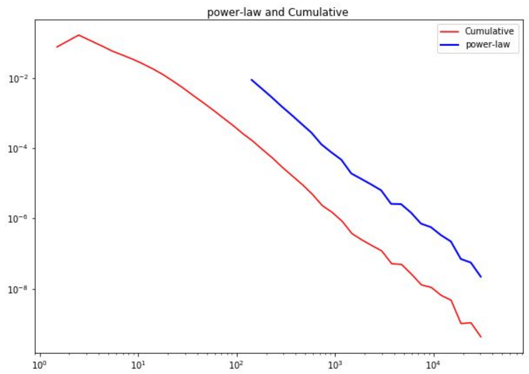
```
Diameter =  10
approx. Effective Diameter =  3.7728468934875985
approx. The average path length =  3.2913043853520336
```

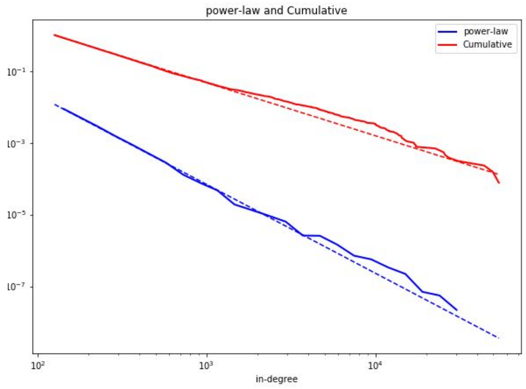
Power-law and cumulative distributions:





Fitting Cumulative and and Power-law Distributions:





Next Steps:

Large graph is not easy and very efficient for analysis, I have to remove data nodes that have very low affect - by conclusion from the given metadata and by low degree nodes removal.

- After removing Low degree nodes (that are low for both in-degree and out-degree together) , i am dealing with 83,035 nodes and 5.6M edges. Now, new calculation for closeness metrics, Comparing the network with similar networks types like Erdos-Renyi, Preferential attachment and more.

Summarize and insert it into the report (I did not include the summarizes for the full graph - before removing low degree nodes although i calculated them)

- Import the spreadsheets of nodes and edges into Gephi App and start making deeper analysis and visualization for the graph: Community detection and visualization, analysis of the impact of how the different parameters i have created until now is affecting and presenting for different communities.