Load Everything Here

[ ] ↳ *3 cells hidden*

Read Data

```
df = pd.read_csv("Edgelist2020_2.csv")
```

```
df
```

	Source	Target	Weight
0	e1122	e6601	0.1273
1	e1122	e7800	0.1195
2	e1122	e785	0.2042
3	e1122	e8770	0.0730
4	e1122	i120	0.2158
...	...	...	...
30100	t2026xa	t2024xa	0.4846
30101	t2027xa	t20211a	0.4361
30102	t22231a	t22232a	0.5137
30103	s92324a	s92334a	0.8462
30104	i69111	i69110	0.6295

30105 rows × 3 columns

Convert to Graph and Visualize

graph conversion & info

```
%%time
graph=nx.convert_matrix.from_pandas_edgelist(df,source='Source', target='Target', edge_attr=N
graph.name = "Covid DisNet for Edgelist2019_2"
print(nx.info(graph))
print("-----------------------------------")
```

```
Name: Covid DisNet for Edgelist2019_2
Type: Graph
Number of nodes: 2101
Number of edges: 30105
Average degree:  28.6578
-----------------------------------
CPU times: user 65.1 ms, sys: 0 ns, total: 65.1 ms
Wall time: 68.1 ms
```

```
degree_centrality = nx.algorithms.centrality.degree_centrality(graph)
first10pairs = {k: degree_centrality[k] for k in sorted(degree_centrality.keys())[:10]}
first10pairs
```

```
{'a0472': 0.029523809523809525,
 'a0839': 0.0009523809523809524,
 'a403': 0.0004761904761904762,
 'a4101': 0.0014285714285714286,
 'a4102': 0.0009523809523809524,
 'a414': 0.0004761904761904762,
 'a4151': 0.007142857142857143,
 'a4152': 0.0004761904761904762,
 'a4159': 0.002857142857142857,
 'a4189': 0.047142857142857146}
```

```
eigenvector_centrality = nx.algorithms.centrality.eigenvector_centrality_numpy(graph)
first10pairs = {k: eigenvector_centrality[k] for k in sorted(eigenvector_centrality.keys())[:
first10pairs
```

```
{'a0472': 0.028865883398139978,
 'a0839': 0.0009338308525216134,
 'a403': -1.910580976727113e-18,
 'a4101': 0.0003981326292866526,
 'a4102': 1.0572806113661003e-05,
 'a414': 0.0001683760786564781 4,
 'a4151': 0.005496076889414611,
 'a4152': 1.1188211825370778e-05,
 'a4159': 0.0015348730393305993,
 'a4189': 0.039779899864510776}
```

```
katz_centrality = nx.algorithms.centrality.katz_centrality_numpy(graph)
first10pairs = {k: katz_centrality[k] for k in sorted(katz_centrality.keys())[:10]}
first10pairs
```

```
{'a0472': -0.05597927468417884,
 'a0839': 0.012955270247768406,
 'a403': 0.0016005343887325531,
 'a4101': 0.0011944416814566 32,
 'a4102': 0.003237604975812472,
 'a414': -0.004157446518558565,
 'a4151': 0.019922031952683765,
 'a4152': 0.003398487389935315,
```

```
 'a4159': -0.00495440684006463,
 'a4189': 0.05672733739024446}
```

```
number_of_triangles = sum(nx.triangles(graph).values()) / 3
number_of_triangles
```

```
842044.0
```

```
nx.algorithms.cluster.transitivity(graph)
```

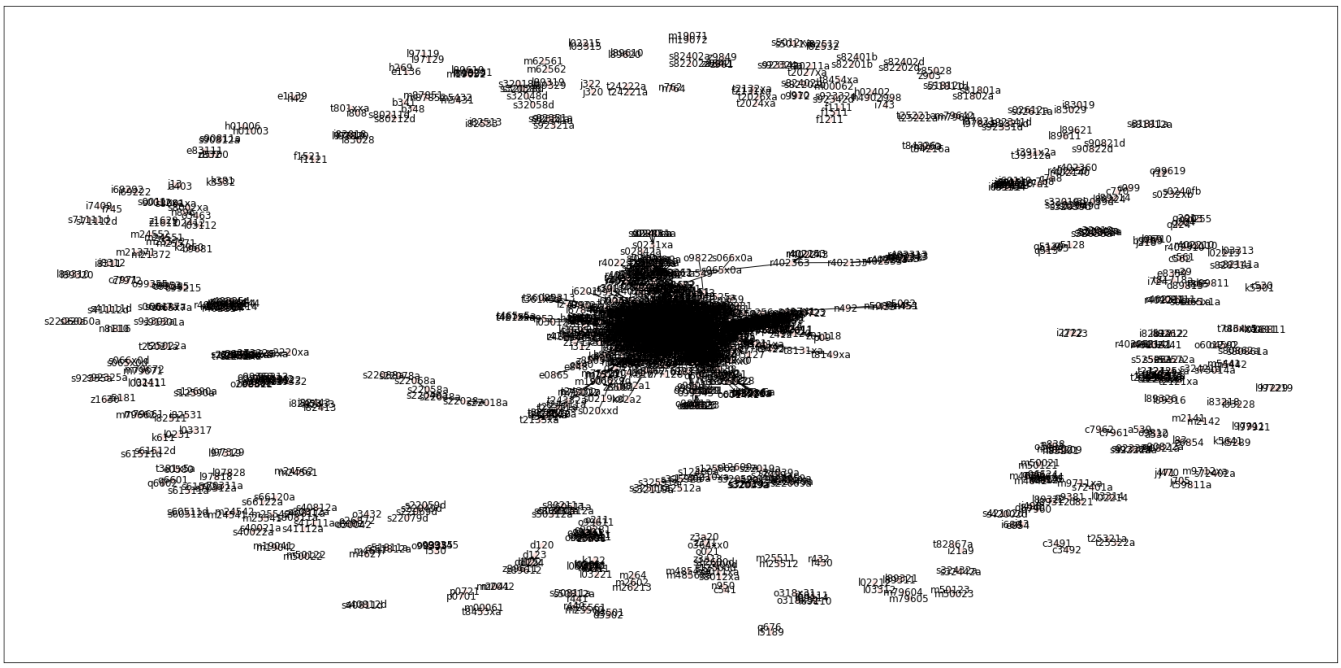
```
0.4906473059509728
```

```
print(nx.average_clustering(graph))
```

```
0.478484940043083
```

whole graph plot

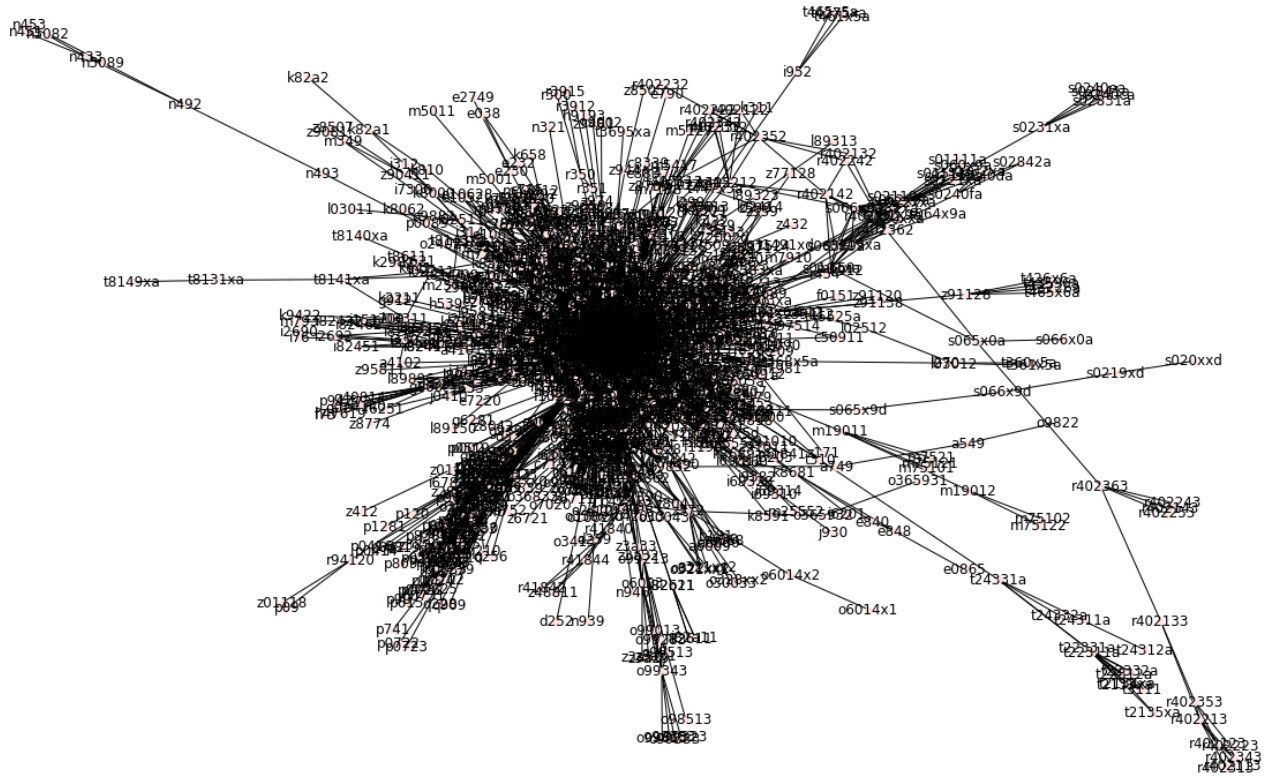
```
%%time
nx.draw_networkx(graph,
                 #pos,
                 with_labels=True,
                 node_size=30,
                 node_color="mistyrose",
                 #edgelist=edges,
                 #edge_color=weights,
                 edge_cmap=plt.cm.Accent,
                 style="solid",
                 width=1)
nx.draw_networkx(graph.subgraph('z20828'),  font_size=16,node_size=120, node_color='red')
plt.subplots_adjust(left=1, bottom=3.2, right=4.8, top=6)
plt.show()

print("----------------------------------------")
print("Density:",nx.classes.function.density(graph))
print("----------------------------------------")
```



```
----------------------------------------
Density: 0.013646562861222547
----------------------------------------
CPU times: user 29.7 s, sys: 551 ms, total: 30.2 s
Wall time: 30 s
```

partial graph plot

```
%%time
plt.figure(figsize=(16, 10))
gcc = max(nx.connected_components(graph), key=lambda x: len(x))
H = graph.subgraph(gcc)
nx.draw(H, node_size=30, node_color='mistyrose',with_labels=True,edge_cmap=plt.cm.Accent,styl
plt.subplots_adjust(left=1, bottom=3.2, right=4.8, top=6)
plt.show()
print("Density:",nx.classes.function.density(H))
print("----------------------------------------")
```

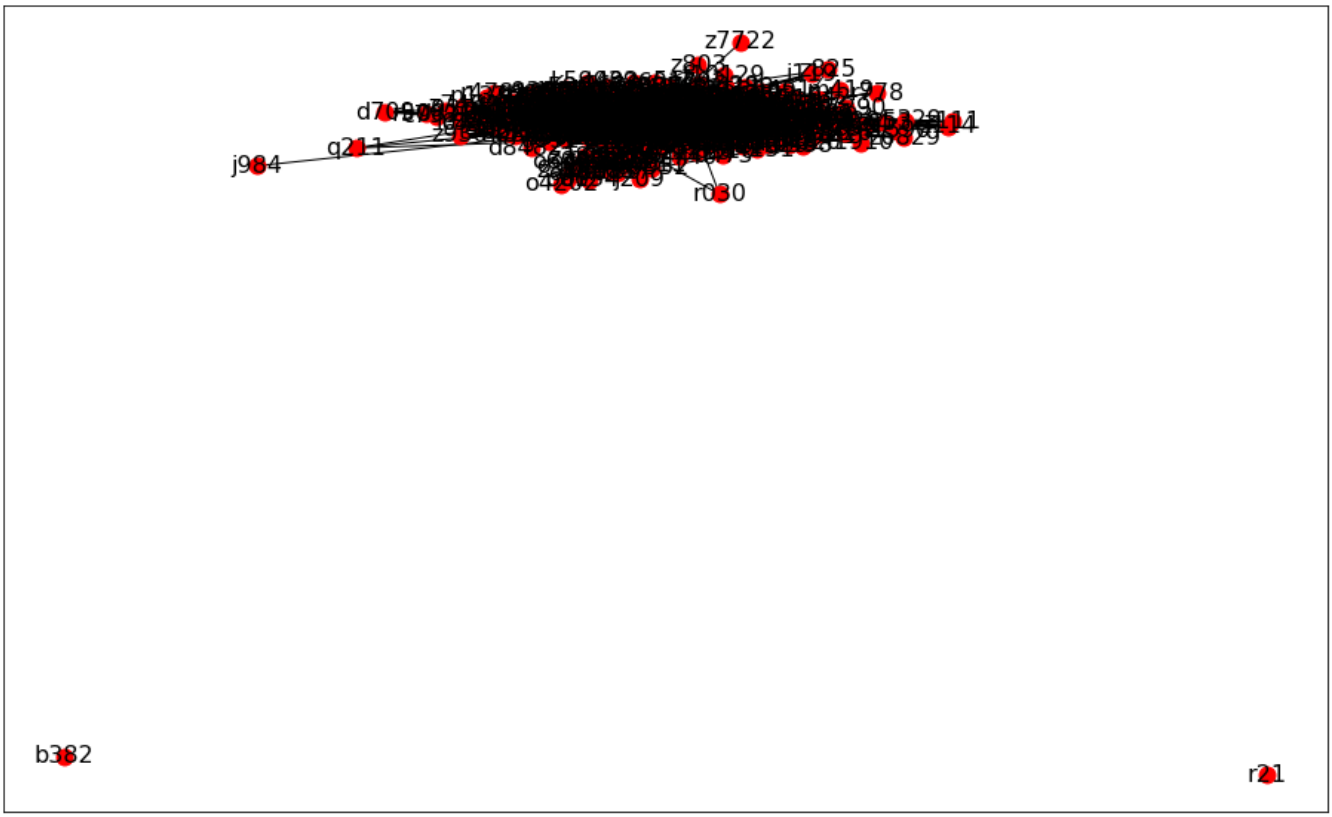


```
Density: 0.025556448085121437
----------------------------------------
CPU times: user 19.9 s, sys: 376 ms, total: 20.3 s
Wall time: 20.4 s
```

plot for z20828's neighbors

```
%%time
plt.figure(figsize=(16, 10))
Sub = nx.classes.function.induced_subgraph(graph,set(graph.neighbors(n="z20828")))
nx.draw_networkx(Sub,  font_size=16,node_size=120, node_color='red')
print("----------------------------------------")
print("Density:",nx.classes.function.density(Sub))
print("----------------------------------------")
```

```
----------------------------------------
Density: 0.18554306773879142
----------------------------------------
CPU times: user 4.82 s, sys: 133 ms, total: 4.95 s
Wall time: 4.88 s
```



Fit node2vec

```
vector_size = round(df.shape[0]**0.25)
vector_size
```

```
13
```

```
%%time
setup = Node2Vec(graph,dimensions=vector_size, walk_length=5, num_walks=5)
model = setup.fit(window=10, min_count=1)
print("----------------------------------------")
```

```
Computing transition probabilities: 100%  2101/2101 [00:48<00:00, 43.02it/s]
Generating walks (CPU: 1):   0%|          | 0/5 [00:00<?, ?it/s]
```

```
%%time
#vocab, vectors = model.wv.key_to_index, model.wv.get_normed_vectors()
vocab, vectors = model.wv.vocab, model.wv.vectors

# get node name and embedding vector index.
name_index = np.array([(v[0], v[1].index) for v in vocab.items()]) #.index

# init dataframe using embedding vectors and set index as node name
node2vec_output = pd.DataFrame(vectors[name_index[:,1].astype(int)])
node2vec_output.index = name_index[:,0]
```

```
CPU times: user 11.6 ms, sys: 0 ns, total: 11.6 ms
Wall time: 23.3 ms
```

```
node2vec_output.shape
```

```
(2101, 13)
```

```
model.wv.most_similar("z20828",topn=10)
```

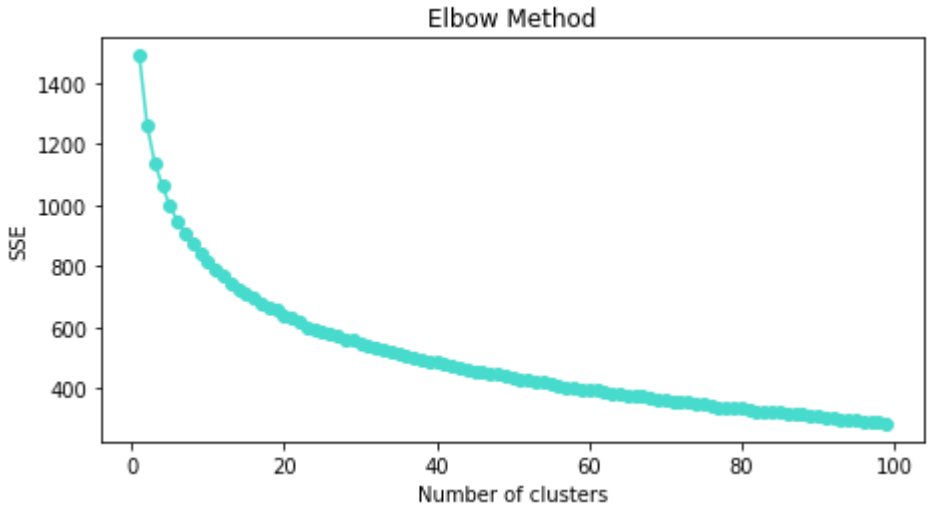
```
[('f419', 0.9946932196617126),
 ('d649', 0.9946731925010681),
 ('f329', 0.9936752319335938),
 ('e6601', 0.9925082921981812),
 ('k219', 0.991245448589325),
 ('j45909', 0.9903380274772644),
 ('z8619', 0.9902105331420898),
 ('d62', 0.9889881014823914),
 ('z79890', 0.9886508584022522),
 ('f17210', 0.9886504411697388)]
```

K-means

Find k

```
%%time
SSE = []
for i in range(1,100):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=100, n_init=50, random_state=42)
    kmeans.fit(node2vec_output)
    SSE.append(kmeans.inertia_)
plt.plot(range(1,100), SSE,"o-",color="#47DBCD")
plt.title('Elbow Method')
```

```
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.subplots_adjust(left=0.25, bottom=0.8, right=1.2, top=1.5)
plt.show()
```




```
CPU times: user 7min 42s, sys: 6min 4s, total: 13min 47s
Wall time: 7min 19s
```

plot k-means clustering

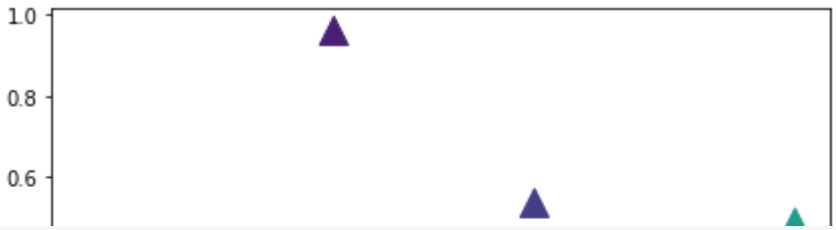
```
n_clusters=kmeans.n_iter_
```

```
kmeans = KMeans(n_clusters=n_clusters, init='k-means++', max_iter=1000, n_init=50, random_sta
```

```
kmeans.fit(node2vec_output)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=1000,
       n_clusters=12, n_init=50, n_jobs=None, precompute_distances='auto',
       random_state=42, tol=0.0001, verbose=0)
```

```
t = np.arange(n_clusters)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=200, c=t,marker="^"
plt.subplots_adjust(left=0.1, bottom=0.1, right=1, top=1)
```




```
subsample=[]
for i in range(kmeans.n_clusters):
  temp = []
  temp=node2vec_output.iloc[kmeans.labels_==i,:]
  subsample.append(temp)
```

```
for list in range(len(subsample)):
  print("Group",list+1)
  print(subsample[list])
  print("------------------------------------------------------------")
```

```
z1611   -0.657927  0.335314  0.615824  ... -0.768926  0.226273  0.473270
z1629   -0.680178  0.371016  0.660271  ... -0.798092  0.244186  0.397708
m25561  -0.352309  0.497809  0.153035  ... -0.340979  0.086676  0.411492
m25562  -0.301636  0.526148  0.055131  ... -0.377859  0.057545  0.373527
m21372  -0.151492  0.375291  0.156460  ... -0.313596 -0.092707  0.730611
m21371  -0.147567  0.339942  0.091610  ... -0.361055 -0.141015  0.749506

[33 rows x 13 columns]
------------------------------------------------------------
Group 8
               0         1         2   ...        10        11        12
s0011xa -1.013764  0.685489 -0.086503  ... -0.764507 -0.674419  0.126911
s0012xa -1.019742  0.627846 -0.019721  ... -0.783056 -0.667281  0.208953
s82141a -0.484977  0.338560  0.529170  ... -0.281430 -0.537410  0.805524
s82831a -0.424556  0.312874  0.523689  ... -0.202763 -0.533579  0.854931
n83202  -0.876571 -0.068047  0.163667  ...  0.030650 -0.683035  0.761253
...           ...       ...       ...  ...       ...       ...       ...
i69292  -0.147071  0.301810  0.508120  ...  0.078250 -1.001883  0.942099
i69222  -0.161404  0.300126  0.466898  ...  0.055880 -0.998618  0.981559
i745    -0.479125  0.170156  0.189046  ... -0.397458 -0.794461  0.829912
i7409   -0.532965  0.134690  0.230104  ... -0.472191 -0.828049  0.813779
t39312a -0.569704  0.552964 -0.198766  ... -0.350568 -0.049482  0.922010

[64 rows x 13 columns]
------------------------------------------------------------
Group 9
               0         1         2   ...        10        11        12
s01511a -0.885492  0.339746 -0.009207  ... -0.228749 -0.476554  0.796858
s0181xa -1.297098  0.543128 -0.133626  ... -0.156680 -0.695563  0.978219
s066x9a -1.326483  0.462020 -0.306336  ... -0.295730 -0.879427  0.988725
s0219xa -1.198019  0.351787 -0.166882  ... -0.310716 -0.771816  0.891203
s061x9a -0.698272  0.265476  0.008015  ... -0.294276 -0.526893  0.691496
...           ...       ...       ...  ...       ...       ...       ...
t25021a -0.249792  0.374913 -0.126905  ... -0.413710 -0.837315  0.670594
t25022a -0.355171  0.297546 -0.168683  ... -0.456682 -0.864084  0.603763
s92342a -1.221680  0.833799 -0.421676  ... -0.301252 -0.172638  0.411101
s060x9a -0.714151  0.307237 -0.000474  ... -0.251497 -0.383288  0.696927
```

```
s02831a -0.606957  0.089607 -0.013732  ... -0.363785 -0.410376  0.794409

[112 rows x 13 columns]
------------------------------------------------------------
Group 10
                0         1         2   ...        10        11        12
o9902   -0.822836  0.283542  0.631595  ... -0.664011 -0.382419  0.581075
o99324  -0.744239  0.164168  0.408576  ... -0.554765 -0.418773  0.571787
z370    -0.911358  0.110838  0.691422  ... -0.809116 -0.364885  0.573919
z3a39   -0.815677  0.235345  0.656020  ... -0.745229 -0.378250  0.586882
z1630   -0.962160  0.235156 -0.254989  ... -0.983584 -0.197939  0.051471
...           ...       ...       ...  ...       ...       ...       ...
o691xx0 -0.577441  0.194571  0.280170  ... -0.503895 -0.408217  0.623720
z3a34   -0.650539  0.227375  0.210187  ... -0.415623 -0.461947  0.582502
l97819  -0.859248  0.151796  0.278849  ... -0.347177 -0.454856  0.317459
o1414   -0.552185  0.183623  0.230646  ... -0.443369 -0.362959  0.583293
z3a30   -0.659069  0.006310  0.191601  ... -0.281982 -0.344986  0.520800

[116 rows x 13 columns]
------------------------------------------------------------
Group 11
              0         1         2   ...        10        11        12
```

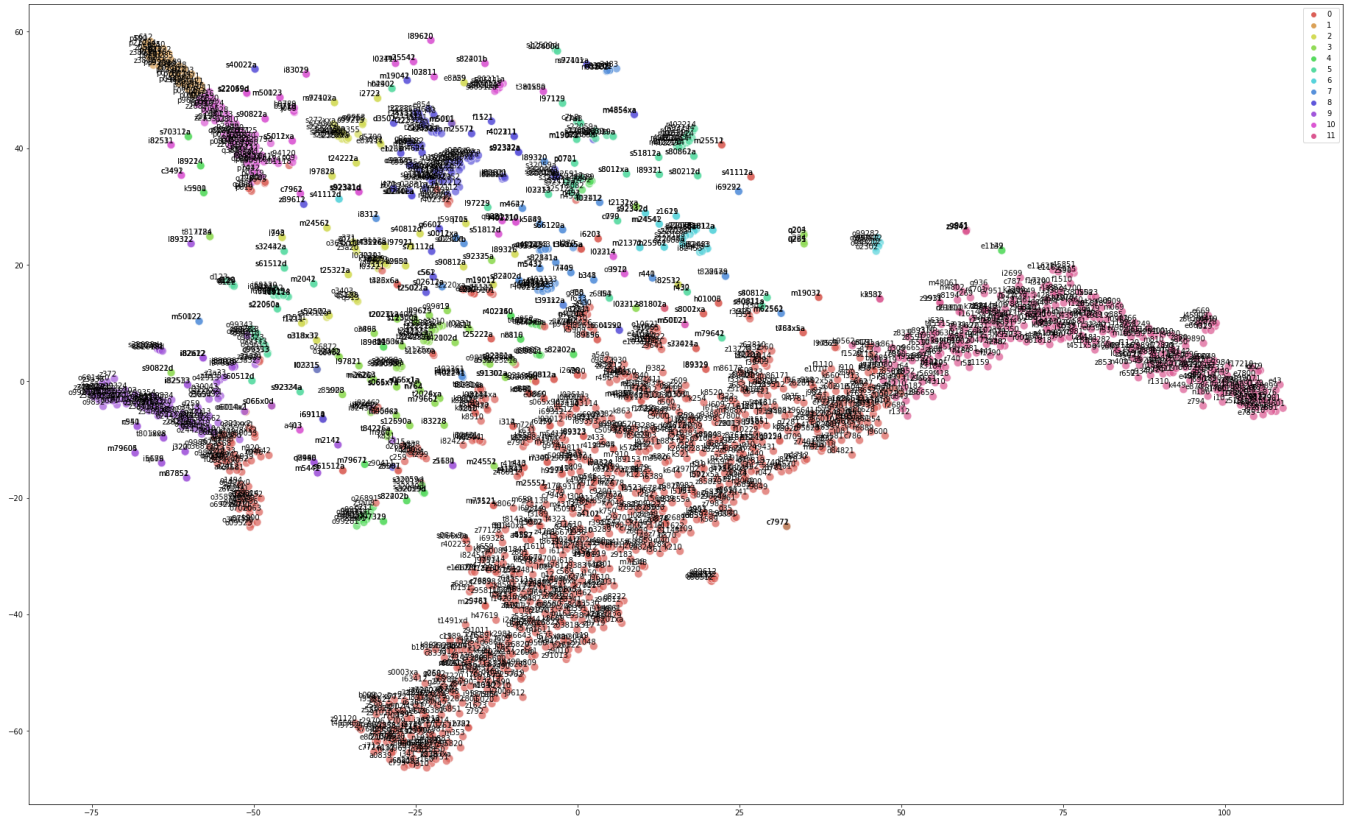
T-SNE

```
def tsne_plot(model):
    "Creates and TSNE model and plots it"
    labels = []
    tokens = []

    for word in model.wv.vocab:
        tokens.append(model[word])
        labels.append(word)

    tsne_model = TSNE(perplexity=30, n_components=2, learning_rate=10, init='random', n_iter=
    new_values = tsne_model.fit_transform(tokens)

    x = []
    y = []
    for value in new_values:
        x.append(value[0])
        y.append(value[1])

    plt.figure(figsize=(32, 20))
    sns.scatterplot(
        x=x, y=y,
        hue= kmeans.labels_,
        palette=sns.color_palette("hls", len(set(kmeans.labels_))),
        legend="full",
        alpha=0.7,
        s=120
```

```
        )
    for i in range(len(x)):

      plt.annotate(labels[i],
                   xy=(x[i], y[i]),
                   xytext=(3, 1),
                   textcoords='offset points',
                   ha='right',
                   va='bottom')


    plt.show()
```

```
%%time
tsne_plot(model)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: DeprecationWarning: Call
  import sys
```



```
CPU times: user 2min 51s, sys: 1.39 s, total: 2min 52s
Wall time: 1min 32s
```

✓ 1m 34s completed at 12:35 AM