Load Everything Here

[ ] ↳ *3 cells hidden*

Read Data

```
df = pd.read_csv("Edgelist2020_1.csv")
```

```
df
```

	Source	Target	Weight
0	e039	e8342	0.0729
1	e039	e871	0.1456
2	e039	e872	0.1122
3	e039	e876	0.1457
4	e039	f17210	0.0793
...	...	...	...
28975	t23341a	t23342a	0.5883
28976	t23341a	t23362a	0.5153
28977	t23342a	t23362a	0.7243
28978	t23342a	t23372a	0.4808
28979	t23362a	t23372a	0.5491

28980 rows × 3 columns

Convert to Graph and Visualize

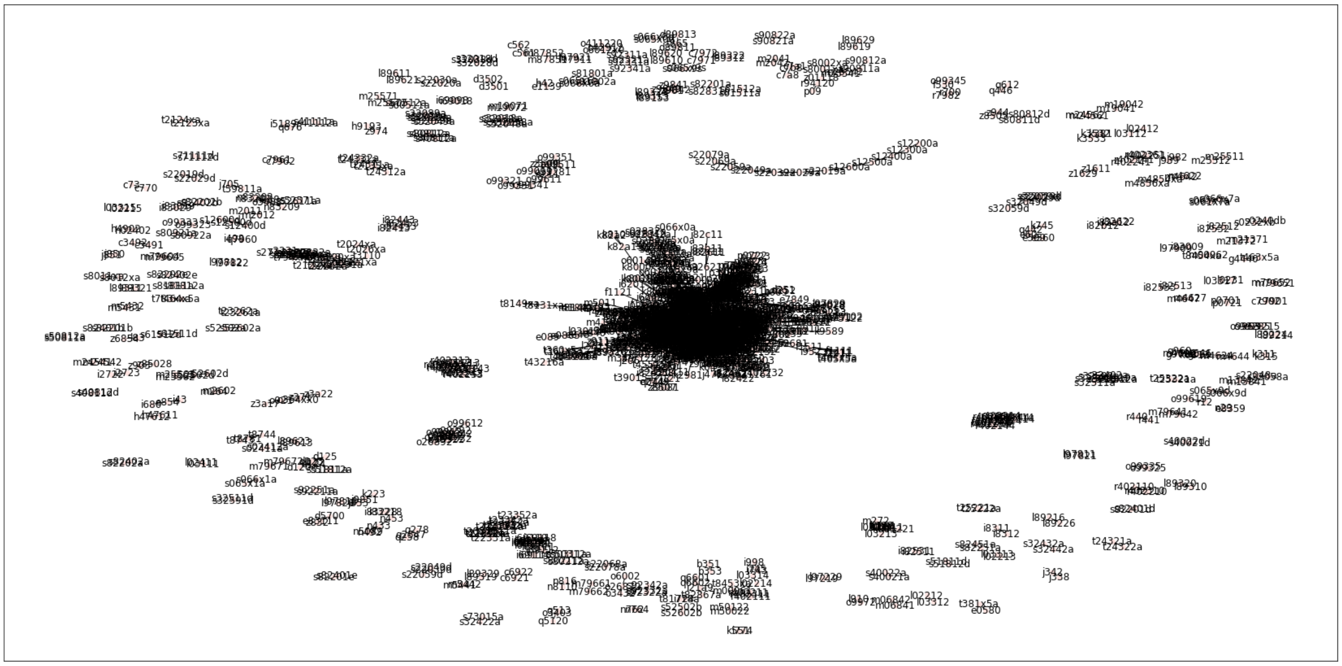
graph conversion & info

```
%%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: 2049
Number of edges: 28980
Average degree:  28.2870
------------------------------------
CPU times: user 72.3 ms, sys: 10.6 ms, total: 82.9 ms
Wall time: 84.4 ms
```

whole graph plot

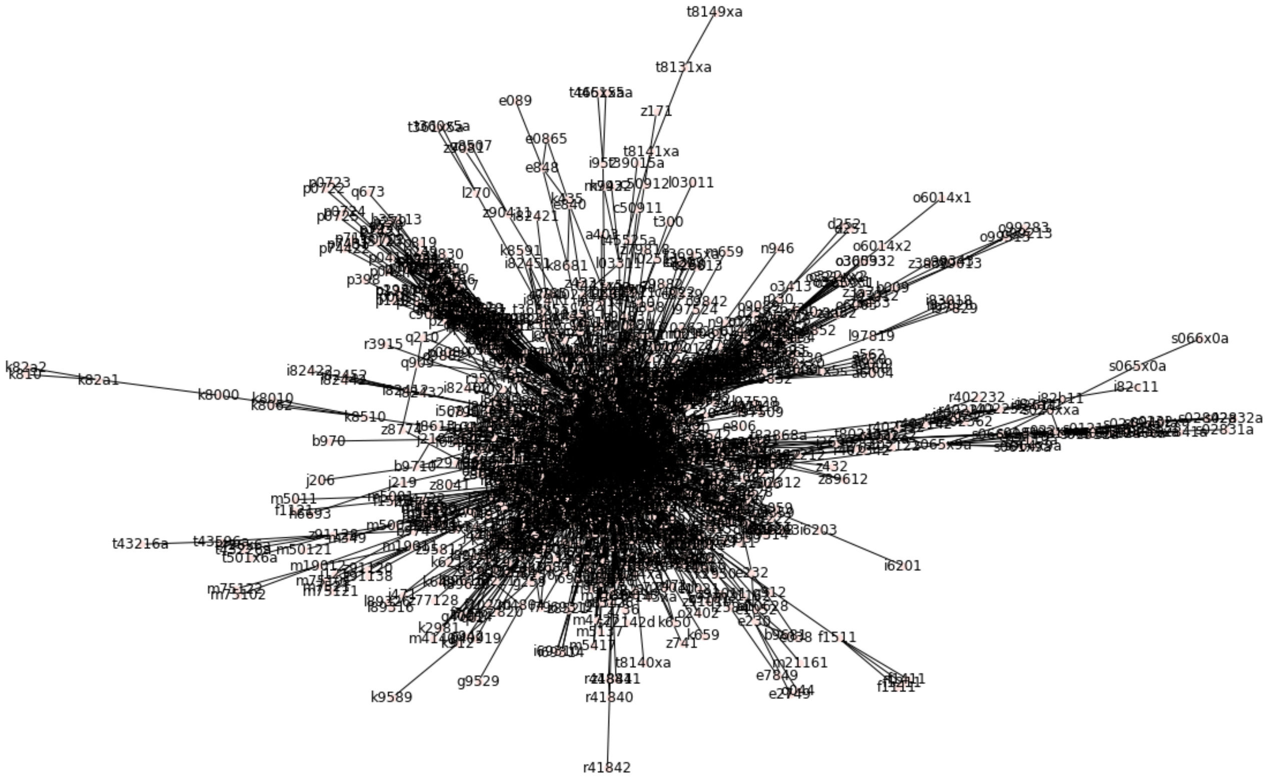
```
%%time
nx.draw_networkx(graph,
                 #pos,
                 with_labels=True,
                 node_size=30,
                 node_color="mistyrose",
                 #edgelist=edges,
                 #edge_color=weights,
                 edge_cmap=plt.cm.Accent,
                 style="solid",
                 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: 0.013811996705710103
----------------------------------------
CPU times: user 28.9 s, sys: 574 ms, total: 29.5 s
Wall time: 29.2 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("----------------------------------------")
```

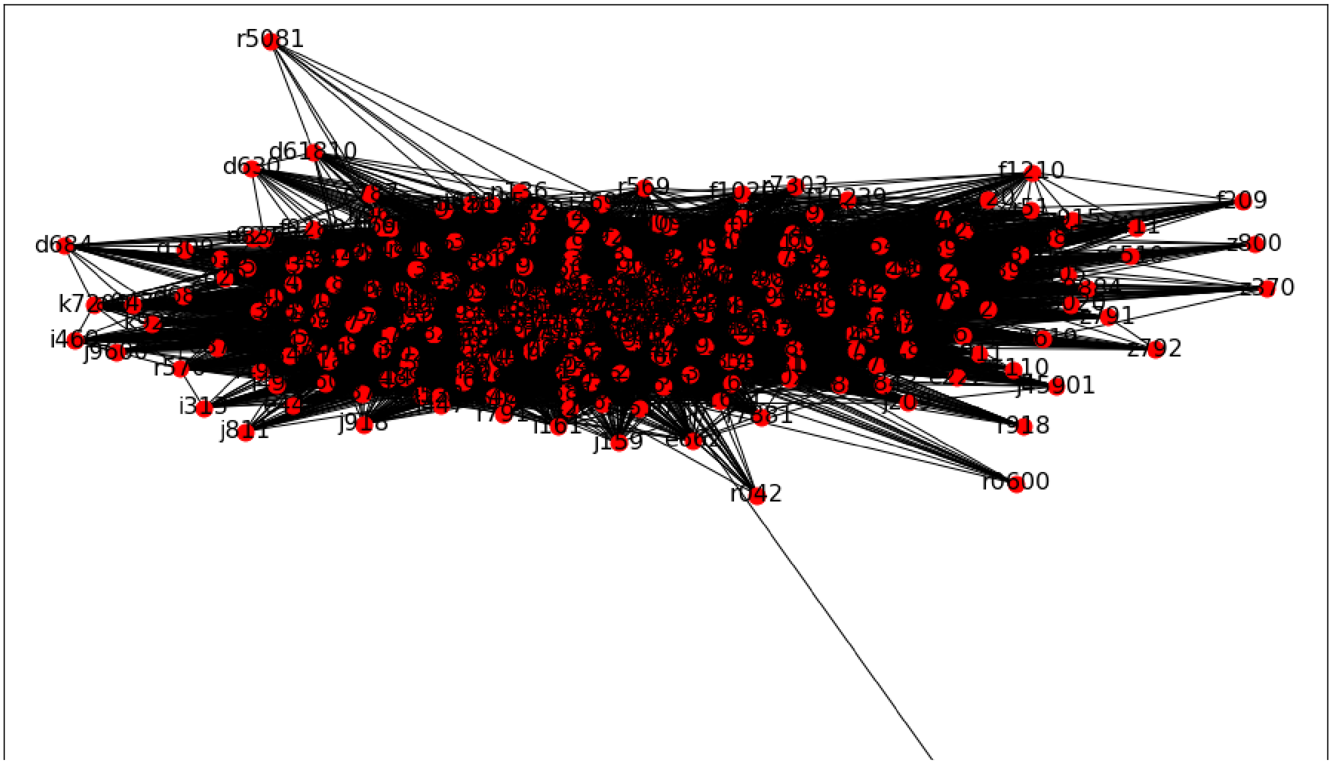


```
Density: 0.026247212373606877
----------------------------------------
CPU times: user 19 s, sys: 404 ms, total: 19.4 s
Wall time: 19.3 s
```

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: 0.46855543347459844
----------------------------------------
CPU times: user 851 ms, sys: 116 ms, total: 967 ms
Wall time: 863 ms
```



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)
print("---------------------------------------")
```

```
Computing transition probabilities: 100%                2049/2049 [00:46<00:00, 44.36it/s]

Generating walks (CPU: 1):   0%|          | 0/5 [00:00<?, ?it/s]
Generating walks (CPU: 1): 100%|██████████| 5/5 [00:02<00:00,  1.92it/s]
---------------------------------------
CPU times: user 49.2 s, sys: 517 ms, total: 49.7 s
Wall time: 50.4 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 7.9 ms, sys: 0 ns, total: 7.9 ms
Wall time: 8.07 ms
```

```
node2vec_output.shape
```

```
(2049, 13)
```

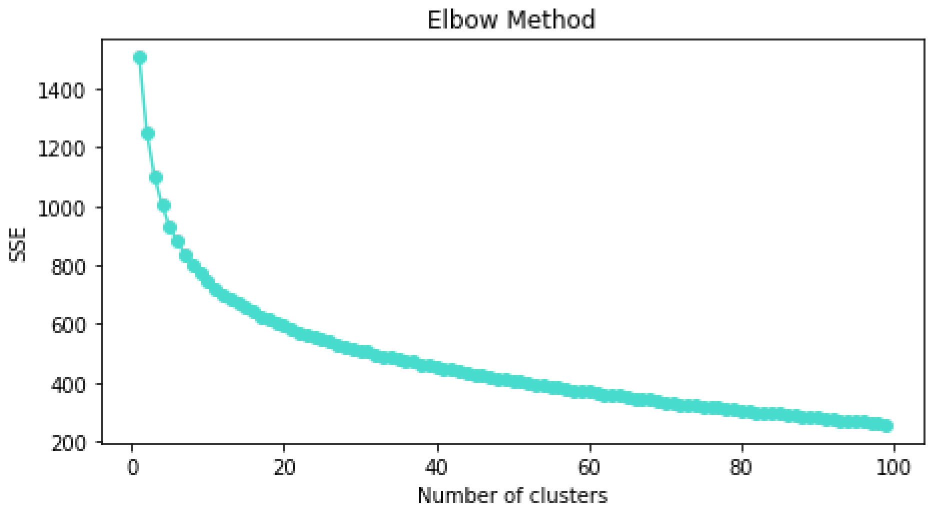
```
model.wv.most_similar("z20828",topn=10)
```

```
[('z86718', 0.9988461136817932),
 ('z79899', 0.9986307621002197),
 ('n189', 0.9984985589981079),
 ('z9049', 0.9982914328575134),
 ('j90', 0.9981628060340881),
 ('z79891', 0.9981063008308411),
 ('e8342', 0.9981051087379456),
 ('z951', 0.9978451728820801),
 ('m1990', 0.9978070855140686),
 ('r531', 0.9977770447731018)]
```

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 35s, sys: 5min 45s, total: 13min 21s
```

plot k-means clustering

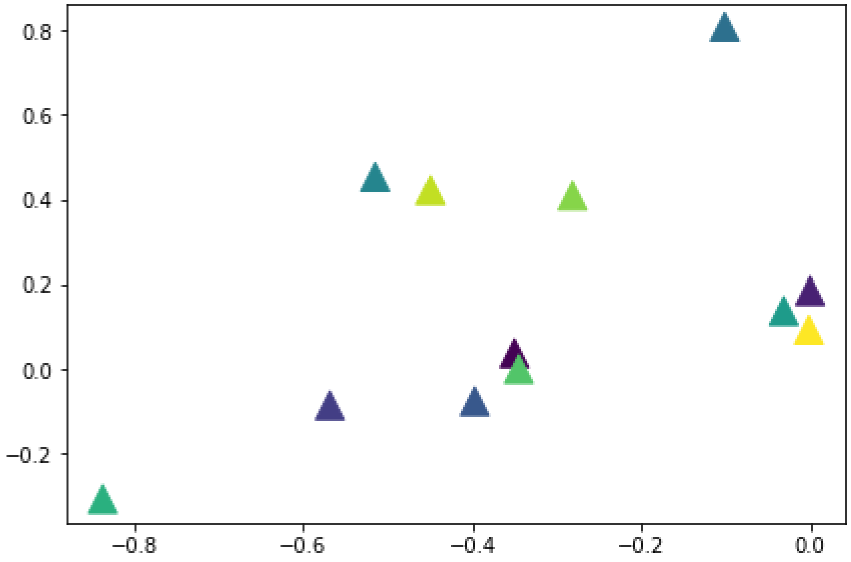
```
n_clusters=kmeans.n_iter_
```

```
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("--------------------------------------------------------------")
```

```
Group 1
              0         1         2  ...        10        11        12
i69311 -0.332997  0.068068  0.438989  ... -0.275655 -0.134233  0.122634
i69392 -0.427167  0.054539  0.409889  ... -0.300064 -0.061988  0.104649
z7682  -0.255241  0.038212  0.300117  ... -0.194074 -0.060573  0.018874
z808   -0.367725  0.014424  0.365180  ... -0.240391 -0.084646  0.026334
i2690  -0.367278  0.178226  0.580148  ... -0.209773 -0.081059  0.041977
...          ...       ...       ...  ...       ...       ...       ...
k435   -0.415025  0.118256  0.437664  ... -0.295965 -0.112512 -0.065598
h532   -0.350877 -0.013724  0.365362  ... -0.273584 -0.031230  0.078145
k56609 -0.437427 -0.056073  0.293641  ... -0.243024 -0.086806  0.040874
m50121 -0.307307  0.031482  0.365888  ... -0.271495 -0.128928 -0.065452
i361   -0.308349 -0.007102  0.265520  ... -0.246609 -0.069961  0.020833

[793 rows x 13 columns]
--------------------------------------------------------------
Group 2
              0         1         2  ...        10        11        12
o99332  0.079230  0.183329  0.592199  ... -0.531545  0.104041 -0.209476
o99322  0.288670  0.172158  0.683489  ... -0.637918  0.089101 -0.288219
o99511 -0.073633  0.169239  1.122136  ... -0.194277  0.155403  0.207611
o99281  0.244532  0.082898  1.846277  ... -0.045132  0.385297  0.505220
z3a09  -0.137971  0.062110  0.776387  ... -0.198451  0.136653  0.199777
...          ...       ...       ...  ...       ...       ...       ...
h4902   0.073927 -0.131014  0.405715  ... -0.758630  0.039351  0.033466
k041   -0.146371 -0.124246  0.945618  ... -0.345320  0.039159 -0.010869
g960    0.377312  0.548437  0.459752  ... -0.462924  0.119093 -0.028205
g9782   0.367270  0.560062  0.444544  ... -0.405018  0.103467  0.007941
o99612  0.016906  0.106560  0.506535  ... -0.468160  0.002641 -0.192293

[88 rows x 13 columns]
--------------------------------------------------------------
Group 3
              0         1         2  ...        10        11        12
r414   -0.462983 -0.021344  0.420166  ... -0.306408 -0.083303  0.146868
i69354 -0.621965 -0.134387  0.302482  ... -0.378521 -0.019878  0.268876
f250   -0.540804 -0.108190  0.365466  ... -0.313353 -0.085353  0.077434
i69351 -0.607631 -0.104943  0.316736  ... -0.345649 -0.002720  0.156841
z803   -0.554210 -0.069415  0.397334  ... -0.374334 -0.007850  0.145013
...          ...       ...       ...  ...       ...       ...       ...
z930   -0.455418  0.034112  0.344418  ... -0.347992 -0.067738  0.143512
k589   -0.502410 -0.016502  0.326418  ... -0.329176 -0.025369  0.103232
j101   -0.444810 -0.038192  0.320931  ... -0.276544 -0.026336  0.111582
r05    -0.575717 -0.127423  0.304440  ... -0.366204 -0.016047  0.193433
```

```
b370   -0.499482 -0.017925  0.367110  ... -0.267554 -0.004573  0.106682

[317 rows x 13 columns]
--------------------------------------------------------------
Group 4
               0         1         2  ...        10        11        12
l89623  -0.868194  0.466029  0.996068  ...  0.073645  0.395050 -0.271373
l89613  -0.853181  0.481907  1.009715  ...  0.061745  0.369307 -0.290757
s50312a -0.460472  0.339707  0.338470  ...  0.099784  0.341267 -0.023384
s50311a -0.546460  0.435088  0.407982  ...  0.152818  0.457629 -0.108874
s80211a -0.529496  0.350362  0.395020  ...  0.137343  0.430098  0.000874
...           ...       ...       ...  ...       ...       ...       ...
s32512a -0.456361 -0.261222  0.774849  ... -0.484270 -0.090393 -0.132639
s32591a -0.560533 -0.242745  0.999955  ... -0.454810 -0.120620 -0.206263
s32511a -0.448762 -0.080948  0.660433  ... -0.323991 -0.090009 -0.138281
```

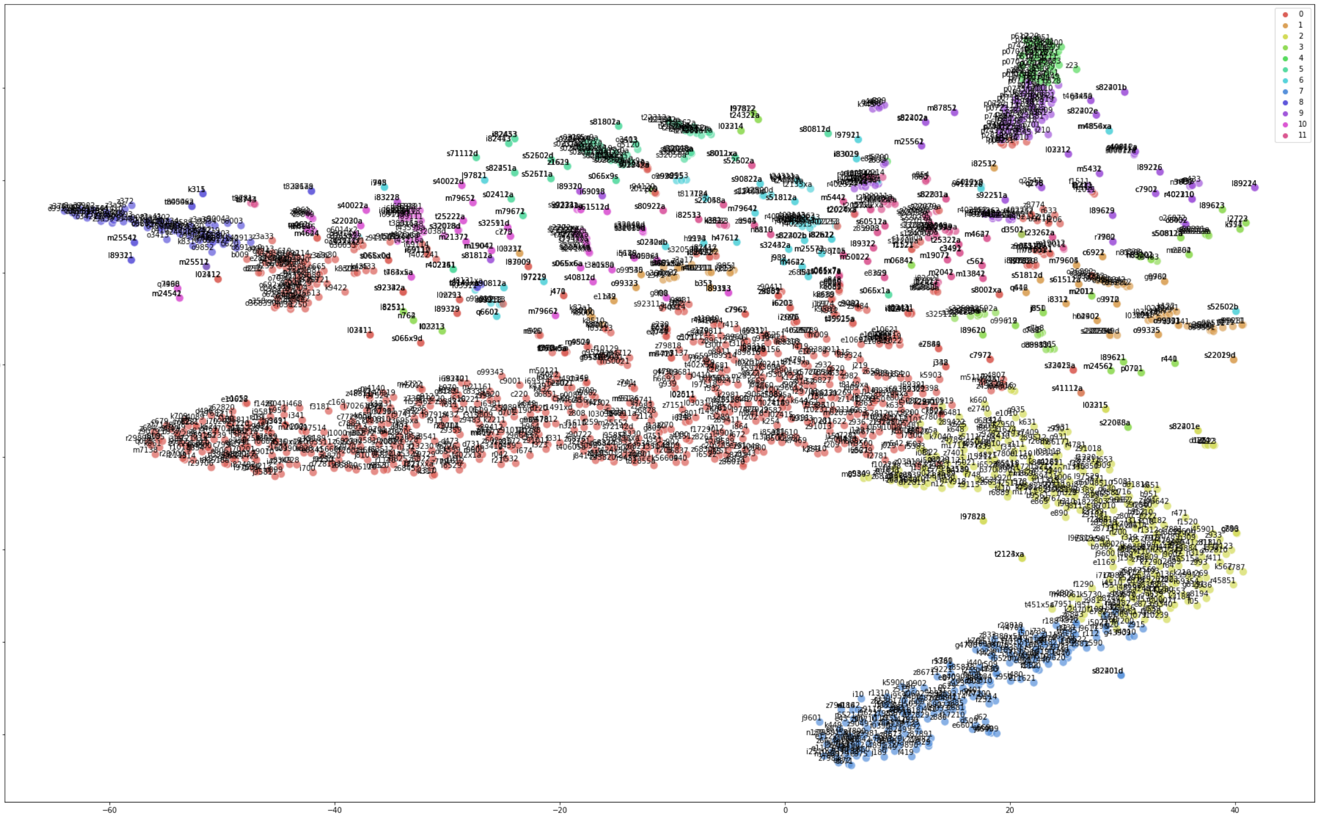
T-SNE

```
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'
```

```
                     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: Cal
  import sys
```



```
CPU times: user 2min 55s, sys: 1.04 s, total: 2min 56s
Wall time: 1min 34s
```

✓ 1m 35s completed at 11:14 PM ● ✕