[CM4]: Part 3: t-SNE

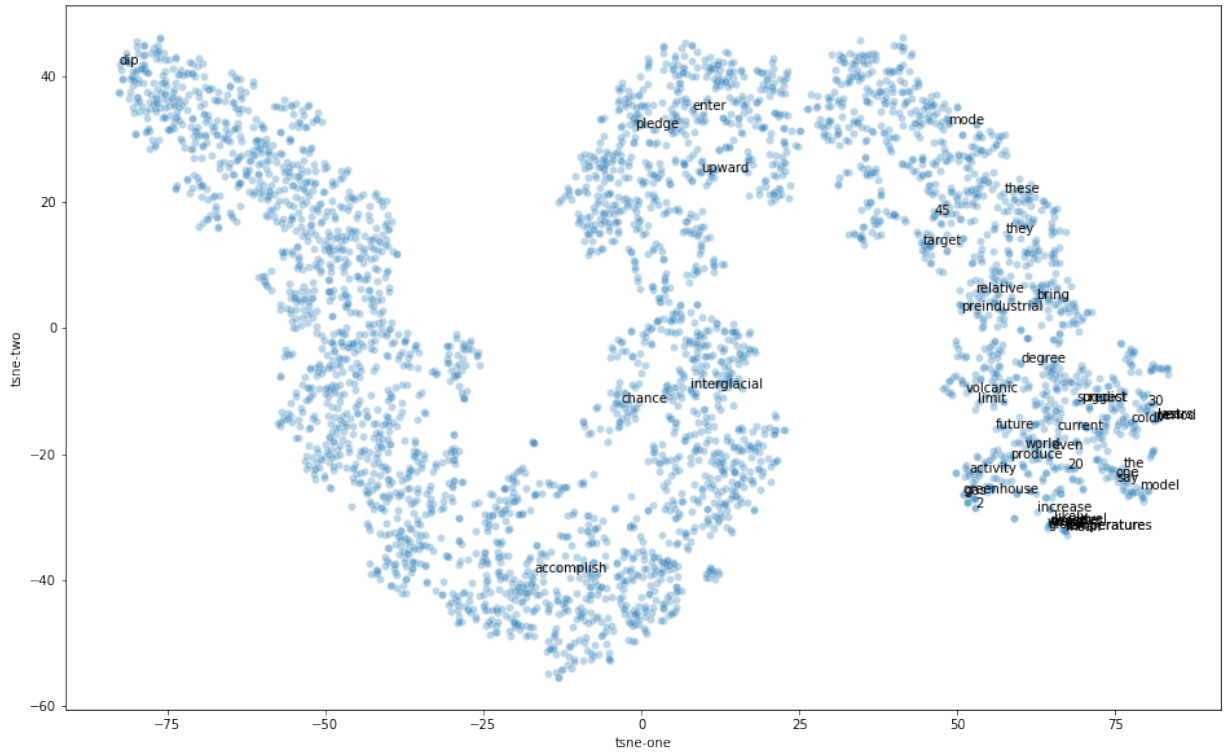
[ ]:
```
from sklearn.manifold import TSNE

# applying t-SNE on word2vec test embedding
tsne = TSNE(n_components=2, init='pca', random_state=0)
tsne_test = tsne.fit_transform(list(test_embedding.values()))
```

t-SNE is primarily used for data exploration and visualizing high-dimensional data in a lower-dimensional space, typically the 2D plane (or maybe 3d). Since the prime purpose of t-SNE is to visualize high-dimensional data, it is best to visualize in 2 dimensions (typically) or max 3 dimensions as more than 3 dimensions will not allow effective visualization.

To visualize high-dimensional data t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance. The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. It then tries to optimize these two similarity measures using a cost function. It tries to balance attention between local and global aspects of the data. Student t-distribution is used to solve the crowding problem caused by the heavy tails of gaussian distribution.

tSNE embedding visualization

[ ]:
```
df = pd.DataFrame()
df['tsne-one'] = tsne_test[:,0]
df['tsne-two'] = tsne_test[:,1]
plt.figure(figsize=(16,10))
sns.scatterplot(
    x="tsne-one", y="tsne-two",
    palette=sns.color_palette("hls", 10),
    data=df,
    legend="full",
    alpha=0.3
)
for i, word in enumerate(list(test_embedding.keys())[:50]):
  plt.annotate(word,xy=(tsne_test[i,0],tsne_test[i,1]))
```



From the t-SNE plot is can be inferred that t-SNE is able to preserve both the local and global aspects of the data. t-SNE tends to expand denser regions of data. It magnifies the small empty spaces. It appears to form clusters of the data.

We annotated few of the words in the subspace. t_SNE is able to preserve semantic relationship of the word embeddings in the lower dimension: - The words 'greenhouse' and 'co2' appear closer in the t_SNE subspace similar to the word2vec embeddings. - The words 'current' and 'future' of similar context appear closer in the plot. - Since 'dip' and 'increase' are in opposite context (generally), we see they appear in opposite ends of the plot.

Visualizing a subset of similar words in the t-SNE subspace

Here we try to take a subset of few important words (as per the context of the dataset) and find 5 most similar words in the word2vec subspace. Applying t-SNE on these word clusters and visualizing them.

[ ]:
```
# subset of words
keys = ['global', 'carbon', 'climate', 'winter', 'polar',␣
 ↪'june','melt','warm','glaciers']

embedding_clusters = []
word_clusters = []

# creating word cluster of the word subset and each of their 5 similar words in the␣
 ↪word2vec subspace
for word in keys:
    embeddings = []
    words = []
    for similar_word, _ in model.most_similar(word, topn=5):
        words.append(similar_word)
```

```
        embeddings.append(model[similar_word])
    embedding_clusters.append(embeddings)
    word_clusters.append(words)
```

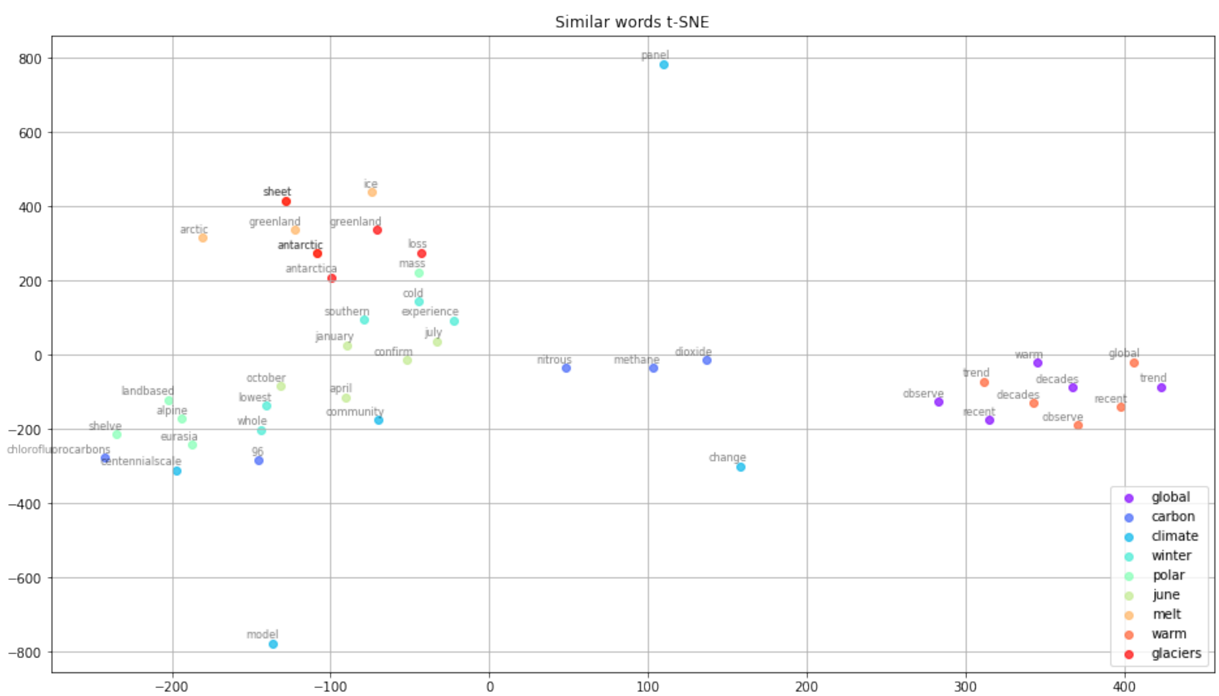
[ ]:
```
from sklearn.manifold import TSNE
import numpy as np

# applying t-SNE on the word cluster
embedding_clusters = np.array(embedding_clusters)
n, m, k = embedding_clusters.shape
tsne_model_en_2d = TSNE(perplexity=10, n_components=2, init='pca', n_iter=5000,␣
↪random_state=0)
embeddings_en_2d = np.array(tsne_model_en_2d.fit_transform(embedding_clusters.
↪reshape(n * m, k))).reshape(n, m, 2)
```

[ ]:
```
import matplotlib.pyplot as plt
import matplotlib.cm as cm

# t-SNE visualizing the word cluster.
def tsne_plot_similar_words(title, labels, embedding_clusters, word_clusters, a,␣
↪filename=None):
    plt.figure(figsize=(16, 9))
    colors = cm.rainbow(np.linspace(0, 1, len(labels)))
    for label, embeddings, words, color in zip(labels, embedding_clusters,␣
↪word_clusters, colors):
        x = embeddings[:, 0]
        y = embeddings[:, 1]
        plt.scatter(x, y, c=color, alpha=a, label=label)
        for i, word in enumerate(words):
            plt.annotate(word, alpha=0.5, xy=(x[i], y[i]), xytext=(5, 2),
                         textcoords='offset points', ha='right', va='bottom', size=8)
    plt.legend(loc=4)
    plt.title(title)
    plt.grid(True)
    if filename:
        plt.savefig(filename, format='png', dpi=150, bbox_inches='tight')
    plt.show()
```

[ ]:
```
tsne_plot_similar_words('Similar words t-SNE', keys, embeddings_en_2d, word_clusters,␣
↪0.7,
                        'similar_words.png')
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
the *color* keyword-argument or provide a 2-D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
the *color* keyword-argument or provide a 2-D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
```

the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.



From the above t-SNE plot of the word clusters, we observe: - words similar to ‘global’(purple color) and ‘warm’(orange color) in the word2vec subspace appear together in the t-SNE subspace aswell. - words similar to ‘carbon’(dark blue) in the word2vec subspace are not appearing together in the t-SNE subspace. We see

words like 'methane' , 'dioxide','nitrous' appear away from 'cluorofluorocarbon' - most of the similar words in the word2vec subspace effectively appear together other than few exceptions. We can say that semantic information is preserved with few information loss when projected in a lower dimension

Comparison of t-SNE embeddings with Word2Vec embeddings

[ ]:
```
# map of test_set words and their word vectors after tSNE
test_embedding_tsne = {}
for key, i in zip(test_embedding.keys(), range(len(tsne_test))):
    test_embedding_tsne[key] = tsne_test[i]
```

[ ]:
```
tsne_test_df = pd.DataFrame(tsne_test)
```

[ ]:
```
cos_sim_tsne = cosine_similarity(tsne_test_df.iloc[:,:].values, Y=None,␣
↪dense_output=False)
cos_df_tsne = pd.DataFrame(cos_sim_tsne, index = test_embedding_tsne.keys(), columns =␣
↪test_embedding_tsne.keys())
```

Pair-wise cosine similarity : TSNE

[ ]:
```
cos_df_tsne
```

[ ]:
```
                the     world     enter  …  subtropics    proper  heatwaves
the        1.000000  0.999821 -0.053707  …   -0.991298 -0.659480   0.803815
world      0.999821  1.000000 -0.072612  …   -0.993613 -0.645123   0.814939
enter     -0.053707 -0.072612  1.000000  …    0.184690 -0.715218  -0.637191
cold       0.995778  0.993860  0.038183  …   -0.975028 -0.725701   0.745814
mode       0.643917  0.629309  0.729408  …   -0.537597 -0.999789   0.062451
…               …         …         …  …           …         …
irregularly -0.610933 -0.595828 -0.757728  …    0.501400  0.998023  -0.020122
periodic   -0.302968 -0.284863 -0.935354  …    0.174879  0.916193   0.323390
subtropics -0.991298 -0.993613  0.184690  …    1.000000  0.554785  -0.875130
proper     -0.659480 -0.645123 -0.715218  …    0.554785  1.000000  -0.082916
heatwaves   0.803815  0.814939 -0.637191  …   -0.875130 -0.082916   1.000000

[3913 rows x 3913 columns]
```

Pair-wise cosine similarity : Word2Vec

[ ]:
```
cos_df_w2v
```

[ ]:
```
                the     world     enter  …  subtropics    proper  heatwaves
the        1.000000  0.937338  0.968786  …    0.940187  0.959345   0.971963
world      0.937338  1.000000  0.967958  …    0.941228  0.961900   0.965301
enter      0.968786  0.967958  1.000000  …    0.972888  0.985430   0.990646
cold       0.952239  0.893898  0.971733  …    0.949144  0.958023   0.967983
mode       0.960934  0.986443  0.989063  …    0.959360  0.983500   0.989856
…               …         …         …  …           …         …
irregularly  0.957003  0.942887  0.984662  …    0.966204  0.989548   0.987611
periodic   0.972193  0.934128  0.986972  …    0.964382  0.983412   0.986317
subtropics 0.940187  0.941228  0.972888  …    1.000000  0.969002   0.969213
proper     0.959345  0.961900  0.985430  …    0.969002  1.000000   0.990005
heatwaves  0.971963  0.965301  0.990646  …    0.969213  0.990005   1.000000
```

```
[3913 rows x 3913 columns]
```

From the above 2 dataframes of pair-wise cosine similarities, we can observe the cosine similarity values of each word in the test set and every other word in the set in the t-SNE and word2vec embedding space.

- Considering the absolute cosine similarities, we notice the cosine similarities are higher in the word2vec embedding space compared to the t-SNE embedding space. This is true for most of the words, signifying that some information is lost. For instance considering the words 'heatwaves' & cold':

  t-SNE cosine similarity : 0.745814

  Word2vec cosine similarity : 0.967983

[ ]:
```
with open('test_embedding_tsne.txt', 'w', encoding="utf-8") as f:
    f.write('%s %s\n' % (len(test_embedding_tsne.keys()), 2))
    for key, value in test_embedding_tsne.items():
        f.write('%s %s %s\n' % (key, value[0],value[1]))
```

[ ]:
```
tsne_model = gensim.models.KeyedVectors.load_word2vec_format('test_embedding_tsne.
↪txt', binary=False)
```

[ ]:
```
tsne_model.most_similar('carbon')
```

[ ]:
```
[('dioxide', 0.9999999403953552),
 ('2', 0.9999989867210388),
 ('co', 0.9999988079071045),
 ('methane', 0.999996542930603),
 ('perry', 0.9999657273292542),
 ('assign', 0.9999539256095886),
 ('inhofe', 0.9999450445175171),
 ('hawaii', 0.9998895525932312),
 ('gas', 0.9998834133148193),
 ('detection', 0.9998701810836792)]
```

[ ]:
```
model.most_similar('carbon')
```

[ ]:
```
[('methane', 0.9804165363311768),
 ('chlorofluorocarbons', 0.9779826402664185),
 ('dioxide', 0.9746006727218628),
 ('nitrous', 0.9659712314605713),
 ('96', 0.9649980068206787),
 ('tropospheric', 0.9622126817703247),
 ('49', 0.9581986665725708),
 ('nonco2', 0.9581005573272705),
 ('oxide', 0.9576266407966614),
 ('release', 0.9563332200050354)]
```

We see the word 'carbon' is most similar to 'dioxide' in the t-SNE subspace whereas similar to 'methane' in the word2vec subspace. Similarly considering the cosine similarities for 'carbon' & 'dioxide':

t-SNE subspace : 0.9999999403953552

Word2vec subspace : 0.9746006727218628

In both the subspace, the most similar words appear in similar context as both 'dioxide' and 'methane' are relevant to 'carbon'

[ ]:
```
tsne_model.most_similar('glaciers')
```

[ ]:
```
[('greenland', 0.9999998211860657),
 ('melt', 0.9999997615814209),
 ('other', 0.9999997615814209),
 ('peerreviewed', 0.9999997019767761),
 ('antarctica', 0.9999996423721313),
 ('sheet', 0.9999995231628418),
 ('today', 0.9999994039535522),
 ('2013', 0.9999994039535522),
 ('cover', 0.9999993443489075),
 ('31000', 0.9999991059303284)]
```

[ ]:
```
model.most_similar('glaciers')
```

[ ]:
```
[('antarctic', 0.9970004558563232),
 ('greenland', 0.9952079653739929),
 ('loss', 0.988825798034668),
 ('antarctica', 0.9882369637489319),
 ('sheet', 0.9868824481964111),
 ('alpine', 0.9844338297843933),
 ('shrinkage', 0.9840260148048401),
 ('snow', 0.9806098341941833),
 ('landbased', 0.9798789024353027),
 ('mass', 0.9784616231918335)]
```

Considering the word 'glacier', we see both the models identify similar context words 'greenland' and 'antarctica', 'sheet' . For the word 'sheet',

t-SNE : 0.9998705387115479

Word2vec: 0.9868824481964111

Similarly considering the context word 'greenland':

t-SNE: 0.9987722039222717

Word2vec: 0.9952079653739929

[ ]:
```
tsne_model.most_similar('june')
```

[ ]:
```
[('5', 0.9999999403953552),
 ('earlier', 0.999998927116394),
 ('drop', 0.9999982714653015),
 ('later', 0.9999980330467224),
 ('ft', 0.9999977946281433),
 ('late', 0.9999963641166687),
 ('observations', 0.9999958276748657),
 ('instrumental', 0.9999943971633911),
 ('month', 0.9999911785125732),
 ('introduction', 0.9999908208847046)]
```

[ ]:
```
model.most_similar('june')
```

[ ]:
```
[('july', 0.9989757537841797),
 ('january', 0.9989748001098633),
 ('confirm', 0.9987897276878357),
 ('april', 0.9987339973449707),
 ('october', 0.9985716342926025),
 ('1988', 0.9983054399490356),
 ('august', 0.9980118870735168),
 ('february', 0.9979966282844543),
 ('december', 0.9978994727134705),
 ('seven', 0.9976174831390381)]
```

t-SNE did not effectively preserve the semantic relationship for the word 'june'. We can observe many out of context words for the t-sne model.

KNN graph TSNE

[ ]:
```
from sklearn.neighbors import kneighbors_graph
A_tsne = kneighbors_graph(tsne_test_df, 5)
```

[ ]:
```
A_tsne.toarray()
```

[ ]:
```
array([[0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       …,
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.]])
```

We obtain the adjacency matrix, which has the value 1 for the 5 closest neighbours of each word and 0 for every other datapoint.

Comparing the adjacency matrix for the word2vec embeddings and the the t-SNE emeddings for the test set, we obtain the f1_score of the embeddings indicating the extent of similarity between them.

The f1-score is particularly useful in this case when the class distubution is uneven. It specifies on average what percent of neighbors for each word are same in word2vec embeddings and t_SNE embeddings.

[ ]:
```
A_tsne = A_tsne.toarray()
```

[ ]:
```
row,column = A_tsne.shape
```

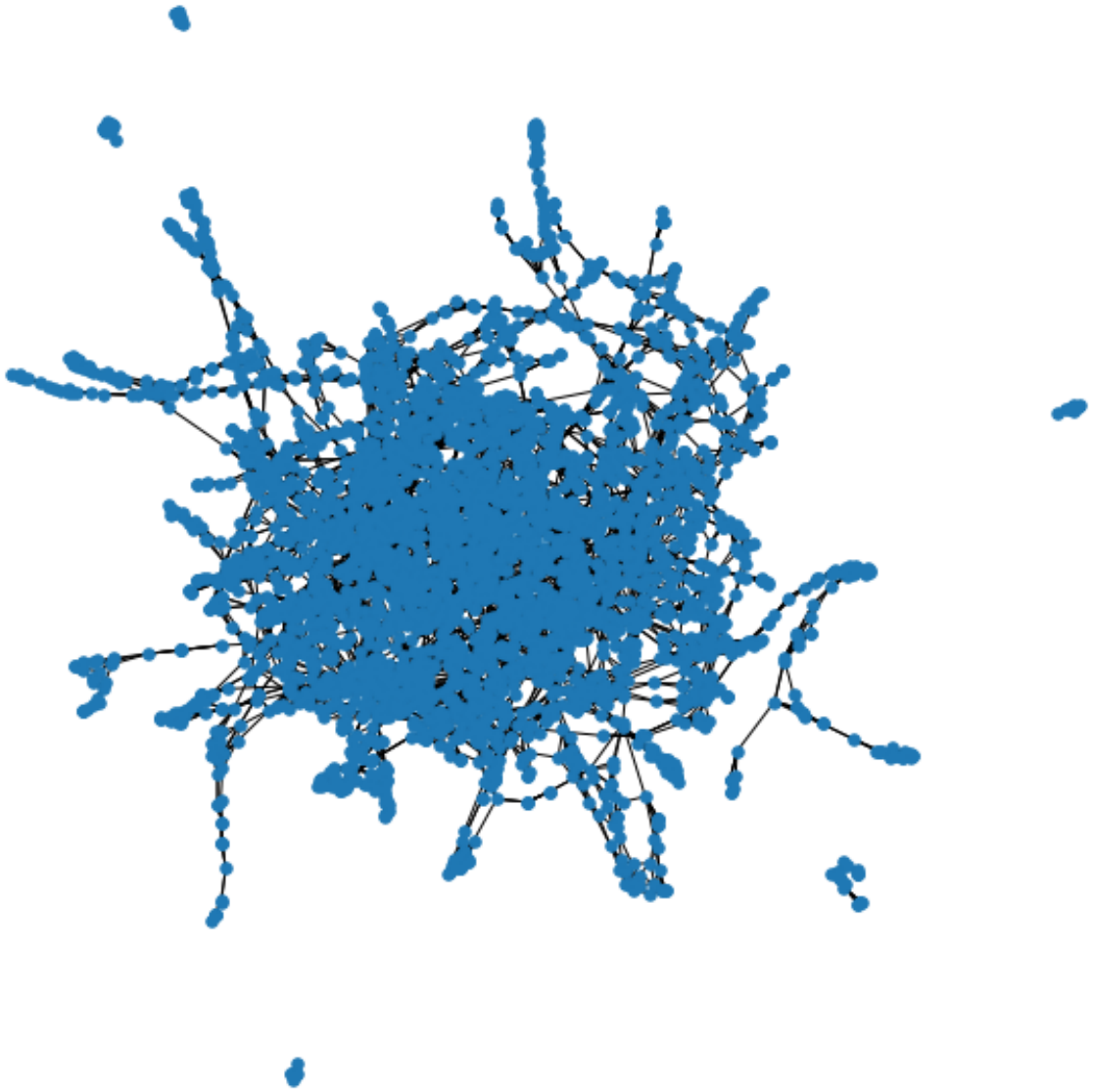
[ ]:
```
f1_score_tsne_sum =0

for i in range(column):
  f1_score_tsne_sum = f1_score_tsne_sum + f1_score(A_word2vec[:,i], A_tsne[:,i])
f1_score_tsne = f1_score_tsne_sum/column
```

[ ]:
```
f1_score_tsne
```

[ ]:
```
0.48610642873100635
```

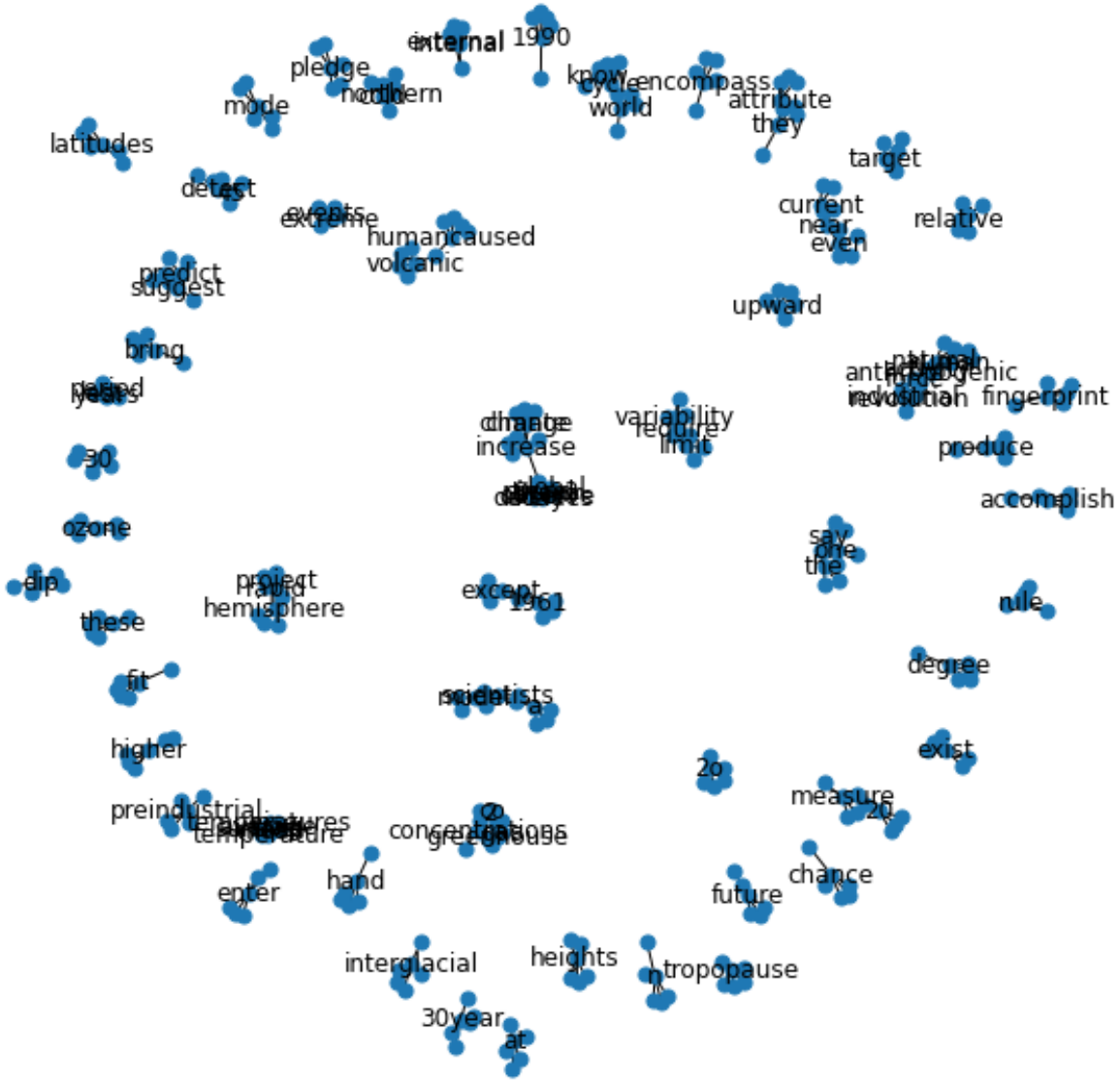
On comparing the KNN-graphs (sparce matrix) columnwise, we compute the average f1_score for the t-SNE and the Word2vec test embedding. We observe that on average 48% of the neighbours of each word in the word2vec embeddings space are same as neighbours in the t-SNE embedding space. Thus some information is lost while conversion to a lower dimension.

[ ]:
```
fig = plt.figure(figsize=(8,8))
show_graph(A_tsne)
```



[ ]:
```
test_embedding_tsne_100 = dict(itertools.islice(test_embedding_tsne.items(),101))
test_word_labels_tsne = make_label_dict(test_embedding_tsne_100.keys())
```

[ ]:
```
fig = plt.figure(figsize=(8,8))
show_graph_with_labels(A_tsne[:101], test_word_labels_tsne)
```



From visualization of KNN graph of t-SNE embeddings for subset of words from test set: - Words like ‘climate’ and ‘change’ appear closer as was the case in w2vec embeddings - Words like ‘greenhouse’ and ‘c02’ appear closer as was the case in w2vec embeddings. - Words like ‘external’ and ‘internal’ which appear in similar context are together as was the case with w2vec embeddings. - We notice similar context words like ‘future’ and ‘current’ far from each other (contrary to w2vec embeddings) - Words ‘northern’ and ‘cold’ are closer to each other (similar case to w2vec embeddings) - Words ‘increase’ and ‘higher’ are not closer to each other (contrary to w2vec embeddings)