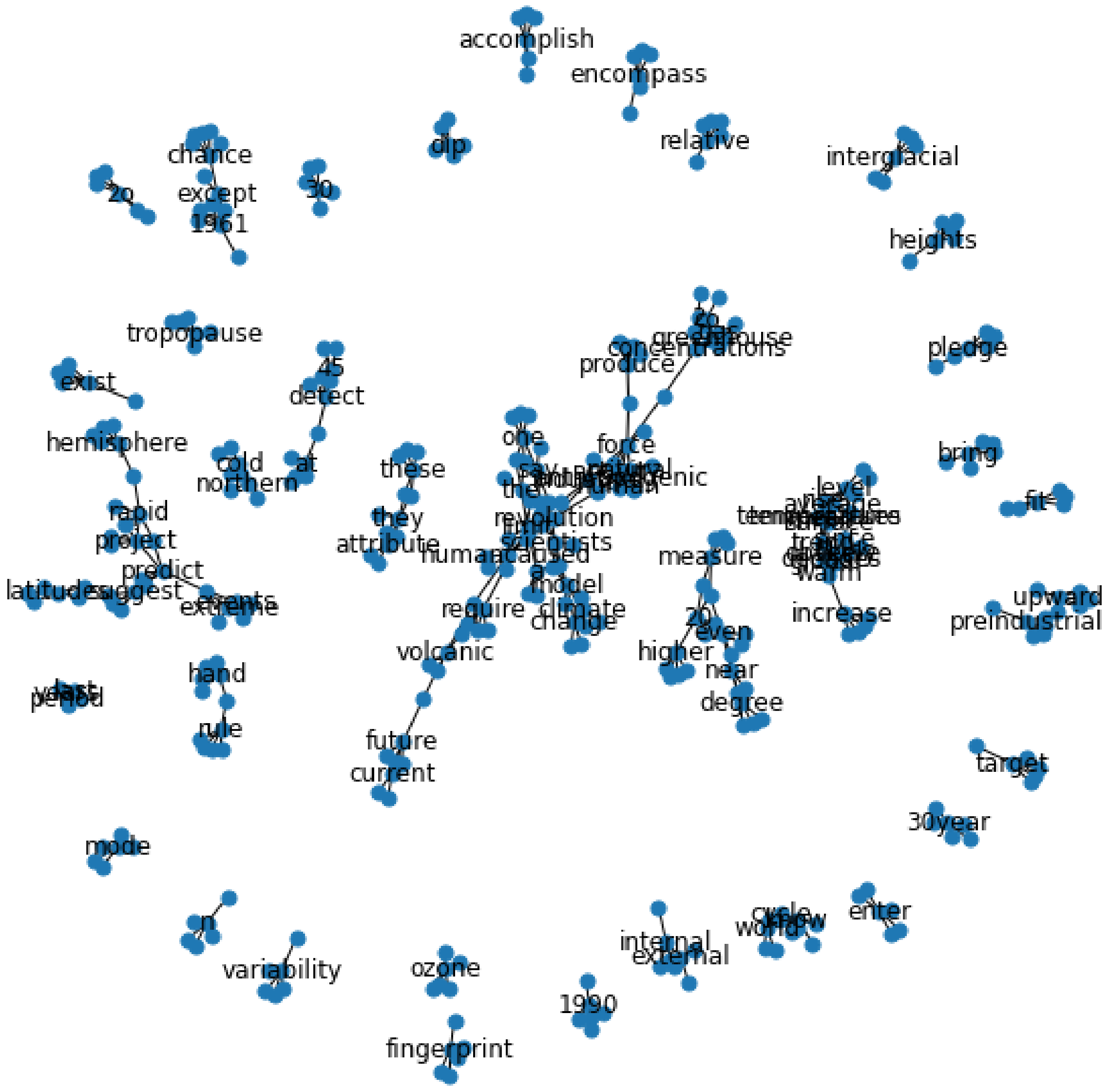


From visualization of the KNN graph of w2vec embeddings for subset of words from test set: - Words like 'greenhouse' and 'c02' appear closer in the w2vec embedding space. - Words like 'external' and 'internal' which appear in similar context are together. - we also notice similar context words like 'future' and 'cuurent' close to each other. - generally north is associated with colder climate. We observe the words 'northern' and 'cold' appear together. - we see opposite words like 'upward','increase' and 'dip' appear farther.

[CM2] Part1 : PCA

Applying PCA on the word2vec embeddings

```
[ ]: from sklearn.decomposition import PCA
     pca = PCA(n_components=30)
```

```
[ ]: pca.fit(list(train_embedding.values()))
[ ]: PCA(copy=True, iterated_power='auto', n_components=30, random_state=None,
         svd_solver='auto', tol=0.0, whiten=False)
```

```
[ ]: pca.transform(list(train_embedding.values()))
```

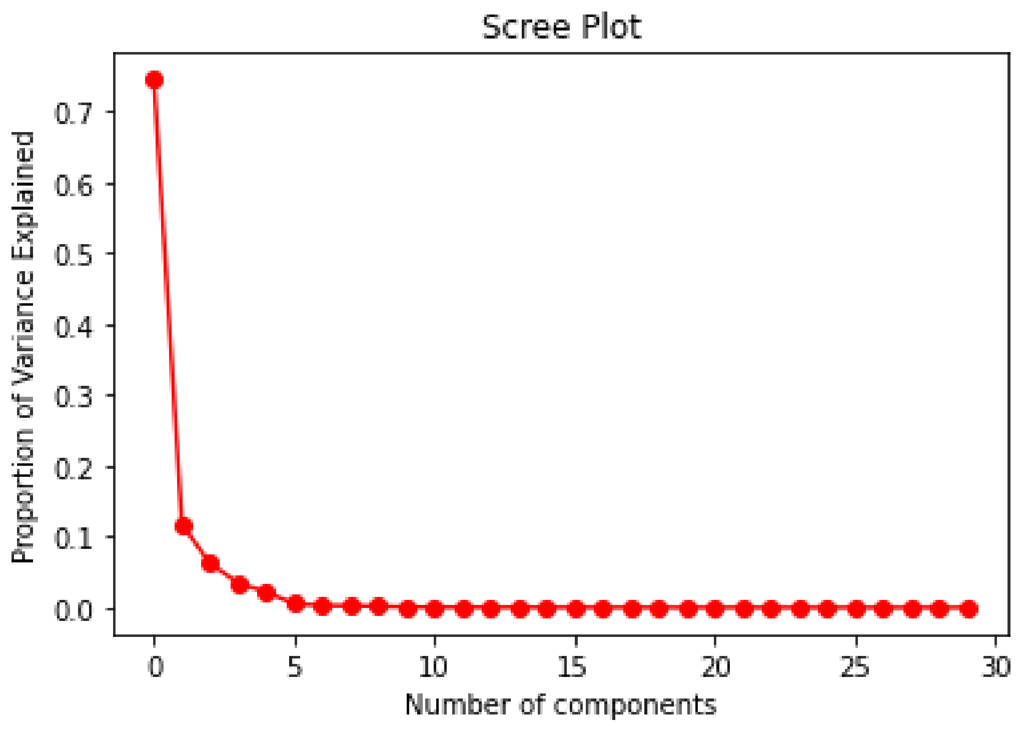
```
[ ]: array([[ 3.87282085e+00,  4.20334800e-01,  2.51241054e+00, …,
              3.31459410e-02,  1.90721085e-02,  3.25079002e-02],
            [ 3.55266811e+00,  6.63576282e-01,  1.79280519e+00, …,
             -4.49735976e-02, -2.96195497e-03, -4.67219325e-02],
            [ 1.92313907e+00,  5.03022436e-01,  1.34305649e-01, …,
             -1.34027107e-02,  3.70457042e-03, -1.28743264e-04],
            …,
            [-3.17859052e-01,  3.86429627e-02,  1.28687361e-02, …,
              3.62537639e-03,  3.30193574e-03,  8.36130100e-04],
            [-6.05596357e-01,  2.52565408e-02,  2.51138336e-02, …,
             -2.40033980e-03,  2.28250228e-03, -3.28717325e-03],
            [-7.76470074e-01,  3.19532157e-02,  3.60747433e-02, …,
              3.42901817e-03,  3.43810597e-03, -1.59703332e-03]])
```

```
[ ]: pca.explained_variance_ratio_
```

```
[ ]: array([7.44995566e-01, 1.17617551e-01, 6.22416387e-02, 3.48808952e-02,
            2.22612852e-02, 5.48456242e-03, 3.10683831e-03, 2.78547588e-03,
            1.95629197e-03, 9.30024503e-04, 5.36743909e-04, 3.83045606e-04,
            3.67237162e-04, 3.44109180e-04, 2.34608930e-04, 1.94853058e-04,
            1.68563651e-04, 1.60019462e-04, 1.11111784e-04, 9.59998797e-05,
            8.22235834e-05, 6.76463519e-05, 6.12879647e-05, 5.35996697e-05,
            4.42476541e-05, 3.74946594e-05, 3.63574616e-05, 3.49072044e-05,
            3.13032872e-05, 2.84629138e-05])
```

Scree plot

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt

     plt.plot(pca.explained_variance_ratio_,'ro-')
     plt.title('Scree Plot')
     plt.xlabel('Number of components')
     plt.ylabel('Proportion of Variance Explained');
```



Principal components are formed in order of the proportion of variation they explain: PC1 captures the most variation, PC2 — the second most, and so on. Each of them contributes some information of the data, and in a PCA, there are as many principal components as there are characteristics. Leaving out PCs and we lose information.

A scree plot shows how much variation each PC captures from the data. The y axis is eigenvalues, which essentially stand for the proportion of variation explained and the number of componetnts in the x-axis. An ideal curve should be steep, then bends at an "elbow" — which is cutting-off point — and after that flattens out.

We observe a steep fall for n_component 1 with a sharp curve at 2 ,3 ,4 and then the curve flattens. In this case, we see the elbow of the scree plot at the fourth principle component. Also from the proportion of variance explained, we hold on to principal components that explain at least 95% of the variance cumulatively. With the fourth principal component, the cumulative proportion of the variance explained surpasses 95%, therefore we would consider to keep four principal components. If a higher threshold were used, then additional principal components would have to be retained.

PCA Pair plot

```
[ ]: # PCA with first 4 dimentionalities
     pca = PCA(n_components=4, random_state = 0)

     # fitting on the train set
     pca.fit(list(train_embedding.values()))
```

```
[ ]: PCA(copy=True, iterated_power='auto', n_components=4, random_state=0,
         svd_solver='auto', tol=0.0, whiten=False)
```

```
[ ]: # transform on train set
     pca_train = pca.transform(list(train_embedding.values()))
```

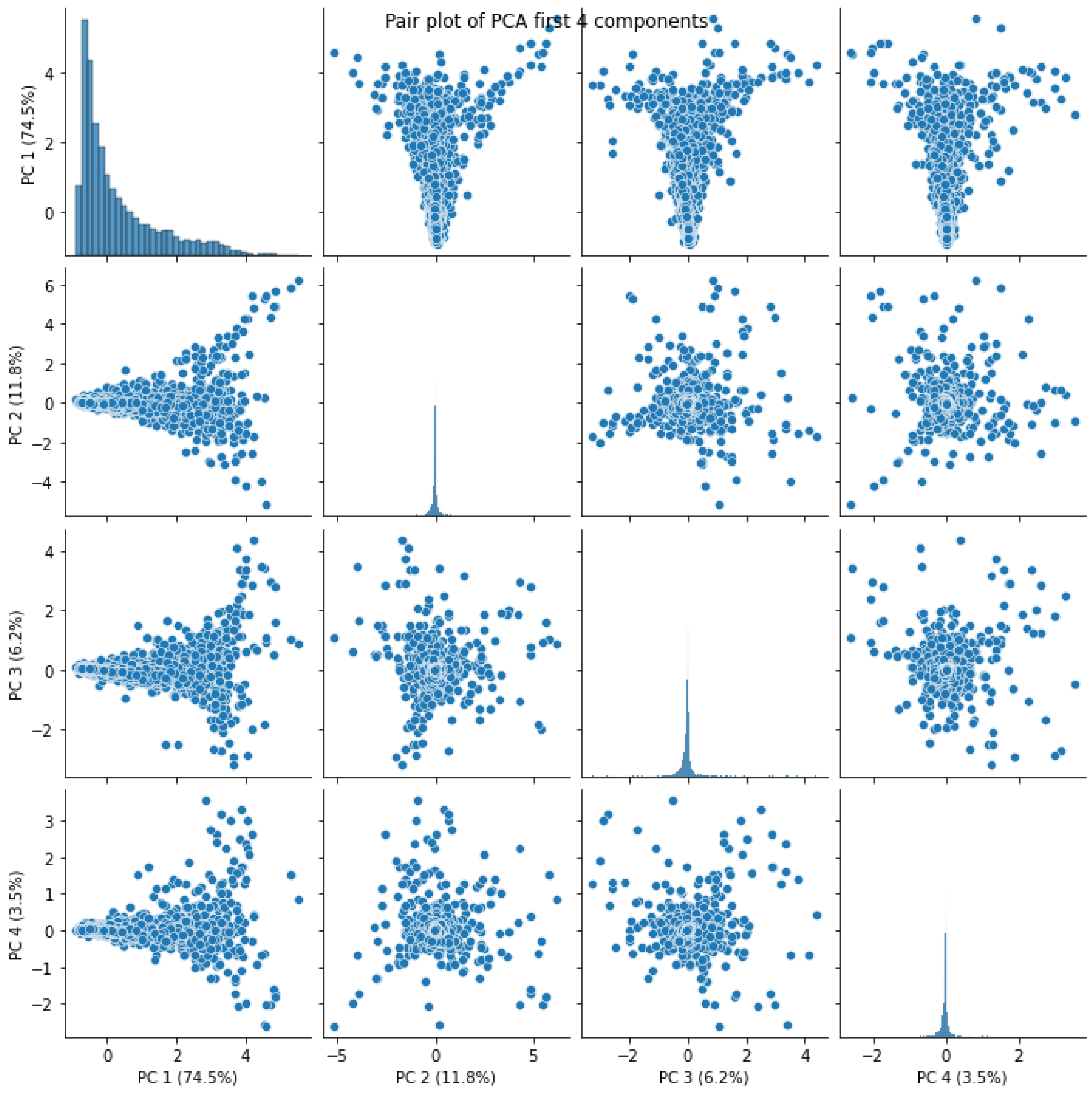
```
[ ]: # transform on test set
     pca_test = pca.transform(list(test_embedding.values()))
```

```
[ ]: pca_test.shape
```

```
[ ]: (3913, 4)
```

```
[ ]: import pandas as pd
     import seaborn as sns

     PC = []

     for i, var in enumerate(pca.explained_variance_ratio_ * 100):
         PC.append(f"PC {i+1} ({var:.1f}%)")

     a = sns.pairplot(pd.DataFrame(pca_test,
                     columns = PC))
     a.fig.suptitle("Pair plot of PCA first 4 components")
```

```
[ ]: Text(0.5, 0.98, 'Pair plot of PCA first 4 components')
```



From the pair plot of the first four PCA dimensionalities, we observe:

- the principal components are orthagonal and there is no inter-dependence. In the graphs, the principal components are perpendicular to one another.
- We observe that most of the datapoints are distributed along the first Principle component. PC1 captures most variance i.e the most variant direction.
- It can be observed that 2nd(PC2) and 3rd(PC3) components are mostly zero until the 1st(PC1) component hits 4. This indicates that a lot of the points are on centred on the X axis in the embedding space.

[ ]:
```python
import numpy
# Calculates the cosine similarity between two arrays
def cos_similarity(A: numpy.ndarray, B: numpy.ndarray) -> float:
    dot_product = A.dot(B)
    norm_of_A = numpy.linalg.norm(A)
    norm_of_B = numpy.linalg.norm(B)
    cos = dot_product/(norm_of_A * norm_of_B)
    return cos
```

[ ]:
```
# PCA with first 4 dimentionalities (as observed from scree plot)
pca = PCA(n_components=4, random_state = 0)

# fitting on the train set
pca.fit(list(train_embedding.values()))

# transform on test set
pca_test = pca.transform(list(test_embedding.values()))
```

[ ]:
```
# creating PCA test embedding dataframe
test_embedding_df = pd.DataFrame(pca_test)
test_embedding_df["words"]=test_embedding.keys()
```

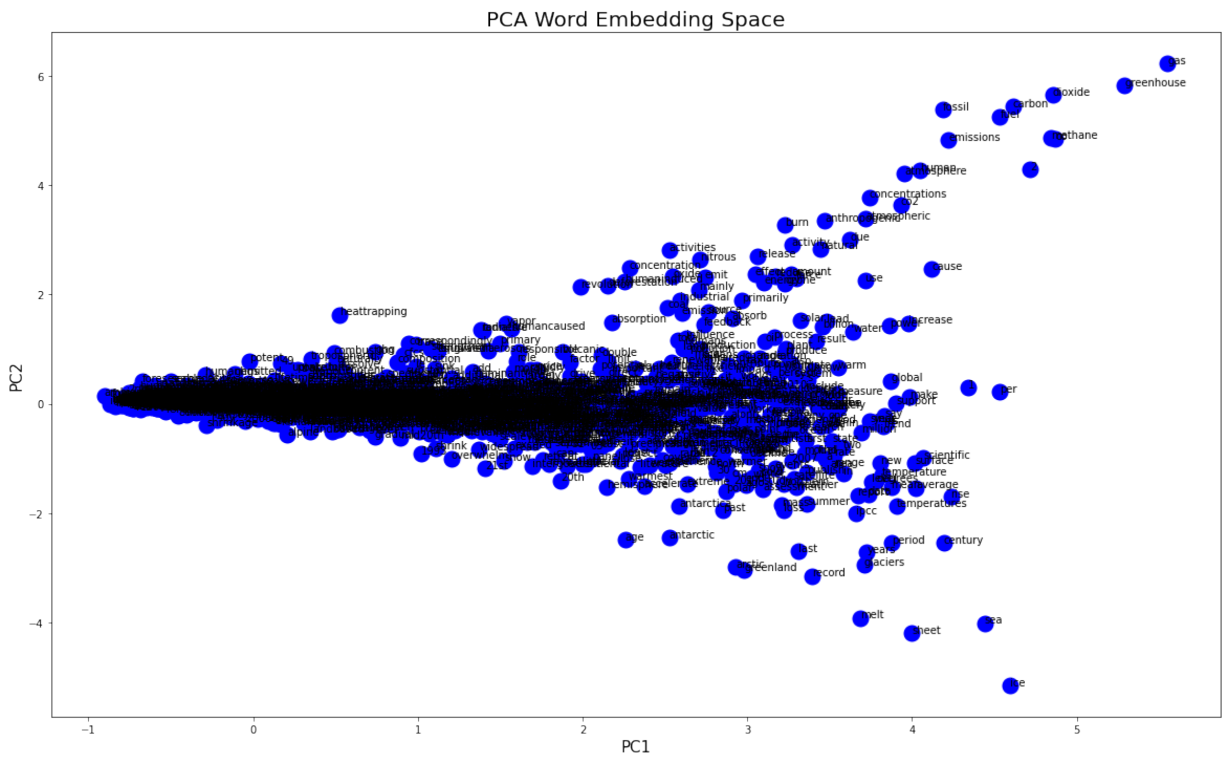
[ ]:
```
test_embedding_df.head()
```

[ ]:
```
          0         1         2         3  words
0  3.223095 -0.598961 -0.652740  0.546134    the
1  2.956923  0.459381 -0.472731 -0.335042  world
2  0.370758 -0.122764 -0.065529 -0.097615  enter
3  3.085381 -1.300527 -0.006679 -0.105545   cold
4  0.941254  0.022955 -0.221132 -0.102709   mode
```

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

[ ]:
```
for key, value in test_embedding_pca.items():
    print(key, ' : ', value)
```

[ ]:
```
# plotting first 2 PC dimentionalities
import matplotlib.pyplot as plt
plt.figure(figsize=(20,12))
plt.scatter(pca_test[:,0],pca_test[:,1],linewidths=10,color='blue')
plt.xlabel("PC1",size=15)
plt.ylabel("PC2",size=15)
plt.title("PCA Word Embedding Space",size=20)
for i, word in enumerate(list(test_embedding.keys())):
  plt.annotate(word,xy=(pca_test[i,0],pca_test[i,1]))
```



Visualizing the embedding space along the first 2 Principle components, we observe: - similar words in the context of global warming like 'carbon', 'dioxide', 'greenhouse', 'methane' ,'CO' , 'emmission' etc. appear close to each other.

- Similarly words like 'ice' ,'melt' ,'sheet' ,'sea' ,'glaciers' , 'greenland', 'antartica' etc. which are all relevant in the context of glacial melt in Antarctica and Greenland and consequent rise in the sea/water level. The model is able to successfully identify words in similar concept as they appear frequently together.
- We see that semantically dissimilar words like 'heat' , 'heattrapping' and 'ice' appear far from each other.
- we also observe words like 'industry' and 'deforestation' appear together. This is indication of the cause effect as formation of industries lead to increased deforestation.

Comparison of PCA embeddings with Word2Vec embeddings

```
[ ]: pca_test_df= pd.DataFrame(pca_test)
```

```
[ ]: # pair-wise cosine similarity
     from sklearn.metrics.pairwise import cosine_similarity
     cos_sim_pca = cosine_similarity(pca_test_df.iloc[:,:].values, Y=None,␣
      ↪dense_output=False)
```

```
[ ]: cos_df_pca = pd.DataFrame(cos_sim_pca, index = test_embedding_pca.keys(), columns =␣
      ↪test_embedding_pca.keys())
```

Pair-wise cosine similarity: PCA

```
[ ]: cos_df_pca
[ ]:                 the     world     enter  …  subtropics    proper  heatwaves
     the          1.000000  0.908707  0.910641  …   -0.963812 -0.961168  -0.713469
     world        0.908707  1.000000  0.887724  …   -0.976891 -0.982604  -0.902158
     enter        0.910641  0.887724  1.000000  …   -0.927138 -0.931666  -0.674697
     cold         0.940426  0.838772  0.961940  …   -0.926769 -0.920986  -0.656603
     mode         0.943576  0.989298  0.934427  …   -0.982107 -0.988565  -0.833331
     …                 …         …        …   …         …         …          …
     irregularly -0.951485 -0.988353 -0.912137  …    0.997740  0.998483   0.883209
     periodic    -0.909643 -0.994670 -0.862743  …    0.980509  0.982707   0.928954
     subtropics  -0.963812 -0.976891 -0.927138  …    1.000000  0.999257   0.861173
     proper      -0.961168 -0.982604 -0.931666  …    0.999257  1.000000   0.859692
     heatwaves   -0.713469 -0.902158 -0.674697  …    0.861173  0.859692   1.000000

     [3913 rows x 3913 columns]
```

```
[ ]: cos_sim_w2v = cosine_similarity(test_embedding_w2vec_df.iloc[:,:].values, Y=None,␣
      ↪dense_output=False)
```

```
[ ]: cos_df_w2v = pd.DataFrame(cos_sim_w2v, index = test_embedding.keys(), columns =␣
      ↪test_embedding.keys())
```

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 PCA and word2vec embedding space.

- we observe negative cosine similarities in PCA subspace. This could be attributed to centering of data during PCA application.
- considering the absolute cosine similarities, we notice the cosine similarities are higher in the word2vec embedding space compared to the PCA embedding space. For instance considering the words 'world' & cold' :

  PCA cosine similarity : 0.838772

  Word2vec cosine similarity : 0.893898

```
[ ]: with open('test_embedding_pca.txt', 'w', encoding="utf-8") as f:
         f.write('%s %s\n' % (len(test_embedding_pca.keys()), 4))
         for key, value in test_embedding_pca.items():
             f.write('%s %s %s %s %s\n' % (key, value[0],value[1],value[2],value[3]))
```

```
[ ]: pca_model = gensim.models.KeyedVectors.load_word2vec_format('test_embedding_pca.txt',␣
      ↪binary=False)
```

Comparing cosine similarities of few signifant/important words:

```
[ ]: # pca model
     pca_model.most_similar('carbon',topn=5)
```

```
[ ]: [('dioxide', 0.9959695339202881),
      ('correspondingly', 0.9943975210189819),
      ('methane', 0.9942948818206787),
      ('content', 0.9927193522453308),
      ('nitrous', 0.9893808364868164)]
```

```
[ ]: # w2vec model
     model.most_similar('carbon',topn=5)
```

```
[ ]: [('methane', 0.9804165363311768),
      ('chlorofluorocarbons', 0.9779826402664185),
      ('dioxide', 0.9746006727218628),
      ('nitrous', 0.9659712314605713),
      ('96', 0.9649980068206787)]
```

We see the word 'carbon' is most similar to 'dioxide' in the PCA subspace whereas similar to 'methane' in the word2vec subspace. Similarly considering the cosine similarities for 'carbon' & 'dioxide':

PCA subspace : 0.9959695339202881

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'

```
[ ]: # pca model
     pca_model.most_similar('glaciers',topn=5)
```

```
[ ]: [('antarctica', 0.9976590871810913),
      ('antarctic', 0.9949178099632263),
      ('dramatic', 0.994631826877594),
      ('bottom', 0.9943311214447021),
      ('greenland', 0.9923969507217407)]
```

```
[ ]: # w2vec model
     model.most_similar('glaciers',topn=5)
```

```
[ ]: [('antarctic', 0.9970004558563232),
      ('greenland', 0.9952079653739929),
      ('loss', 0.988825798034668),
      ('antarctica', 0.9882369637489319),
 ('sheet', 0.9868824481964111)]
```

Considering the word ‘glaciers’, we see both the models identify similar context words ‘antarica’/‘antarctic’ with relatively similar cosine similarity.

PCA : 0.9976590871810913

Word2vec: 0.9970004558563232

Similarly considering the context word ‘greenland’:

PCA: 0.9923969507217407

Word2vec: 0.9952079653739929

But we also notice words like ‘dramatic’ and ‘bottom’ in the PCA subspace which aren’t essentially relevant to ‘glaciers’

```
[ ]: # pca model
     pca_model.most_similar('june',topn=5)
```

```
[ ]: [('october', 0.9998493790626526),
      ('extend', 0.9997817277908325),
      ('2003', 0.9995821118354797),
      ('sit', 0.9992489814758301),
      ('america', 0.9992311000823975)]
```

```
[ ]: # w2vec model
     model.most_similar('june',topn=5)
```

```
[ ]: [('july', 0.9989757537841797),
      ('january', 0.9989748001098633),
      ('confirm', 0.9987897276878357),
      ('april', 0.9987339973449707),
      ('october', 0.9985716342926025)]
```

Considering the word ‘june’ , we observe the cosine similarities in Word2vec subspace show similar moth words like ‘july’, ‘january’, ‘april’ etc, in the PCA subspace other than ‘october’ we see out of context words like ‘sit’, ‘america’ etc. Considering ‘june’ and ‘october’:

PCA : 0.9998493790626526

Word2vec : 0.9985716342926025

KNN graph for PCA

Applying KNN-graph over the PCA test embeddings with 5 nearest neighbours.

```
[ ]: A_pca = kneighbors_graph(pca_test_df, 5)
```

```
[ ]: A_pca.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 PCA 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 pca embedding.

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

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

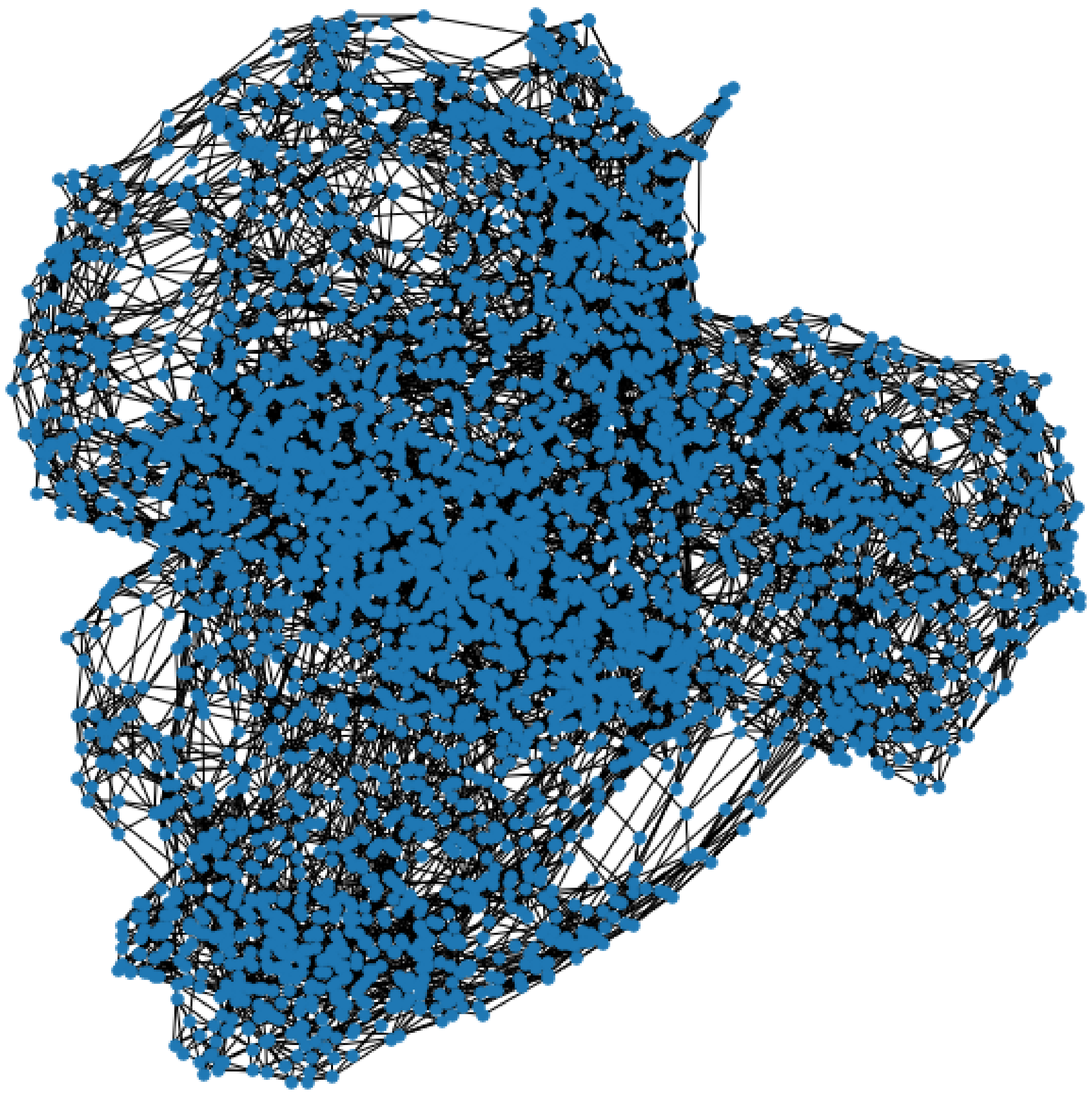
```
[ ]: from sklearn.metrics import f1_score

     f1_score_pca_sum =0

     for i in range(column):
       f1_score_pca_sum = f1_score_pca_sum + f1_score(A_word2vec[:,i], A_pca[:,i])
     f1_score_pca = f1_score_pca_sum/column
```

```
[ ]: f1_score_pca
```

```
[ ]: 0.590302227153497
```

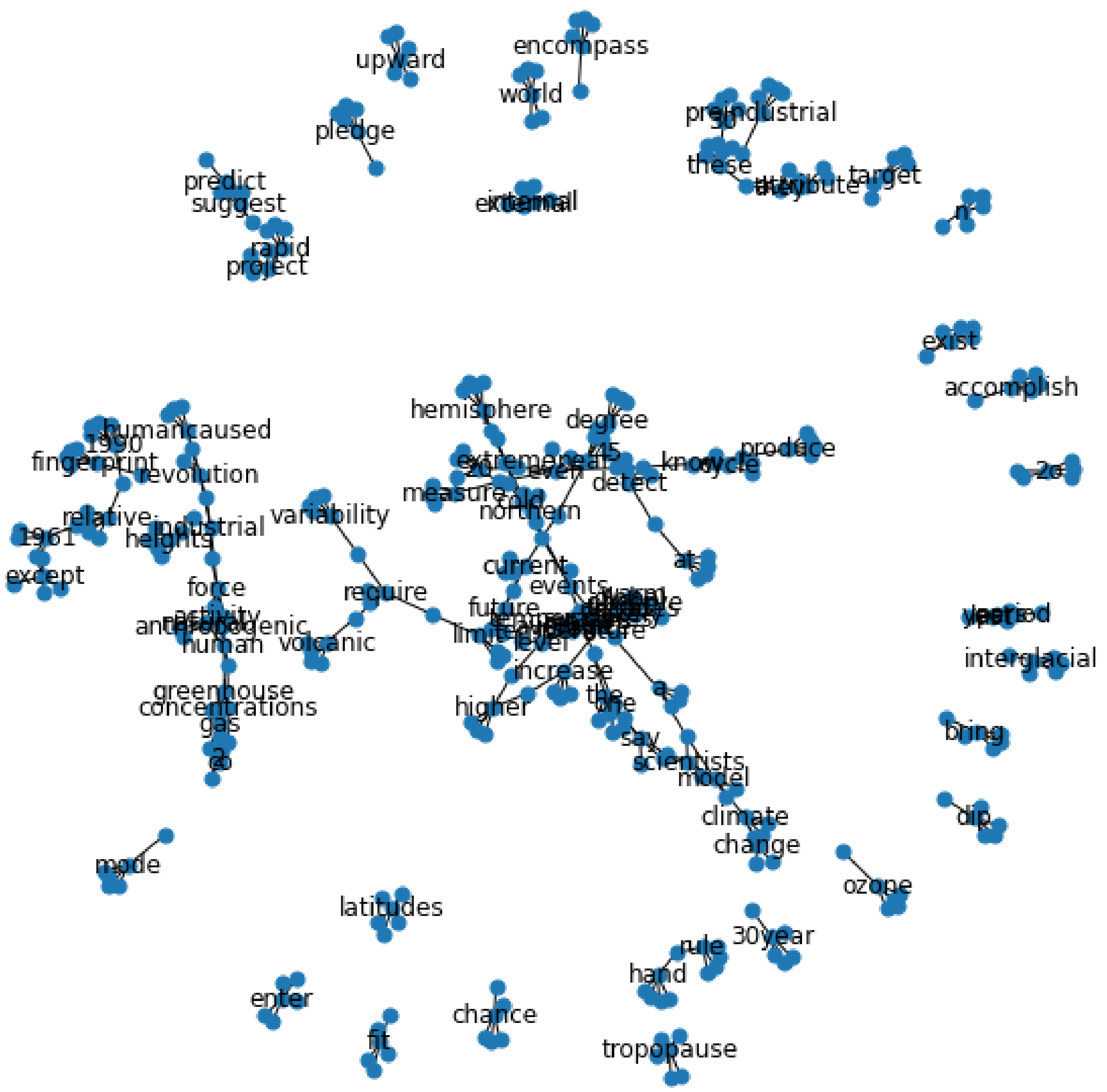
On comparing the KNN-graphs (sparce matrix) columnwise, we conpute the average f1_scoe for the PCA and the Word2vec test embedding. We observe that 59% of the neighbours of each word in the word2vec embeddings space are same as neighbours in the PCA embedding space. Thus PCA is able to maintain the semantic similarity of the word vectors.

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



```
[ ]: test_embedding_pca_100 = dict(itertools.islice(test_embedding_pca.items(),101))
     test_word_labels_pca = make_label_dict(test_embedding_pca_100.keys())
```

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



From visualization of the KNN graph of PCA 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', 'co2', 'gas' appear closer in the PCA embedding space. This is similar to what we observed in the word2vec subspace. - Words like 'external' and 'internal' which appear in similar context are overlapping. - We observe the words 'northern' and 'cold' appear together. This is similar to what we observed in the word2vec subspace. - we see opposite words like 'upward','increase' and 'dip' appear farther in the PCA embedding space. - In general, visually comparing the Word2vec KNN-Graph and the PCA KNN-Graph, we see the in the PCA, the distance between the nearest neighbours is reduced and they appear overlapping to each other, which is not the case in word2vec.