ECE 657A : Data and Knowledge Modeling and Analysis

Assignment 3 : Embedding with dimensionality reduction methods and embedding methods for natural language

Import libraries

[ ]:
```
!pip install datasets
```

[ ]:
```
from datasets import load_dataset
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
import gensim
import pandas as pd
import numpy as np
import re
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings(action = 'ignore')
```

Load dataset

CITATION (CLIMATE-FEVER Dataset)

@misc{diggelmann2020climatefever, title={CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims}, author={Thomas Diggelmann and Jordan Boyd-Graber and Jannis Bulian and Massimiliano Ciaramita and Markus Leippold}, year={2020}, eprint={2012.00614}, archivePrefix={arXiv}, primaryClass={cs.CL} }

[ ]:
```
dataset = load_dataset("climate_fever")
```

```
Using custom data configuration default
Reusing dataset climate_fever (/root/.cache/huggingface/datasets/climate_fever/default/1.
0.1/3b846b20d7a37bc0019b0f0dcbde5bf2d0f94f6874f7e4c398c579f332c4262c)
```

[CM1]

[ ]:
```
dataset.data
```

[ ]:
```
{'test': pyarrow.Table
 claim: string
 claim_id: string
 claim_label: int64
 evidences: list<item: struct<article: string, entropy: float, evidence: string,
evidence_id: string, evidence_label: int64, votes: list<item: string>>>
   child 0, item: struct<article: string, entropy: float, evidence: string, evidence_id:
string, evidence_label: int64, votes: list<item: string>>
       child 0, article: string
       child 1, entropy: float
       child 2, evidence: string
       child 3, evidence_id: string
       child 4, evidence_label: int64
```

```
        child 5, votes: list<item: string>
            child 0, item: string}
```

[ ]:
```
dataset.column_names
```

[ ]:
```
{'test': ['claim', 'claim_id', 'claim_label', 'evidences']}
```

[ ]:
```
dataset.shape
```

[ ]:
```
{'test': (1535, 4)}
```

[ ]:
```
datadict= dataset['test']
```

[ ]:
```
datadict[0]
```

[ ]:
```
{'claim': 'Global warming is driving polar bears toward extinction',
 'claim_id': '0',
 'claim_label': 0,
 'evidences': [{'article': 'Extinction risk from global warming',
   'entropy': 0.6931471824645996,
   'evidence': '"Recent Research Shows Human Activity Driving Earth Towards Global
Extinction Event".',
   'evidence_id': 'Extinction risk from global warming:170',
   'evidence_label': 2,
   'votes': ['SUPPORTS', 'NOT_ENOUGH_INFO', None, None, None]},
  {'article': 'Global warming',
   'entropy': 0.0,
   'evidence': 'Environmental impacts include the extinction or relocation of many
species as their ecosystems change, most immediately the environments of coral reefs,
mountains, and the Arctic.',
   'evidence_id': 'Global warming:14',
   'evidence_label': 0,
   'votes': ['SUPPORTS', 'SUPPORTS', None, None, None]},
  {'article': 'Global warming',
   'entropy': 0.6931471824645996,
   'evidence': 'Rising temperatures push bees to their physiological limits, and could
cause the extinction of bee populations.',
   'evidence_id': 'Global warming:178',
   'evidence_label': 2,
   'votes': ['SUPPORTS', 'NOT_ENOUGH_INFO', None, None, None]},
  {'article': 'Habitat destruction',
   'entropy': 0.0,
   'evidence': 'Rising global temperatures, caused by the greenhouse effect, contribute
to habitat destruction, endangering various species, such as the polar bear.',
   'evidence_id': 'Habitat destruction:61',
   'evidence_label': 0,
   'votes': ['SUPPORTS', 'SUPPORTS', None, None, None]},
  {'article': 'Polar bear',
   'entropy': 0.6931471824645996,
   'evidence': '"Bear hunting caught in global warming debate".',
   'evidence_id': 'Polar bear:1328',
   'evidence_label': 2,
```

```
'votes': ['SUPPORTS', 'NOT_ENOUGH_INFO', None, None, None]}]}
```

Data Fields

- claim_id: a string feature, unique claim identifier.
- claim: a string feature, claim text.
- claim_label: a int feature, overall label assigned to claim (based on evidence majority vote). The label correspond to 0: "refutes", 1: "supports" and 2: "not enough info"
- evidences: a list of evidences with fields:
- evidence_id: a string feature, unique evidence identifier.
- evidence_label: a int feature, micro-verdict label. The label correspond to 0: "refutes", 1: "supports" and 2: "not enough info"
- article: a string feature, title of source article (Wikipedia page).
- evidence: a string feature, evidence sentence.
- entropy: a float32 feature, entropy reflecting uncertainty of evidence_label.
- votes: a list of string features, corresponding to individual votes.

We see from the dataset that the fields "claim" and the "evidence" (within "evidences") have significant word data. Hence we will extract these particular fields of the dataset to build the text corpus for our word2vec model and further analysis.

[ ]:
```
sentences = []
```

[ ]:
```
# extract 'claim' and 'evidences' from the dataset
for key in datadict:
    print('claim : ' , key['claim'])
    sentences.append(key['claim'])
    #paragraph = paragraph + key['claim']
    for evidence in key['evidences']:
        print('evidence : ' , evidence['evidence'])
        sentences.append(evidence['evidence'])
        #paragraph = paragraph + evidence['evidence']
```

Preparing the dataset :

- word tokenization.
- stop word removal (like 'the','is' etc.)
- lower case conversion.
- punctuation , special characters removal
- word lemmatization

[ ]:
```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[ ]:
```
True
```

[ ]:
```
# Preparing the dataset
#sentences = nltk.sent_tokenize(paragraph)

# word tokenization
sentences = [nltk.word_tokenize(sentence) for sentence in sentences]

import string
table = str.maketrans('', '', string.punctuation)
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

for i in range(len(sentences)):
    # stopword removal
    sentences[i] = [word.lower() for word in sentences[i] if word not in stopwords.
 ↪words('english')]
    # punctuation removal
    sentences[i] = [word.translate(table) for word in sentences[i]]
    sentences[i] = [word for word in sentences[i] if word.isalnum()]
    # lemmatization of words
    sentences[i] = [lemmatizer.lemmatize(word, pos="v") for word in sentences[i]]
```

[ ]:
```
for i in range(3):
    print(sentences[i])
```

```
['global', 'warm', 'drive', 'polar', 'bear', 'toward', 'extinction']
['recent', 'research', 'show', 'human', 'activity', 'drive', 'earth', 'towards',
'global', 'extinction', 'event']
['environmental', 'impact', 'include', 'extinction', 'relocation', 'many', 'species',
'ecosystems', 'change', 'immediately', 'environments', 'coral', 'reef', 'mountains',
'arctic']
```

Train-Test split dataset

Splitting the data corpus into train set (80%) and test set (20%) maintaining the order (shuffle = False)

[ ]:
```
from sklearn.model_selection import train_test_split

print(len(sentences))

# dividing the data into train and test sets (80%, 20%) with random_state=0
train_set, test_set = train_test_split(sentences, test_size=0.2,␣
 ↪random_state=0,shuffle=False)
```

```
9210
```

[ ]:
```
print('train: ',len(train_set))
print('test: ',len(test_set))
```

```
train:  7368
test:  1842
```

[ ]:
```
for i in range(3):
    print(train_set[i])
```

```
['global', 'warm', 'drive', 'polar', 'bear', 'toward', 'extinction']
['recent', 'research', 'show', 'human', 'activity', 'drive', 'earth', 'towards',
'global', 'extinction', 'event']
['environmental', 'impact', 'include', 'extinction', 'relocation', 'many', 'species',
'ecosystems', 'change', 'immediately', 'environments', 'coral', 'reef', 'mountains',
'arctic']
```

Initializing and training the Word2vec model

Initializing and training the word2vec model. Keeping the min_count as 2 – Ignores all words with total frequency lower than this. This will filter out insignificant words that appear only once in the data corpus.

[ ]:
```
# Training the Word2Vec model
model = Word2Vec(min_count=2,seed=0)
model.build_vocab(sentences)  # prepare the model vocabulary
model.train(sentences, total_examples=model.corpus_count, epochs=model.epochs)
```

[ ]:
```
# # Training the Word2Vec model
# model = Word2Vec(sentences, seed=0, min_count=2)
```

[ ]:
```
model.save("word2vec.model")
```

[ ]:
```
model = Word2Vec.load("word2vec.model")
```

[ ]:
```
words= model.wv.vocab
len(words)
```

[ ]:
```
6268
```

[ ]:
```
# getting unique words from the train set
flat_list = []
for sent in train_set:
    for word in sent:
      if word not in flat_list:
        flat_list.append(word)
train_set = flat_list
```

Obtain the embeddings of the train set

[ ]:
```
# map of train_set words and their word2vec embedding
train_embedding ={}
for word in train_set:
    try:
        train_embedding[word]=model.wv[word]
    except KeyError:
        print("The word "+word+" does not appear in this model")
```

[ ]:
```
print(train_embedding.keys())
```

[ ]:
```
# gettign unique words from the test set
flat_list = []
for sent in test_set:
    for word in sent:
      if word not in flat_list:
        flat_list.append(word)
test_set = flat_list
```

obtaining the embeddings of the test set

[ ]:
```
# map of test_set words and their word2vec embedding
test_embedding ={}
for word in test_set:
    try:
        test_embedding[word]=model.wv[word]
    except KeyError:
        print("The word "+word+" does not appear in this model")
```

[ ]:
```
print(test_embedding.keys())
```

[ ]:
```
len(train_embedding.keys())
len(test_embedding.keys())
```

[ ]:
```
3913
```

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

Model exploration and analysis

[ ]:
```
# Most similar words
model.most_similar('global', topn=1)
```

[ ]:
```
[('warm', 0.9816120862960815)]
```

In the context of the dataset "climate-fever" which is primarily regarding climate change, we can expect that most frequently the words "global" & "warming" appear together. From the word2vec model, we observe the word 'global' is most similar to 'warm' (lemmatized word for 'warming').

[ ]:
```
# Most similar words
model.most_similar('climate', topn=1)
```

[ ]:
```
[('change', 0.9676931500434875)]
```

Similarly in the context of the dataset "climate-fever" which is primarily regarding climate change, we can expect frequent occurrance of "climate" & "change" appear tofether frequently. From the word2vec model, we observe the words 'climate' and 'change' have the most similarity.

[ ]:
```
# Most similar words
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)]
```

Considering the word "carbon", in the context of the dataset "climate-fever", we observe:

- 'carbon' has high cosine similarity with 'dioxide'. In word2vec, words that appear together or in the same sentences are considered very similar.
- we see other greenhouse gases like methane, chlorofluorocarbons in the most similar words. The model is able to succesfully identify similar gases.
- we also see words like 'nitrous', 'oxide' , 'tropospheric' etc. which are words associated with 'carbon' and gases in general in most scientific disciplines.

[ ]:
```
# Most similar words
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)]
```

The dataset is about scientific claims regarding climate change and evidences that support or refute these claims. It discusses about melting of glaciers and ice sheets, specifically in the regions of Antartica and Greenland and the subsequent rise in sea/water level. The model has done very well in identifying these context words. Here we notice that 'glacier' has similar context words 'antartica' , 'greenland' , and also has other related words like 'snow' , 'sheet' , 'alpine' etc.

[ ]:
```
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)]
```

Let's see if the model is able to identify words which aren't particularly relevent to context of the dataset i.e. climate change and global warming. Considering the word 'june', we observe the model has successfuly found similar month and date related words 'july' , 'april', 'august','january','february' etc. Also it identifies related years.

Arithmetic Relations

The word2vec model is trained on a small dataset and training it from scratch may not arrive at the best possible arithmetic relationships. Let's consider the 5-10 most similar words and see if we can find any relevant arithmetic relation.

summer - warm + cold ~ refreezing

Considering the relation 'Summer' - 'Warm' + 'Cold' should ideally result in 'Winter' as removinf warmth from summer and including cold essentially is winter. Applying the relation on the word2vec model, we see words like 'alpine' , 'refreezing', 'polar' etc. This gives us a crude arithmetic relation : 'summer'-'warm'+'cold' ~ refreezing (as winter essentially is refreezing)

```
[ ]: model.most_similar(positive=['summer','cold'], negative=['warm'], topn=10)
```

```
[ ]: [('1550', 0.9293469190597534),
      ('impassable', 0.9237136840820312),
      ('alpine', 0.9208871126174927),
      ('refreezing', 0.9150192737579346),
      ('40y', 0.9142007827758789),
      ('dieoff', 0.911822497844696),
      ('navy', 0.9110009670257568),
      ('polar', 0.9089462757110596),
      ('13300', 0.9063601493835449),
      ('eurasia', 0.9051717519760132)]
```

january - february + june ~ july

Considering the sequencing of the months in the calender, if 'January' is followed by 'February' , then 'June' is followed by 'July'. Applying this relation in the word2vec model, we see that 'july' does appear in one of the top most similar words.

```
[ ]: model.most_similar(positive=['january','june'], negative=['february'])
```

```
[ ]: [('every', 0.9972406625747681),
      ('seven', 0.9969903230667114),
      ('july', 0.9963981509208679),
      ('confirm', 0.9960317611694336),
      ('on', 0.9959323406219482),
      ('6', 0.9954519271850586),
      ('april', 0.9953603744506836),
      ('october', 0.9952815771102905),
      ('december', 0.9952090382575989),
      ('1983', 0.994957685470581)]
```

glaciers + heat ~ seawater

Global warming results in melting of glaciers into seawater/oceanwater. Applying this relation on the word2vec model, we observe word 'seawater' as one of the most similar words.

```
[ ]: model.most_similar(positive=['glaciers','heat'])
```

[ ]:
```
[('expansion', 0.9939089417457581),
 ('seawater', 0.9708890318870544),
 ('continental', 0.9691341519355774),
 ('darker', 0.9684371948242188),
 ('polar', 0.9649854898452759),
 ('cover', 0.9648905396461487),
 ('gain', 0.9628483057022095),
 ('mmyear', 0.9628068208694458),
 ('float', 0.9625867605209351),
 ('landbased', 0.9614496231079102)]
```

'energy' - 'pollution' + 'renewable' ~ 'biofuels'

Another interesting Arithmetic relation would be 'energy' - 'pollution' + 'renewable'. This should result in clean and renewable energy sources with low pollution. Applying this relation on the word2vec model, we observe words like 'biofuels' , 'solar' etc. which are essentially clean renewable energy sources. Also we see words like 'lowcarbon', 'cheaper' etc. which are properties of clean enery sources.

[ ]:
```
model.most_similar(positive=['energy','renewable'], negative=['pollution'])
```

[ ]:
```
[('biofuels', 0.955267608165741),
 ('concentrate', 0.93912672996521),
 ('power', 0.9225142002105713),
 ('source', 0.9130250215530396),
 ('lowcarbon', 0.9109171628952026),
 ('nonrenewable', 0.8967939615249634),
 ('cheaper', 0.8908413648605347),
 ('modulations', 0.8907933235168457),
 ('solar', 0.8899375200271606),
 ('fertiliser', 0.8811256885528564)]
```

energy + renewable ~ biofuels

Similarly 'energy' + 'renewable' should result in clean and renewable energy sources. We observe words like 'biofuels','solar'. Also properties of renewable energy sources like 'cheaper', 'lowcarbon'. etc.

[ ]:
```
model.most_similar(positive=['energy','renewable'])
```

[ ]:
```
[('power', 0.9934578537940979),
 ('lowcarbon', 0.9864583611488342),
 ('biofuels', 0.9843843579292297),
 ('source', 0.9831869602203369),
 ('capture', 0.9740509986877441),
 ('cheaper', 0.973953902721405),
 ('artificial', 0.9714807271957397),
 ('convert', 0.9713934063911438),
 ('solar', 0.9706745743751526),
 ('modulations', 0.9705424308776855)]
```

[ ]:
```
model.doesnt_match("mars jupiter pluto earth saturn".split())
```

[ ]:
```
'earth'
```

water + vaporize ~ air

In general physics, vaporizing water should result in water vapors which are essentially in gaseous form. Thus 'water' + 'vaporize' should result in 'vapor' or 'gas'. We observe applying this arithmetic relation gives us 'air' which can be considered 'gas'.

[ ]:
```
# water + vaporize ~ air
model.most_similar(positive=['water','vaporize'])
```

[ ]:
```
[('air', 0.9855868220329285),
 ('would', 0.9784165024757385),
 ('acidification', 0.9757006168365479),
 ('radiation', 0.9707937240600586),
 ('contribute', 0.9706382155418396),
 ('lead', 0.9705119132995605),
 ('infrared', 0.9660016298294067),
 ('result', 0.9651824235916138),
 ('continue', 0.9625753164291382),
 ('oceans', 0.9619765281677246)]
```

KNN-graph analysis

KNN-graph finds the k-closest neighbours of a word and returns a sparce adjacency matrix. Considering 5 closest neighbours.

The k-nearest neighbor graph (k-NNG) is a graph in which two vertices p and q are connected by an edge, if the distance between p and q is among the k-th smallest distances from p to other objects from P.

[ ]:
```
# plot KNN-graph
import networkx as nx
def show_graph(adjacency_matrix):
    rows, cols = np.where(adjacency_matrix == 1)
    edges = zip(rows.tolist(), cols.tolist())
    gr = nx.Graph()
    gr.add_edges_from(edges)
    nx.draw(gr, node_size=30)
    plt.show()
```

[ ]:
```
def make_label_dict(labels):
    l = {}
    for i, label in enumerate(labels):
        l[i] = label
    return l
```

[ ]:
```
# plot KNN-graph with labels
def show_graph_with_labels(adjacency_matrix, mylabels):
    rows, cols = np.where(adjacency_matrix == 1)
    edges = zip(rows.tolist(), cols.tolist())
    gr = nx.Graph()
    gr.add_edges_from(edges)
    nx.draw(gr, node_size=50, labels=mylabels)
    plt.show()
```

[ ]:
```
test_embedding_w2vec_df = pd.DataFrame(test_embedding.values())
```

```
from gensim.models import Word2Vec
import warnings
warnings.filterwarnings(action = 'ignore')
```

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Load Pre-trained models

Glove (glove.6B.50d.txt)

Word2vec (GoogleNews-vectors-negative300.bin)

[4]:
```
word2vec_model = gensim.models.KeyedVectors.load_word2vec_format('/content/drive/
↪MyDrive/models/GoogleNews-vectors-negative300.bin', binary=True)
```

[5]:
```
glove_model = gensim.models.KeyedVectors.load_word2vec_format('/content/drive/MyDrive/
↪models/glove.6B.50d.txt', binary=False)
```

Arithmetic Retlations : Pre-trained Glove model

summer - warm + cold ~ refreezing

[6]:
```
glove_model.most_similar(positive=['summer','cold'], negative=['warm'])
```

[6]:
```
[('beginning', 0.7870005965232849),
 ('winter', 0.7821016311645508),
 ('decade', 0.7730634808540344),
 ('spring', 0.7685137987136841),
 ('during', 0.7624093890190125),
 ('since', 0.7607327103614807),
 ('years', 0.7551678419113159),
 ('began', 0.7520835995674133),
 ('fall', 0.7488090395927429),
 ('1970s', 0.7334692478179932)]
```

january - february + june ~ july

[7]:
```
glove_model.most_similar(positive=['january','june'], negative=['february'])
```

[7]:
```
[('july', 0.9906297326087952),
 ('april', 0.9890736937522888),
 ('december', 0.9836546182632446),
 ('november', 0.9835898876190186),
 ('september', 0.9797542691230774),
 ('march', 0.9792064428329468),
 ('october', 0.9773406386375427),
 ('august', 0.9732651114463806),
 ('1997', 0.8851771354675293),
 ('2003', 0.8739202618598938)]
```

glaciers + heat ~ seawater

[9]:
```
glove_model.most_similar(positive=['glaciers','heat'])
```

[9]:
```
[('melting', 0.8702500462532043),
 ('melt', 0.8458359837532043),
 ('temperatures', 0.8333518505096436),
 ('dust', 0.805218517780304),
 ('snow', 0.8004544973373413),
 ('ice', 0.7872774004936218),
 ('moisture', 0.7816673517227173),
 ('temperature', 0.7707889676094055),
 ('clouds', 0.7609851956367493),
 ('surface', 0.7441148161888123)]
```

energy - pollution + renewable ~ biofuels

[11]:
```
glove_model.most_similar(positive=['energy','renewable'], negative=['pollution'])
```

[11]:
```
[('petroleum', 0.6494619846343994),
 ('generating', 0.6310369968414307),
 ('geothermal', 0.6235097646713257),
 ('venture', 0.618798553943634),
 ('energies', 0.6125441193580627),
 ('partnership', 0.6076962351799011),
 ('exploration', 0.6071913838386536),
 ('ethanol', 0.6046498417854309),
 ('portfolio', 0.5932126045227051),
 ('resources', 0.5902050137519836)]
```

water + vaporize ~ air

[12]:
```
glove_model.most_similar(positive=['water','vaporize'])
```

[12]:
```
[('moisture', 0.8320767283439636),
 ('seawater', 0.8137732148170471),
 ('soak', 0.7827821969985962),
 ('melt', 0.7818216681480408),
 ('contaminate', 0.7801991701126099),
 ('seep', 0.7770259976387024),
 ('liquid', 0.7680455446243286),
 ('absorb', 0.7642533183097839),
 ('dust', 0.7550925016403198),
 ('boiling', 0.7518549561500549)]
```

Arithmetic Retlations : Pre-trained Word2Vec model

summer - warm + cold ~ refreezing

[13]:
```
word2vec_model.most_similar(positive=['summer','cold'], negative=['warm'])
```

[13]:
```
[('winter', 0.5936813950538635),
 ('spring', 0.550656259059906),
 ('summertime', 0.5165988206863403),
 ('summers', 0.5085427165031433),
 ('autumn', 0.49106645584106445),
 ('week', 0.45701584219932556),
 ('midwinter', 0.4565219581127167),
```

```
 ('Summer', 0.4489292800426483),
 ('springtime', 0.4475139379501343),
 ('month', 0.4461055397987366)]
```

january - february + june ~ july

[14]:
```
word2vec_model.most_similar(positive=['january','june'], negative=['february'])
```

[14]:
```
[('november', 0.5794603824615479),
 ('september', 0.564159631729126),
 ('april', 0.5304318070411682),
 ('july', 0.5125601291656494),
 ('feb', 0.49811315536499023),
 ('october', 0.49126890301704407),
 ('december', 0.48233509063720703),
 ('monday', 0.4683570861816406),
 ('friday', 0.45918336510658264),
 ('nov', 0.4566681981086731)]
```

glaciers + heat ~ seawater

[15]:
```
word2vec_model.most_similar(positive=['glaciers','heat'])
```

[15]:
```
[('mountain_glaciers', 0.6199347972869873),
 ('glacial_melting', 0.6081671714782715),
 ('icecaps', 0.6007778644561768),
 ('glaciers_melt', 0.5998938083648682),
 ('glacier', 0.5900542736053467),
 ('Alpine_glaciers', 0.5851379632949829),
 ('glacial_melt', 0.5845843553543091),
 ('meltwater', 0.5841852426528931),
 ('temperature', 0.5816020369529724),
 ('Melting_glaciers', 0.5737295150756836)]
```

energy - pollution + renewable ~ biofuels

[16]:
```
word2vec_model.most_similar(positive=['energy','renewable'], negative=['pollution'])
```

[16]:
```
[('renewable_energy', 0.6533094644546509),
 ('renewables', 0.5936083197593689),
 ('renewable_fuels', 0.5332350134849548),
 ('Iberdrola_Renovables_SA_IBR.MC', 0.5124478936195374),
 ('dispatchable', 0.5093342065811157),
 ('renewable_energies', 0.5059199333190918),
 ('Renewable', 0.503628134727478),
 ('Renewable_Energy', 0.492643266916275),
 ('bioenergy', 0.48423200845718384),
 ('baseload', 0.48131799697875977)]
```

water + vaporize ~ air

[17]:
```
word2vec_model.most_similar(positive=['water','vaporize'])
```

[17]:
```
[('vapor_condenses', 0.6094914674758911),
 ('porpoise_vaults', 0.5915391445159912),
 ('ionize', 0.5787661075592041),
 ('vaporise', 0.5759463906288147),
 ('backwashed', 0.573494553565979),
 ('electrolyze', 0.5729873180389404),
 ('wax_melts', 0.569877564907074),
 ('water_vapor_condenses', 0.5677950382232666),
 ('vaporises', 0.5676530599594116),
 ('Rinse_thoroughly', 0.5656528472900391)]
```

Comparing Arithmetic relation between our w2vec model (climate-fever) with Pre-trained Glove(glove-wiki-gigaword-50) and Word2vec models(GoogleNews)

Since the pre-trained models are trained on a large data corpus, the models will be able to retain semantic similarity of the words better than our model which is trained on a small dataset (climate-fever)

Arithmetic Relation	Model (climate-fever)	Pre-trained Glove	Pre-trained Word2Vec
summer - warm + cold ~ refreezing On the above arithmetic relation, while we get '**refreezing**' on our model, the pre-trained glove and word2vec models are able to effectively identify '**winter**' which is more semantically appropriate to the arithmetic relation.	[('1550', 0.9293469190597534), ('impassable', 0.9237136840820312), ('alpine', 0.9208871126174927), ('refreezing', 0.9150192737579346), ('40y', 0.9142007827758789), ('dieoff', 0.911822497844696), ('navy', 0.9110009670257568), ('polar', 0.9089462757110596), ('13300', 0.9063601493835449), ('eurasia', 0.9051717519760132)]	[('beginning', 0.7870005965232849), ('winter', 0.7821016311645508), ('decade', 0.7730634808540344), ('spring', 0.768513798713684), ('during', 0.7624093890190125), ('since', 0.7607327103614807), ('years', 0.7551678419113159), ('began', 0.7520835995674133), ('fall', 0.7488090395927429), ('1970s', 0.7334692478179932)]	[('winter', 0.5936813950538635), ('spring', 0.550656259059906), ('summertime', 0.5165988206863403), ('summers', 0.5085427165031433), ('autumn', 0.49106645584106445), ('week', 0.45701584219932556), ('midwinter', 0.4565219581127167), ('Summer', 0.4489292800426483), ('springtime', 0.4475139379501343), ('month', 0.4461055397987366)]
january - february + june ~ july January is followed by February, thus June is followed by **July**. We notice that all the 3 models are able to appropriately identify (at least in top five most similar words) the above relationship. Thus we can say that all the models are able to preserve semantic similarities between generic words like calendar months which aren't necessarily specific to the context of the dataset they are trained on.	[('every', 0.9972406625747681), ('seven', 0.9969903230667114), ('july', 0.9963981509208679), ('confirm', 0.9960317611694336), ('on', 0.9959323406219482), ('6', 0.9954519271850586), ('april', 0.9953603744506836), ('october', 0.9952815771102905), ('december', 0.9952090382575989), ('1983', 0.994957685470581)]	[('july', 0.9906297326087952), ('april', 0.9890736937522888), ('december', 0.9836546182632446), ('november', 0.9835898876190186), ('september', 0.9797542691230774), ('march', 0.9792064428329468), ('october', 0.9773406386375427), ('august', 0.9732651114463806), ('1997', 0.8851771354675293), ('2003', 0.8739202618598938)]	[('november', 0.5794603824615479), ('september', 0.564159631729126), ('april', 0.5304318070411682), ('july', 0.5125601291656494), ('feb', 0.49811315536499023), ('october', 0.49126890301704407), ('december', 0.48233509063720703), ('monday', 0.4683570861816406), ('friday', 0.45918336510658264), ('nov', 0.456681981086731)]
glaciers + heat ~ seawater While our model identifies '**seawater'** (which is closely relevant as glacier melts to become seawater), the pre-trained models have effectively identifies	[('expansion', 0.9939089417457581), ('seawater', 0.9708890318870544), ('continental', 0.9691341519355774), ('darker', 0.9684371948242188),	[('melting', 0.8702500462532043), ('melt', 0.8458359837532043), ('temperatures', 0.8333518505096436), ('dust', 0.805218517780304), ('snow', 0.8004544973373413),	[('mountain_glaciers', 0.6199347972869873), ('glacial_melting', 0.6081671714782715), ('icecaps', 0.6007778644561768), ('glaciers_melt', 0.5998938083648682),

'**melting'** and '**glacial_melting'** in the context of the above arithmetic relation.	('polar', 0.9649854898452759), ('cover', 0.9648905396461487), ('gain', 0.9628483057022095), ('mmyear', 0.9628068208694458), ('float', 0.9625867605209351), ('landbased', 0.9614496231079102)]	('ice', 0.7872774004936218), ('moisture', 0.7816673517227173), ('temperature', 0.7707889676094055), ('clouds', 0.7609851956367493), ('surface', 0.7441148161888123)]	('glacier', 0.5900542736053467), ('Alpine_glaciers', 0.5851379632949829), ('glacial_melt', 0.5845843553543091), ('meltwater', 0.5841852426528931), ('temperature', 0.5816020369529724), ('Melting_glaciers', 0.5737295150756836)]
energy - pollution + renewable ~ biofuels For the above arithmetic relation, while our model identifies '**biofuel'** (which is relevant as biofuels are clean and renewable), the pre-trained glove model doesn't effectively identify context words. But '**geothermal'** can be considered close as it is a renewable energy source. The pre-trained word2vec model correctly identifies '**renewable_energy**' as the context word for the arithmetic relation.	[('biofuels', 0.955267608165741), ('concentrate', 0.93912672996521), ('power', 0.9225142002105713), ('source', 0.9130250215530396), ('lowcarbon', 0.9109171628952026), ('nonrenewable', 0.8967939615249634), ('cheaper', 0.8908413648605347), ('modulations', 0.8907933235168457), ('solar', 0.8899375200271606), ('fertiliser', 0.8811256885528564)]	[('petroleum', 0.6494619846343994), ('generating', 0.6310369968414307), ('geothermal', 0.6235097646713257), ('venture', 0.618798553943634), ('energies', 0.6125441193580627), ('partnership', 0.6076962351799011), ('exploration', 0.6071913838386536), ('ethanol', 0.6046498417854309), ('portfolio', 0.5932126045227051), ('resources', 0.5902050137519836)]	[('renewable_energy', 0.6533094644546509), ('renewables', 0.5936083197593689), ('renewable_fuels', 0.5332350134849548), ('Iberdrola_Renovables_SA_IBR.MC', 0.5124478936195374), ('dispatchable', 0.5093342065811157), ('renewable_energies', 0.5059199333190918), ('Renewable', 0.503628134727478), ('Renewable_Energy', 0.492643266916275), ('bioenergy', 0.48423200845718384), ('baseload', 0.48131799697875977)]
water + vaporize ~ air While water vaporization is not necessarily '**air'** as identified by our model. The pre-trained glove model effectively identifies '**moisture'** (as it is vapourized water) whereas the pre-trained word2vec model doesn't really identify semantically similar context words.	[('air', 0.9855868220329285), ('would', 0.9784165024757385), ('acidification', 0.9757006168365479), ('radiation', 0.9707937240600586), ('contribute', 0.9706382155418396), ('lead', 0.9705119132995605), ('infrared', 0.9660016298294067), ('result', 0.9651824235916138), ('continue', 0.9625753164291382), ('oceans', 0.9619765281677246)]	[('moisture', 0.8320767283439636), ('seawater', 0.8137732148170471), ('soak', 0.7827821969985962), ('melt', 0.7818216681480408), ('contaminate', 0.7801991701126099), ('seep', 0.7770259976387024), ('liquid', 0.7680455446243286), ('absorb', 0.7642533183097839), ('dust', 0.7550925016403198), ('boiling', 0.7518549561500549)]	[('vapor_condenses', 0.6094914674758911), ('porpoise_vaults', 0.5915391445159912), ('ionize', 0.5787661075592041), ('vaporise', 0.5759463906288147), ('backwashed', 0.573494553565979), ('electrolyze', 0.5729873180389404), ('wax_melts', 0.569877564907074), ('water_vapor_condenses', 0.5677950382232666), ('vaporises', 0.5676530599594116), ('Rinse_thoroughly', 0.5656528472900391)]

In general physics, vaporizing water should result in water vapors which are essentially in gaseous form. Thus 'water' + 'vaporize' should result in 'vapor' or 'gas'. We observe applying this arithmetic relation gives us 'air' which can be considered 'gas'.

[ ]:
```
# water + vaporize ~ air
model.most_similar(positive=['water','vaporize'])
```

[ ]:
```
[('air', 0.9855868220329285),
 ('would', 0.9784165024757385),
 ('acidification', 0.9757006168365479),
 ('radiation', 0.9707937240600586),
 ('contribute', 0.9706382155418396),
 ('lead', 0.9705119132995605),
 ('infrared', 0.9660016298294067),
 ('result', 0.9651824235916138),
 ('continue', 0.9625753164291382),
 ('oceans', 0.9619765281677246)]
```

KNN-graph analysis

KNN-graph finds the k-closest neighbours of a word and returns a sparce adjacency matrix. Considering 5 closest neighbours.

The k-nearest neighbor graph (k-NNG) is a graph in which two vertices p and q are connected by an edge, if the distance between p and q is among the k-th smallest distances from p to other objects from P.

[ ]:
```
# plot KNN-graph
import networkx as nx
def show_graph(adjacency_matrix):
    rows, cols = np.where(adjacency_matrix == 1)
    edges = zip(rows.tolist(), cols.tolist())
    gr = nx.Graph()
    gr.add_edges_from(edges)
    nx.draw(gr, node_size=30)
    plt.show()
```

[ ]:
```
def make_label_dict(labels):
    l = {}
    for i, label in enumerate(labels):
        l[i] = label
    return l
```

[ ]:
```
# plot KNN-graph with labels
def show_graph_with_labels(adjacency_matrix, mylabels):
    rows, cols = np.where(adjacency_matrix == 1)
    edges = zip(rows.tolist(), cols.tolist())
    gr = nx.Graph()
    gr.add_edges_from(edges)
    nx.draw(gr, node_size=50, labels=mylabels)
    plt.show()
```

[ ]:
```
test_embedding_w2vec_df = pd.DataFrame(test_embedding.values())
```

[ ]:
```
# KNN-graph for word2vec embedding
from sklearn.neighbors import kneighbors_graph
A_word2vec = kneighbors_graph(test_embedding_w2vec_df, 5)
```

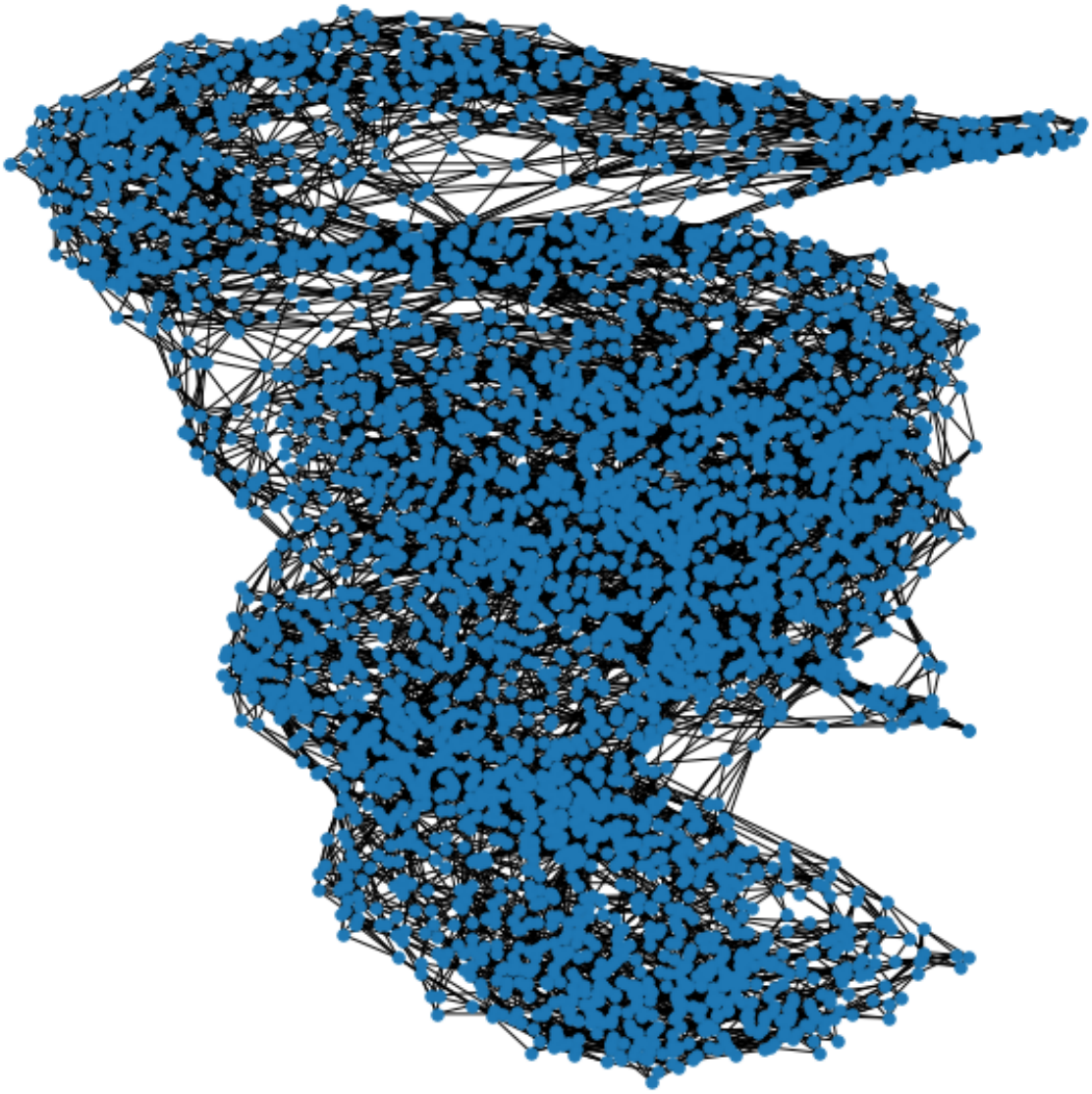
[ ]:
```
A_word2vec.toarray()
```

[ ]:
```
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.

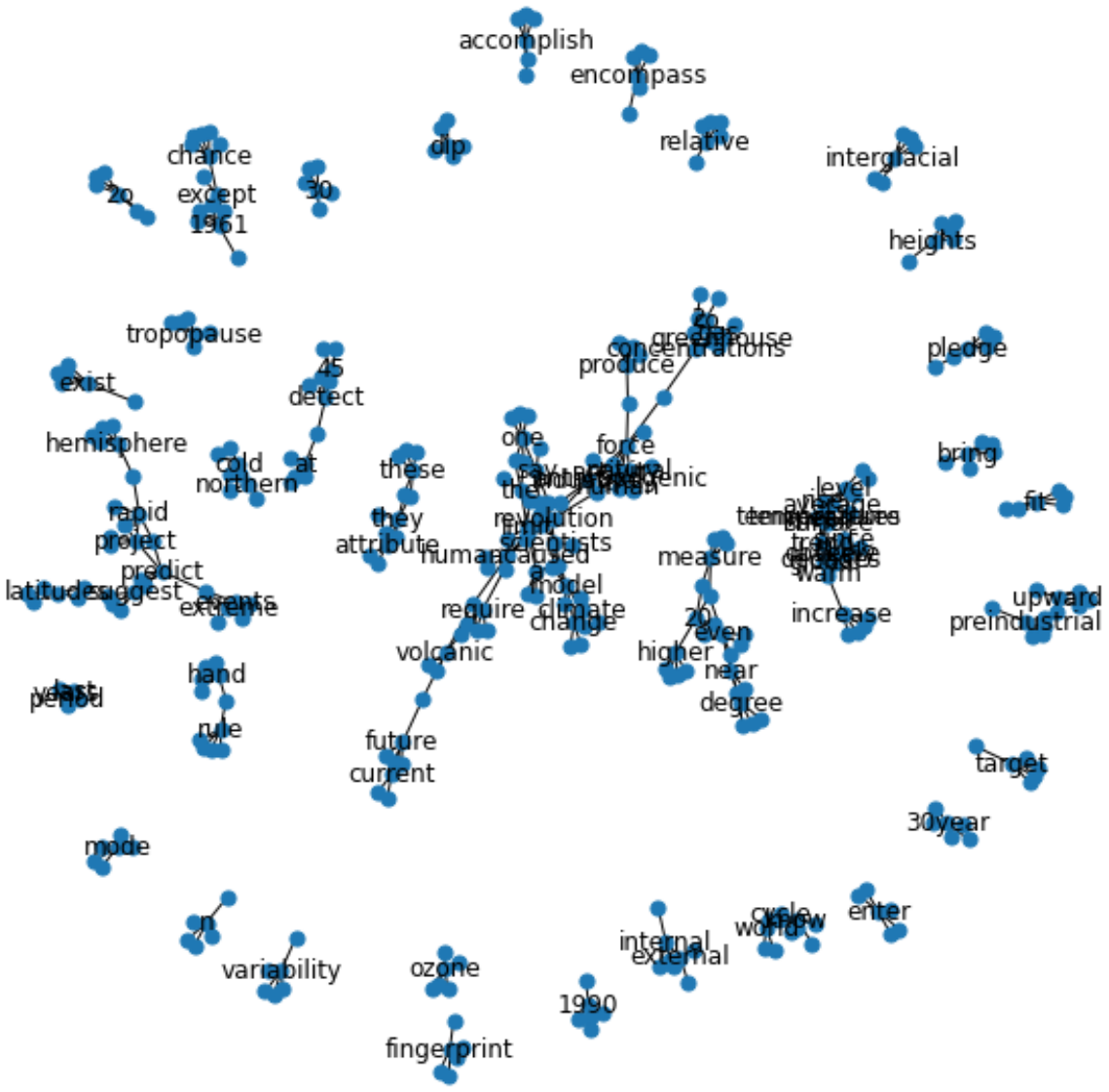
[ ]:
```
A_word2vec = A_word2vec.toarray()
```

[ ]:
```
# KNN-graph for the entire embedding space
fig = plt.figure(figsize=(8,8))
show_graph(A_word2vec)
```



[ ]:
```
import itertools
test_embedding_w2vec_100 = dict(itertools.islice(test_embedding.items(),101))
test_word_labels_w2vec = make_label_dict(test_embedding_w2vec_100.keys())
```

[ ]:
```
# KNN-graph for a subset of the embeddings
fig = plt.figure(figsize=(8,8))
show_graph_with_labels(A_word2vec[:101], test_word_labels_w2vec)
```



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()))
```