Assignment 2 HAL Last date for submission
100 marks 05- March-2023

Fine Print first :)

- Try your assignment with a smaller corpus initially.
- If you are familiar with functional-style programming, use it.
- Write at least 1-3 lines of comments for every function.
- Share the working code.
- Python notebooks should be available on the Colab platform (Google).
- Please make sure that all the results are available when you share them. Incomplete python notebooks will not be evaluated.
- Share your notebooks to ramaseshan.ta@gmail.com and aslahahmadfaizi@gmail.com with viewing rights.
- We will not run/change your Python notebook.
- If no result was found, the assignment will NOT be evaluated.
- Follow the same naming convention as followed in assignment one.
- Assignments will not be graded if they are sent to personal email ids as attachments. Use the share option of Colab to be considered for evaluation.

Implement HAL Experiment 2 - Multidimensional Scaling

Implement experiment 2 as mentioned in the paper using the COVID-19 database. After building the word vectors, pick up four nouns from the vocabulary list, list similar words in a table and plot a graph similar to the one found in the HAL paper (Figure 2 -Multidimensional scaling of co-occurrence vectors. You may display the results in 4 different plots, if you find it difficult to represent in a single plot. You may use TNSE plots for this purpose.

If you have access to powerful machine, you may try the entire corpus for creating the incidence matrix. Otherwise, try to reduce the vocabulary size to around 14000 to compute the cooccurrence relations.

Assignment 2 HAL Last date for submission
100 marks 05- March-2023

Large matrices are very common in machine learning and NLP applications. There are several ways to handle large matrix. If the matrix is sparse, then you may try https://docs.scipy.org/doc/scipy/reference/sparse.html. Otherwise, you may use pandas or use h5py.

Same code is given for your convenience -You may use it, but at your own risk 😄

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.zeros((30000,30000)),dtype=np.int16)
hdf5file = pd.HDFStore('cm_hal.hdf5', 'w') #file size = 1.8G
hdf5file['data'] = df
hdf5file.close()
```

```
#Direct approach using h5py
import h5py
f = h5py.File('halmatrix.hdf5','w')
hm = f.create_dataset("HAL_CM",
                        (voc_count,voc_count),
                        dtype=np.int16,compression="gzip")

#Access the matrix 'HAL_CM' using hm or using f['HAL_CM']
# hm[1:], hm[:1], hm[1,1],hm[100,23]... or
a = f['HAL_CM'][1,234]
```

```
#Creating word vectors - left to right
grams11 = ngrams(tokens, 11, pad_right=True, pad_left=True)

f = h5py.File('halmatrix.hdf5', 'w')

hal_l2r = f.create_dataset('L2R',
                           (voc_count, voc_count),
                           dtype=np.int16)

ramp = [0, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

for index, grams in enumerate(grams11):
    reversed_grams = grams[::-1]

    if not None in grams:
        # Left to right
        for idx, gram in enumerate(grams):
           hal_l2r[vocab_idx[grams[0]],
                    vocab_idx[grams[idx]]] += ramp[idx]

        # right to left
        for idx, gram in enumerate(reversed_grams):
            hal_r2l[vocab_idx[reversed_grams[0]],
                    vocab_idx[reversed_grams[idx]]] += ramp[idx]
```

```
#To reduce the matrix size, you may want to try
# a code similar to the one below.

freq_dist = FreqDist(tokens)
vocab = []
for word in tokens:
     if not stop_words(word) and freq_dist[word] > 20:
        vocab.append(word)
```