Assignment 4
Submit by 08-April-2024
Total Marks - 30

1 Instructions

Please note the following guidelines for submitting your assignments:

1. Use Python to code your assignments 😃.
2. Use Colab to submit your assignments. Learn all the tricks of Colab from the web, especially how to read a remote file.
3. Share the final version of your assignments only with the following email IDs:
   (a) ramaseshan.ta@gmail.com
   (b) meryanhere2@gmail.com
   (c) aditimishra555551@gmail.com
   (d) vardhan.vkr@gmail.com

   Assignment sent to any other address will not be evaluated
4. We will NOT run/change your Python notebook.
5. Naming Conventions for the assignments:
   (a) The first part of the filename should be your First name.
   (b) The second part of the filename should be your roll number.
   (c) The third part of your assignment should be Assignment0X, where X is the assignment number.

   Example: `SriramBMC202204_Assignment01.ipynb`
6. We will not evaluate files with arbitrary names.
7. DO NOT send the file as an attachment to the email IDs.
8. Make sure that all the results are available when you share them. Incomplete Python notebooks **will not be evaluated.**
9. **Optional**: You may use Github to store your versions of the assignment. Advantages - You will never lose your code if you check-in the code into the GitHub repository after changes regularly.

Be warned:
This assignment takes a lot of time to run. Start early to avoid any issues

2 Assignment 4 (A4) - Word Embedding - Implementation

2.1 Overview

This assignment focuses on the concept of word embeddings, which is a key aspect of Natural Language Processing (NLP). The task at hand involves working with Hindi textual data and computing word embeddings utilizing Positive Point-wise Mutual Information (PPMI) (refer to section 2.5). The resulting embeddings will be analyzed to comprehend the semantic relationships between words.

2.2 Corpus creation - 5 marks

To accomplish this task, you will be using a Hindi Wikipedia dump as the text corpus. You may additionally try other language texts as well, if you want to test your algorithm efficiency in capturing the semantic relationship. The links to the wiki dumps for various languages are given below:

- Bengali
- Hindi
- Kannada
- Malyalam
- Marati
- Tamil
- Telugu
- English

For any other language, replace XX with the ISO code of the language of your choice. Make sure that the language you choose has at least around 300,000 articles. A sample code for extracting the wiki dump is given in the section 3.1.

2.3 Preprocessing - 5 marks

Browse the content to check if preprocessing required- For example, removal of =, Foreign words, stop words (check the attached file), numbers, etc. You may use regular expression library, `import re`, to perform preprocessing operations.

2.4 Counting Tokens and Building Vocabulary - 5 marks

- Find the total number of tokens before and after preprocessing (2.5 marks)
- Find the vocabulary and its count (2.5 marks)
  Note: 5 marks will be awarded only if you show the output in Colab.

For counting operations you may use collections module (`from collections import Counter`)

2.5 Co-occurrence Matrix Creation - 10 marks

1. Initialize a matrix with dimensions (vocabulary size, vocabulary size). This represents word co-occurrence counts. Refer to section 3.3 for two different methods to create the co-occurrence matrix.
2. Run through the entire corpus using a ramped window of size 5

target word	$word_1$	$word_2$	$word_3$	$word_4$	$word_5$
0	5	4	3	2	1

3. For each word (w_j) within the window of w_i, increment the corresponding cell (i, j) in the co-occurrence matrix by using the weight. This captures how often words co-occur.
4. Use the following to convert the elements of the matrix using Positive PMI as follows:

$$\textbf{Probability:} p_{ij} = p(w_j \mid w_i) = \frac{a_{ij}}{a_i}$$

$$\textbf{Positive Point-wise Mutual Information(PPMI)} : ppmi(w_i, w_j) = max\left(\log_2\left(\frac{p_{ij}}{p_i * p_j}\right), 0\right)$$

where a_{ij} is the co-occurrence count of word i co-occurring with word j and a_i is the count of word i co-occurring with any other word in the vocabulary.

Be warned:
This assignment takes a lot of time to run. Start early to avoid any issues

You may either use `numpy` module to create the matrix or file-based `H5py`. `H5py` is a Python library that allows yooneu to work with the HDF5 data format. HDF5 (Hierarchical Data Format version 5) is a popular format for storing large datasets, running to several terabytes. Use `import h5py`. If `h5py` is unavailable in Colab, use `!pip install h5py` to install it. Check the sample code section (3.3) for the usage.

2.6 Word Embeddings

Each row of the co-occurrence matrix (having PPMI as elements) can be considered as a word vector, where elements represent the word's relationship with other words in the vocabulary.

2.7 Similar words - 5 marks

Select a list of popular Hindi 10 nouns (by frequency count) from the vocabulary. For each noun, use the cosine distance function ($1 - \cos\theta$) to compute the nearest neighbors and store them in a dictionary using the module `collections`. Use the `most_common` function from the `collections` library to arrange the similar words in the descending order of cosine distance. **Print the top 10 similar words for every chosen noun**.

3 Sample Code

3.1 Corpus Creation

```
from wiki_dump_reader import Cleaner, iterate

def write_corpus():
    corpus_file = 'CorpusFileName.txt'
    page_count = 0
    cleaner = Cleaner()
    #with open(corpus_file, 'w', encoding='utf-8') as output:
    for title, text in iterate('your wikidump file in XML format'):
        text = cleaner.clean_text(text)
        cleaned_text, links = cleaner.build_links(text)
        #output.write(title + '\n' + cleaned_text + '\n')
        page_count += 1
        if page_count % 1000 == 0:
            print(f'Pages dumped = {page_count}', end='\r')

            print(title + '\n' + cleaned_text + '\n')
    #output.close()
    print(f"\npage count = {page_count}")
write_corpus()
```

3.2 Ngrams

To capture ngrams to fill a window of size 5, use the following code:

```
from nltk import word_tokenize, ngrams
#replace the text with the corpus created earlier
text = "This assignment (4) focuses on the concept of word embeddings!"
tokens = word_tokenize(text)
grams = ngrams(tokens,2)
for gram in grams:
    print(gram)
```

3.3 Co-occurrence Matrix using numpy or h5py

You may use `numpy` or `h5py` to create the matrix (initialized with zero) as follows:

1. Numpy

```
import numpy as np
word_vectors = np.zeros((vocab_size, vocab_size), dtype=np.float16)
```

2. h5py

```
import h5py
fd = h5py.File('mat_cooccur.hdf5', 'w')
co_occur_matrix = f.create_dataset("cooccur", (vocab_count, vocab_count), dtype=np.float16,compression='gzip')
#You may access the co_occur_matrix using the integer indexes
co_occur_matrix[vocab_index[targer_word_index], vocab_index[context_word_index] += weight
```

Be warned:
This assignment takes a lot of time to run. Start early to avoid any issues