I081_Aniruddh_Kulkarni_NLP_Exp7

May 28, 2023

1 Name: Aniruddh Kulkarni

2 Roll no: I081

3 Stream: CS (AI)

4 Division: I

5 Semester: 5th Semester

6 Batch: I-3

7 Subject: NLP

8 Assignment-7

[1]:
```
!pip install scikit-learn==0.21.3
!pip install wget==3.2
!pip install gensim==3.6.0
!pip install psutil==5.4.8
!pip install spacy==2.2.4
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting scikit-learn==0.21.3
  Downloading scikit_learn-0.21.3-cp37-cp37m-manylinux1_x86_64.whl (6.7 MB)
     |                  | 6.7 MB 5.0 MB/s
Requirement already satisfied: scipy>=0.17.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn==0.21.3) (1.7.3)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn==0.21.3) (1.21.6)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn==0.21.3) (1.2.0)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.0.2
    Uninstalling scikit-learn-1.0.2:
      Successfully uninstalled scikit-learn-1.0.2
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
yellowbrick 1.5 requires scikit-learn>=1.0.0, but you have scikit-learn 0.21.3
which is incompatible.
imbalanced-learn 0.8.1 requires scikit-learn>=0.24, but you have scikit-learn
0.21.3 which is incompatible.
Successfully installed scikit-learn-0.21.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting wget==3.2
  Downloading wget-3.2.zip (10 kB)
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) … done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9675
sha256=553a12a34018e829d2f6b34a597d2cd5ed0fb2824f18b8640e2d54c296ba7beb
  Stored in directory: /root/.cache/pip/wheels/a1/b6/7c/0e63e34eb06634181c63adac
ca38b79ff8f35c37e3c13e3c02
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: gensim==3.6.0 in /usr/local/lib/python3.7/dist-
packages (3.6.0)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.7/dist-
packages (from gensim==3.6.0) (1.21.6)
Requirement already satisfied: smart-open>=1.2.1 in
/usr/local/lib/python3.7/dist-packages (from gensim==3.6.0) (5.2.1)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.7/dist-
packages (from gensim==3.6.0) (1.15.0)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-
packages (from gensim==3.6.0) (1.7.3)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: psutil==5.4.8 in /usr/local/lib/python3.7/dist-
packages (5.4.8)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting spacy==2.2.4
  Downloading spacy-2.2.4-cp37-cp37m-manylinux1_x86_64.whl (10.6 MB)
     |                  | 10.6 MB 5.3 MB/s
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (4.64.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (2.0.6)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (2.23.0)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (0.10.1)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (3.0.7)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (1.0.8)
Collecting blis<0.5.0,>=0.4.0
  Downloading blis-0.4.1-cp37-cp37m-manylinux1_x86_64.whl (3.7 MB)
     |                  | 3.7 MB 33.7 MB/s
Requirement already satisfied: setuptools in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (57.4.0)
Collecting plac<1.2.0,>=0.9.6
  Downloading plac-1.1.3-py2.py3-none-any.whl (20 kB)
Collecting catalogue<1.1.0,>=0.0.7
  Downloading catalogue-1.0.1-py2.py3-none-any.whl (16 kB)
Collecting srsly<1.1.0,>=1.0.2
  Downloading srsly-1.0.5-cp37-cp37m-manylinux2014_x86_64.whl (184 kB)
     |                  | 184 kB 60.2 MB/s
Collecting thinc==7.4.0
  Downloading thinc-7.4.0-cp37-cp37m-manylinux1_x86_64.whl (2.2 MB)
     |                  | 2.2 MB 45.6 MB/s
Requirement already satisfied: numpy>=1.15.0 in
/usr/local/lib/python3.7/dist-packages (from spacy==2.2.4) (1.21.6)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from
catalogue<1.1.0,>=0.0.7->spacy==2.2.4) (4.1.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from catalogue<1.1.0,>=0.0.7->spacy==2.2.4) (3.9.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from
requests<3.0.0,>=2.13.0->spacy==2.2.4) (2022.9.24)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from
requests<3.0.0,>=2.13.0->spacy==2.2.4) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests<3.0.0,>=2.13.0->spacy==2.2.4) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from
requests<3.0.0,>=2.13.0->spacy==2.2.4) (1.24.3)
Installing collected packages: srsly, plac, catalogue, blis, thinc, spacy
  Attempting uninstall: srsly
    Found existing installation: srsly 2.4.4
    Uninstalling srsly-2.4.4:
      Successfully uninstalled srsly-2.4.4
  Attempting uninstall: catalogue
    Found existing installation: catalogue 2.0.8
    Uninstalling catalogue-2.0.8:
      Successfully uninstalled catalogue-2.0.8
  Attempting uninstall: blis
    Found existing installation: blis 0.7.8
    Uninstalling blis-0.7.8:
      Successfully uninstalled blis-0.7.8
  Attempting uninstall: thinc
    Found existing installation: thinc 8.1.3
    Uninstalling thinc-8.1.3:
      Successfully uninstalled thinc-8.1.3
  Attempting uninstall: spacy
    Found existing installation: spacy 3.4.1
    Uninstalling spacy-3.4.1:
      Successfully uninstalled spacy-3.4.1
ERROR: pip's dependency resolver does not currently take into account all

the packages that are installed. This behaviour is the source of the following

dependency conflicts.

en-core-web-sm 3.4.0 requires spacy<3.5.0,>=3.4.0, but you have spacy 2.2.4

which is incompatible.

confection 0.0.3 requires srsly<3.0.0,>=2.4.0, but you have srsly 1.0.5 which is

incompatible.
Successfully installed blis-0.4.1 catalogue-1.0.1 plac-1.1.3 spacy-2.2.4
srsly-1.0.5 thinc-7.4.0
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[3]:
```
import os
import wget
import gzip
import shutil

gn_vec_path = "GoogleNews-vectors-negative300.bin"
if not os.path.exists("GoogleNews-vectors-negative300.bin"):
    if not os.path.exists("/content/drive/MyDrive/
 ↪GoogleNews-vectors-negative300.bin.gz"):
        #Downloading the reqired model
        if not os.path.exists("/content/drive/MyDrive/
 ↪GoogleNews-vectors-negative300.bin.gz"):
```

```
            if not os.path.exists("GoogleNews-vectors-negative300.bin.gz"):
                wget.download("https://s3.amazonaws.com/dl4j-distribution/
 ↪GoogleNews-vectors-negative300.bin.gz")
            gn_vec_zip_path = "GoogleNews-vectors-negative300.bin.gz"
        else:
            gn_vec_zip_path = "/content/drive/MyDrive/
 ↪GoogleNews-vectors-negative300.bin.gz"
        #Extracting the required model
        with gzip.open(gn_vec_zip_path, 'rb') as f_in:
            with open(gn_vec_path, 'wb') as f_out:
                shutil.copyfileobj(f_in, f_out)
    else:
        gn_vec_path = "../Ch2/" + gn_vec_path

print(f"Model at {gn_vec_path}")
```

```
Model at ../Ch2/GoogleNews-vectors-negative300.bin
```

[4]:
```
!pip install wget
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: wget in /usr/local/lib/python3.7/dist-packages
(3.2)
```

[5]:
```
!free -h
```

```
              total        used        free      shared  buff/cache   available
Mem:            12G        584M        9.6G        1.2M        2.6G         11G
Swap:            0B          0B          0B
```

[6]:
```
!nvidia-smi -L
```

```
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver.
Make sure that the latest NVIDIA driver is installed and running.
```

[7]:
```
!lscpu |grep 'Model name'
```

```
Model name:          Intel(R) Xeon(R) CPU @ 2.20GHz
```

[4]:
```
import warnings #This module ignores the various types of warnings generated
warnings.filterwarnings("ignore")

import psutil #This module helps in retrieving information on running processes␣
 ↪and system resource utilization
process = psutil.Process(os.getpid())
from psutil import virtual_memory
```

```
mem = virtual_memory()

import time #This module is used to calculate the time
```

[5]:
```
import os
import psutil #This module helps in retrieving information on running processes␣
  ↪and system resource utilization
process = psutil.Process(os.getpid())
from psutil import virtual_memory
mem = virtual_memory()

import time #This module is used to calculate the time
from gensim.models import Word2Vec, KeyedVectors
pretrainedpath = "/content/drive/MyDrive/GoogleNews-vectors-negative300.bin.gz"

#Load W2V model. This will take some time, but it is a one time effort!
pre = process.memory_info().rss
print("Memory used in GB before Loading the Model: %0.2f"%float(pre/(10**9)))␣
  ↪#Check memory usage before loading the model
print('-'*10)

start_time = time.time() #Start the timer
ttl = mem.total #Toal memory available

w2v_model = KeyedVectors.load_word2vec_format(pretrainedpath, binary=True)␣
  ↪#load the model
print("%0.2f seconds taken to load"%float(time.time() - start_time)) #Calculate␣
  ↪the total time elapsed since starting the timer
print('-'*10)

print('Finished loading Word2Vec')
print('-'*10)

post = process.memory_info().rss
print("Memory used in GB after Loading the Model: {:.2f}".format(float(post/
  ↪(10**9)))) #Calculate the memory used after loading the model
print('-'*10)

print("Percentage increase in memory usage: {:.2f}% ".format(float((post/
  ↪pre)*100))) #Percentage increase in memory after loading the model
print('-'*10)

print("Number of words in vocabulary: ",len(w2v_model.vocab)) #Number of words␣
  ↪in the vocabulary.
```

```
Memory used in GB before Loading the Model: 0.15
----------
```

```
140.71 seconds taken to load
----------
Finished loading Word2Vec
----------
Memory used in GB after Loading the Model: 5.09
----------
Percentage increase in memory usage: 3366.03%
----------
Number of words in vocabulary:  3000000
```

[ ]:
```
#Let us examine the model by knowing what the most similar words are, for a␣
 ↪given word!
w2v_model.most_similar('beautiful')
```

[ ]:
```
[('gorgeous', 0.8353004455566406),
 ('lovely', 0.810693621635437),
 ('stunningly_beautiful', 0.7329413890838623),
 ('breathtakingly_beautiful', 0.7231341004371643),
 ('wonderful', 0.6854087114334106),
 ('fabulous', 0.6700063943862915),
 ('loveliest', 0.6612576246261597),
 ('prettiest', 0.6595001816749573),
 ('beatiful', 0.6593326330184937),
 ('magnificent', 0.6591402292251587)]
```

[ ]:
```
#What is the vector representation for a word?
w2v_model['computer']
```

[ ]:
```
array([ 1.07421875e-01, -2.01171875e-01,  1.23046875e-01,  2.11914062e-01,
       -9.13085938e-02,  2.16796875e-01, -1.31835938e-01,  8.30078125e-02,
        2.02148438e-01,  4.78515625e-02,  3.66210938e-02, -2.45361328e-02,
        2.39257812e-02, -1.60156250e-01, -2.61230469e-02,  9.71679688e-02,
       -6.34765625e-02,  1.84570312e-01,  1.70898438e-01, -1.63085938e-01,
       -1.09375000e-01,  1.49414062e-01, -4.65393066e-04,  9.61914062e-02,
        1.68945312e-01,  2.60925293e-03,  8.93554688e-02,  6.49414062e-02,
        3.56445312e-02, -6.93359375e-02, -1.46484375e-01, -1.21093750e-01,
       -2.27539062e-01,  2.45361328e-02, -1.24511719e-01, -3.18359375e-01,
       -2.20703125e-01,  1.30859375e-01,  3.66210938e-02, -3.63769531e-02,
       -1.13281250e-01,  1.95312500e-01,  9.76562500e-02,  1.26953125e-01,
        6.59179688e-02,  6.93359375e-02,  1.02539062e-02,  1.75781250e-01,
       -1.68945312e-01,  1.21307373e-03, -2.98828125e-01, -1.15234375e-01,
        5.66406250e-02, -1.77734375e-01, -2.08984375e-01,  1.76757812e-01,
        2.38037109e-02, -2.57812500e-01, -4.46777344e-02,  1.88476562e-01,
        5.51757812e-02,  5.02929688e-02, -1.06933594e-01,  1.89453125e-01,
       -1.16210938e-01,  8.49609375e-02, -1.71875000e-01,  2.45117188e-01,
       -1.73828125e-01, -8.30078125e-03,  4.56542969e-02, -1.61132812e-02,
        1.86523438e-01, -6.05468750e-02, -4.17480469e-02,  1.82617188e-01,
        2.20703125e-01, -1.22558594e-01, -2.55126953e-02, -3.08593750e-01,
        9.13085938e-02,  1.60156250e-01,  1.70898438e-01,  1.19628906e-01,
        7.08007812e-02, -2.64892578e-02, -3.08837891e-02,  4.06250000e-01,
       -1.01562500e-01,  5.71289062e-02, -7.26318359e-03, -9.17968750e-02,
       -1.50390625e-01, -2.55859375e-01,  2.16796875e-01, -3.63769531e-02,
        2.24609375e-01,  8.00781250e-02,  1.56250000e-01,  5.27343750e-02,
        1.50390625e-01, -1.14746094e-01, -8.64257812e-02,  1.19140625e-01,
       -7.17773438e-02,  2.73437500e-01, -1.64062500e-01,  7.29370117e-03,
        4.21875000e-01, -1.12792969e-01, -1.35742188e-01, -1.31835938e-01,
       -1.37695312e-01, -7.66601562e-02,  6.25000000e-02,  4.98046875e-02,
       -1.91406250e-01, -6.03027344e-02,  2.27539062e-01,  5.88378906e-02,
       -3.24218750e-01,  5.41992188e-02, -1.35742188e-01,  8.17871094e-03,
       -5.24902344e-02, -1.74713135e-03, -9.81445312e-02, -2.86865234e-02,
        3.61328125e-02,  2.15820312e-01,  5.98144531e-02, -3.08593750e-01,
       -2.27539062e-01,  2.61718750e-01,  9.86328125e-02, -5.07812500e-02,
        1.78222656e-02,  1.31835938e-01, -5.35156250e-01, -1.81640625e-01,
        1.38671875e-01, -3.10546875e-01, -9.71679688e-02,  1.31835938e-01,
       -1.16210938e-01,  7.03125000e-02,  2.85156250e-01,  3.51562500e-02,
       -1.01562500e-01, -3.75976562e-02,  1.41601562e-01,  1.42578125e-01,
       -5.68847656e-02,  2.65625000e-01, -2.09960938e-01,  9.64355469e-03,
       -6.68945312e-02, -4.83398438e-02, -6.10351562e-02,  2.45117188e-01,
       -9.66796875e-02,  1.78222656e-02, -1.27929688e-01, -4.78515625e-02,
       -7.26318359e-03,  1.79687500e-01,  2.78320312e-02, -2.10937500e-01,
       -1.43554688e-01, -1.27929688e-01,  1.73339844e-02, -3.60107422e-03,
       -2.04101562e-01,  3.63159180e-03, -1.19628906e-01, -6.15234375e-02,
        5.93261719e-02, -3.23486328e-03, -1.70898438e-01, -3.14941406e-02,
       -8.88671875e-02, -2.89062500e-01,  3.44238281e-02, -1.87500000e-01,
        2.94921875e-01,  1.58203125e-01, -1.19628906e-01,  7.61718750e-02,
        6.39648438e-02, -4.68750000e-02, -6.83593750e-02,  1.21459961e-02,
       -1.44531250e-01,  4.54101562e-02,  3.68652344e-02,  3.88671875e-01,
        1.45507812e-01, -2.55859375e-01, -4.46777344e-02, -1.33789062e-01,
       -1.38671875e-01,  6.59179688e-02,  1.37695312e-01,  1.14746094e-01,
        2.03125000e-01, -4.78515625e-02,  1.80664062e-02, -8.54492188e-02,
       -2.48046875e-01, -3.39843750e-01, -2.83203125e-02,  1.05468750e-01,
       -2.14843750e-01, -8.74023438e-02,  7.12890625e-02,  1.87500000e-01,
       -1.12304688e-01,  2.73437500e-01, -3.26171875e-01, -1.77734375e-01,
       -4.24804688e-02, -2.69531250e-01,  6.64062500e-02, -6.88476562e-02,
       -1.99218750e-01, -7.03125000e-02, -2.43164062e-01, -3.66210938e-02,
       -7.37304688e-02, -1.77734375e-01,  9.17968750e-02, -1.25000000e-01,
       -1.65039062e-01, -3.57421875e-01, -2.85156250e-01, -1.66992188e-01,
        1.97265625e-01, -1.53320312e-01,  2.31933594e-02,  2.06054688e-01,
        1.80664062e-01, -2.74658203e-02, -1.92382812e-01, -9.61914062e-02,
       -1.06811523e-02, -4.73632812e-02,  6.54296875e-02, -1.25732422e-02,
        1.78222656e-02, -8.00781250e-02, -2.59765625e-01,  9.37500000e-02,
       -7.81250000e-02,  4.68750000e-02, -2.22167969e-02,  1.86767578e-02,
        3.11279297e-02,  1.04980469e-02, -1.69921875e-01,  2.58789062e-02,
       -3.41796875e-02, -1.44042969e-02, -5.46875000e-02, -8.78906250e-02,
                1.96838379e-03,  2.23632812e-01, -1.36718750e-01,  1.75781250e-01,
               -1.63085938e-01,  1.87500000e-01,  3.44238281e-02, -5.63964844e-02,
               -2.27689743e-05,  4.27246094e-02,  5.81054688e-02, -1.07910156e-01,
               -3.88183594e-02, -2.69531250e-01,  3.34472656e-02,  9.81445312e-02,
                5.63964844e-02,  2.23632812e-01, -5.49316406e-02,  1.46484375e-01,
                5.93261719e-02, -2.19726562e-01,  6.39648438e-02,  1.66015625e-02,
                4.56542969e-02,  3.26171875e-01, -3.80859375e-01,  1.70898438e-01,
                5.66406250e-02, -1.04492188e-01,  1.38671875e-01, -1.57226562e-01,
                3.23486328e-03, -4.80957031e-02, -2.48046875e-01, -6.20117188e-02],
              dtype=float32)
```

[ ]:
```
#What if I am looking for a word that is not in this vocabulary?
w2v_model['practicalnlp']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-12-354849ef77a2> in <module>
      1 #What if I am looking for a word that is not in this vocabulary?
----> 2 w2v_model['practicalnlp']

/usr/local/lib/python3.7/dist-packages/gensim/models/keyedvectors.py in␣
 ↪__getitem__(self, entities)
    335         if isinstance(entities, string_types):
    336             # allow calls like trained_model['office'], as a shorthand␣
 ↪for trained_model[['office']]
--> 337             return self.get_vector(entities)
    338 
    339         return vstack([self.get_vector(entity) for entity in entities])

/usr/local/lib/python3.7/dist-packages/gensim/models/keyedvectors.py in␣
 ↪get_vector(self, word)
    453 
    454     def get_vector(self, word):
--> 455         return self.word_vec(word)
    456 
    457     def words_closer_than(self, w1, w2):

/usr/local/lib/python3.7/dist-packages/gensim/models/keyedvectors.py in␣
 ↪word_vec(self, word, use_norm)
    450             return result
    451         else:
--> 452             raise KeyError("word '%s' not in vocabulary" % word)
    453 
    454     def get_vector(self, word):

KeyError: "word 'practicalnlp' not in vocabulary"
```

```
[ ]: !python -m spacy download en_core_web_md
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting en_core_web_md==2.2.5
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_md-2.2.5/en_core_web_md-2.2.5.tar.gz (96.4
MB)
     |                 | 96.4 MB 1.2 MB/s
Requirement already satisfied: spacy>=2.2.2 in
/usr/local/lib/python3.7/dist-packages (from en_core_web_md==2.2.5) (2.2.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-
packages (from spacy>=2.2.2->en_core_web_md==2.2.5) (57.4.0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (3.0.7)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (0.10.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (1.0.8)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (0.4.1)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (1.0.5)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (1.1.3)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (2.23.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (4.64.1)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (1.0.1)
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/python3.7/dist-
packages (from spacy>=2.2.2->en_core_web_md==2.2.5) (7.4.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from
spacy>=2.2.2->en_core_web_md==2.2.5) (2.0.6)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/dist-
packages (from spacy>=2.2.2->en_core_web_md==2.2.5) (1.21.6)
```

```
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from catalogue<1.1.0,>=0.0.7->spacy>=2.2.2->en_core_web_md==2.2.5)
(3.9.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from
catalogue<1.1.0,>=0.0.7->spacy>=2.2.2->en_core_web_md==2.2.5) (4.1.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests<3.0.0,>=2.13.0->spacy>=2.2.2->en_core_web_md==2.2.5)
(2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->en_core_web_md==2.2.5) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->en_core_web_md==2.2.5) (2022.9.24)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->en_core_web_md==2.2.5) (1.24.3)
Building wheels for collected packages: en-core-web-md
  Building wheel for en-core-web-md (setup.py) … done
  Created wheel for en-core-web-md: filename=en_core_web_md-2.2.5-py3-none-
any.whl size=98051301
sha256=97f2bcbc937069ae4fe7256fe96e369569387c9e98d635b02245d969b30525ee
  Stored in directory: /tmp/pip-ephem-wheel-cache-
lhd6kydn/wheels/69/c5/b8/4f1c029d89238734311b3269762ab2ee325a42da2ce8edb997
Successfully built en-core-web-md
Installing collected packages: en-core-web-md
Successfully installed en-core-web-md-2.2.5
 Download and installation successful
You can now load the model via spacy.load('en_core_web_md')
```

[ ]:
```python
import spacy

%time
nlp = spacy.load('en_core_web_md')
# process a sentence using the model
mydoc = nlp("Canada is a large country")
#Get a vector for individual words
#print(doc[0].vector) #vector for 'Canada', the first word in the text
print(mydoc.vector) #Averaged vector for the entire sentence
```

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 7.63 µs
[-1.12055197e-01  2.26087615e-01 -5.15111461e-02 -1.21812008e-01
  4.13958639e-01 -8.56475979e-02 -2.84600933e-03 -2.26096585e-01
  6.98113963e-02  2.27946019e+00 -4.49774921e-01 -6.39050007e-02
 -1.80326015e-01 -8.79765972e-02  9.93399299e-04 -1.57384202e-01
 -1.23817801e-01  1.54990411e+00  2.00794004e-02  1.38399601e-01
```

```
-1.48897991e-01 -2.23025799e-01 -1.48171991e-01  4.68924567e-02
-3.17026004e-02  1.19096041e-02 -6.10985979e-02  9.57068056e-02
 9.37099904e-02  1.70955807e-01 -9.29740071e-03  7.88536817e-02
 1.74508005e-01 -1.04450598e-01  1.04872189e-01 -1.16961405e-01
 6.23028055e-02 -2.23016590e-01 -1.44107476e-01 -2.03423887e-01
 2.61404991e-01  2.43404001e-01  1.51980996e-01 -1.12484001e-01
 1.18055798e-01 -9.51323956e-02  8.66319984e-02 -2.54322797e-01
 3.84932049e-02  1.18278004e-01 -3.21602583e-01  3.73764008e-01
 1.13018408e-01 -8.05834010e-02  1.84921592e-01  9.38879885e-03
 1.22166201e-01 -3.24288011e-02  1.01590000e-01 -1.56877995e-01
-2.57006437e-02  1.63392588e-01  1.06118001e-01  2.25193188e-01
 8.06204006e-02 -1.21081993e-01 -1.52107209e-01  8.25726017e-02
-6.09899946e-02  1.44145802e-01  2.01554038e-02  2.54258011e-02
 1.06071997e-02  6.37948066e-02  1.10551611e-01 -6.40176088e-02
-6.36451989e-02 -9.99798030e-02 -7.01020136e-02  3.09334368e-01
 5.68300001e-02  3.63879651e-03 -1.65255398e-01  2.98442870e-01
 4.01660334e-03 -1.73631594e-01  5.15965708e-02  1.61811799e-01
 2.20304996e-01 -8.29614028e-02 -2.64678001e-01  2.44114012e-01
 3.48895532e-03 -1.57521993e-01  1.67974800e-01  1.05541132e-01
-1.31224409e-01  7.17941970e-02  1.39708191e-01 -1.95359858e-03
-8.55428055e-02  1.20119795e-01 -6.84404075e-02  5.14601183e-04
-2.86250003e-02 -1.10662603e+00  2.02491835e-01 -1.50410801e-01
 6.51507173e-03 -3.30360234e-03  1.21523812e-01 -1.61614027e-02
-1.43233404e-01 -9.88139957e-02 -2.17486005e-02  1.81988999e-01
 8.85506049e-02  2.72242010e-01 -7.73219988e-02  1.43622067e-02
-1.57062009e-01  4.01146002e-02  3.90305184e-02 -1.42812401e-01
-2.08329991e-01  9.64459926e-02  1.42821997e-01 -1.94155991e-01
 5.37982993e-02 -1.00471973e-02  1.94714032e-02 -9.83514041e-02
-4.17162031e-02  1.23069003e-01  1.68428212e-01 -1.17991492e-01
-2.56704390e-01 -1.89464003e-01  9.22677964e-02 -1.72503412e-01
-1.11929202e+00  6.42500073e-03  3.51435989e-01  8.19059983e-02
 4.92408946e-02 -1.80243999e-01  1.82863399e-01  8.92240033e-02
 2.47399211e-01  2.74492018e-02 -2.49322020e-02  2.35055804e-01
 8.12319964e-02 -1.86482631e-02 -1.06439434e-01  5.28851971e-02
-1.02569997e-01  1.35777995e-01 -2.32603997e-01  9.24602076e-02
 1.92440599e-01  1.48551196e-01  5.57186007e-02  3.97088043e-02
-6.74048066e-03  9.73687991e-02  2.62231939e-02 -8.26509967e-02
 1.30085424e-01 -1.38572007e-01 -4.11808006e-02 -4.13070023e-02
-3.41880023e-02  1.28202796e-01 -6.66912049e-02 -7.41944537e-02
-5.87003939e-02  1.36300415e-01  1.67494014e-01  1.71119809e-01
 1.18692197e-01  2.30142009e-02 -2.06086040e-02 -3.85930002e-01
-1.17673976e-02 -7.34595209e-02 -3.43096368e-02 -7.80718103e-02
-2.81003956e-02 -7.30765983e-02 -2.21649408e-01 -1.02057599e-01
 5.11020012e-02 -9.07440037e-02 -4.69896048e-02 -2.10200553e-03
 1.05816983e-01  4.79107983e-02  1.03080198e-01 -8.96641985e-02
 8.85651931e-02 -9.09178331e-02 -5.16167991e-02  1.50742605e-01
 3.07500213e-01 -4.05239780e-03  1.04269005e-01  3.55780013e-02
 1.16165996e-01 -2.97939777e-03 -1.42966792e-01  5.00957891e-02
-1.08308598e-01  1.68199837e-03  1.36314392e-01  1.48694202e-01
-3.17817986e-01  1.21000603e-01 -1.59556001e-01  7.51644000e-02
-1.03386201e-01  1.10754207e-01  8.20529982e-02 -6.02059904e-03
 1.35578603e-01 -4.08943966e-02  6.05328009e-02  1.03734590e-01
-6.22724071e-02  2.30276197e-01  1.30762011e-01  1.51950002e-01
 7.40183964e-02 -1.84507206e-01 -1.33174613e-01 -1.49338007e-01
 1.19309977e-01 -2.41554022e-01 -1.00904807e-01  1.54562384e-01
-7.63845369e-02  1.66379198e-01  2.20374197e-01  1.58361979e-02
 1.80677801e-01 -1.77342609e-01  2.22857997e-01 -2.99477577e-01
-1.53620601e-01 -2.67919600e-01  1.56353399e-01 -1.74718007e-01
 1.83644608e-01  1.28259212e-01 -6.30084053e-02  2.68236816e-01
 2.10368007e-01 -4.73994762e-02 -1.09680817e-01  1.62620202e-01
 8.96113962e-02  1.50361210e-01 -1.55037967e-02  1.50141995e-02
 1.76618043e-02 -2.28057191e-01  7.85290003e-02 -4.59632799e-02
 1.98103897e-02 -1.71379801e-02 -1.45676598e-01 -3.32076550e-02
-2.09102005e-01 -2.48584002e-01 -8.51256028e-02  4.25900035e-02
-1.33966401e-01  2.89979968e-02  2.10713193e-01 -1.86206046e-02
 1.71603993e-01  2.21868396e-01 -2.10479975e-01  1.49794608e-01
-1.10692397e-01 -4.47340589e-03  5.13652042e-02 -7.27116019e-02
 6.07413948e-02 -8.13369974e-02 -1.94639400e-01 -5.06809242e-02
 6.40980080e-02 -2.20814198e-01  2.96969917e-02  1.53438210e-01
-2.18270030e-02 -1.93358198e-01 -2.26220220e-01  1.84093148e-01]
```

#Exercises

Programming Assignment 7

1. Use a pre-trained word embedding model and find out the embedding of some random words and their most similar words .
2. Create a toy corpus having four sentences with maximum 5 words into each sentence. Peform Text cleaning and preprocessing and use gensim to create a CBoW and Skipgram model and then trained them on your toy corpus. Observe the embeddings
3. Create a corpus by taking a raw text. Peform required text cleaning and preprocessing and use gensim to create a CBoW and Skipgram model and then trained them your corpus. Observe the embeddings

[ ]:
```
#TASK-1
```

[ ]:
```
w2v_model.most_similar('smart')
```

[ ]:
```
[('intelligent', 0.6495277881622314),
 ('dumb', 0.5792694687843323),
 ('smartest', 0.5717369318008423),
 ('savvy', 0.5674132108688354),
 ('clever', 0.5656732320785522),
 ('smarter', 0.5632593631744385),
 ('shrewd', 0.5591170191764832),
 ('IPICO_produces', 0.5158056020736694),
 ('economic_certainty_Kracmer', 0.5094711184501648),
 ('Smart', 0.4991162121295929)]
```

[ ]:
```
w2v_model['smart']
```

[ ]:
```
array([ 1.04492188e-01,  1.37939453e-02, -6.03027344e-02,  9.96093750e-02,
       -1.47460938e-01, -5.61523438e-02, -4.93164062e-02,  9.58251953e-03,
       -6.53076172e-03, -9.52148438e-02, -3.44848633e-03, -1.26953125e-01,
       -9.47265625e-02, -2.12890625e-01,  6.83593750e-02, -1.34765625e-01,
        4.56542969e-02,  4.93164062e-02, -2.16796875e-01,  2.01416016e-03,
        1.97753906e-02, -1.79443359e-02,  1.30859375e-01,  1.12792969e-01,
        1.11816406e-01,  2.51953125e-01, -1.21093750e-01,  1.40625000e-01,
       -9.86328125e-02, -2.33398438e-01, -8.39843750e-02,  1.66015625e-01,
        1.55273438e-01,  1.12304688e-01,  2.87109375e-01,  6.64062500e-02,
       -6.07910156e-02,  1.69677734e-02, -4.15039062e-02,  7.42187500e-02,
        1.62109375e-01, -1.81640625e-01,  3.08593750e-01, -5.83496094e-02,
       -3.10058594e-02,  9.86328125e-02, -1.17675781e-01, -2.71484375e-01,
        7.22656250e-02, -5.37109375e-02, -4.73632812e-02,  1.54296875e-01,
       -5.81054688e-02, -4.54101562e-02,  1.17301941e-04,  3.20312500e-01,
       -7.51953125e-02, -1.74804688e-01,  1.43554688e-01,  9.76562500e-03,
       -1.89453125e-01, -5.27343750e-02, -3.30078125e-01,  7.51953125e-02,
        2.08740234e-02,  1.15234375e-01, -1.77734375e-01,  2.71484375e-01,
       -8.34960938e-02,  2.89306641e-02,  1.69921875e-01,  9.22851562e-02,
        2.63671875e-01, -7.47070312e-02, -2.15820312e-01,  3.34472656e-02,
        1.11328125e-01,  3.43750000e-01, -4.88281250e-02,  2.44140625e-01,
       -8.54492188e-02,  2.37304688e-01, -3.24707031e-02,  6.78710938e-02,
        2.83203125e-01, -2.67578125e-01,  5.44433594e-02,  1.03515625e-01,
        1.28906250e-01,  6.03027344e-02, -1.31835938e-01,  1.81640625e-01,
       -3.02734375e-01, -1.14257812e-01,  1.20605469e-01, -1.83593750e-01,
       -1.65039062e-01,  2.92968750e-01, -8.88671875e-02, -1.16699219e-01,
       -1.54296875e-01, -4.15039062e-02, -5.61523438e-02,  1.29882812e-01,
        1.36718750e-01, -1.81640625e-01, -1.87500000e-01, -1.80664062e-01,
        2.16064453e-02, -1.30859375e-01, -4.95605469e-02, -1.60156250e-01,
       -2.02148438e-01,  2.97851562e-02,  2.51953125e-01, -3.08837891e-02,
        4.88281250e-02, -8.72802734e-03,  8.59375000e-02, -1.69921875e-01,
       -1.89453125e-01,  1.25976562e-01, -1.08886719e-01,  1.19628906e-01,
       -6.07910156e-02, -2.47070312e-01, -8.15429688e-02,  1.22558594e-01,
        2.92968750e-02, -2.35595703e-02, -1.24023438e-01, -2.00195312e-01,
       -1.13281250e-01,  3.58886719e-02, -5.34667969e-02,  7.51953125e-02,
        2.23632812e-01,  3.02734375e-01, -4.54101562e-02,  1.28906250e-01,
        2.34985352e-03, -2.85156250e-01,  1.07421875e-02,  3.36914062e-02,
       -8.23974609e-03,  1.04492188e-01, -2.69775391e-02, -3.32031250e-02,
        1.60156250e-01, -1.22070312e-01, -1.10839844e-01, -8.25195312e-02,
        6.25000000e-02,  1.63085938e-01, -2.05078125e-01, -5.17578125e-02,
        1.75781250e-01,  9.47265625e-02,  7.91015625e-02, -1.69921875e-01,
       -1.19140625e-01,  1.20605469e-01,  1.36718750e-01, -1.51367188e-01,
       -1.24511719e-02,  5.29785156e-02,  1.38671875e-01, -1.78710938e-01,
        8.34960938e-02, -1.84570312e-01, -2.55859375e-01, -1.68945312e-01,
        -4.05883789e-03, -2.08984375e-01,  7.14111328e-03,  2.34375000e-01,
         2.10937500e-01,  8.72802734e-03, -1.98242188e-01,  2.10937500e-01,
         7.08007812e-02, -2.83203125e-01, -3.92578125e-01,  1.04522705e-03,
        -1.17187500e-01, -1.06933594e-01, -2.41210938e-01, -4.29687500e-02,
        -2.27539062e-01, -1.05468750e-01, -5.32226562e-02,  2.16796875e-01,
        -7.56835938e-02, -2.87109375e-01, -6.83593750e-02,  1.14746094e-01,
        -1.78710938e-01, -1.26953125e-01, -1.64062500e-01,  7.81250000e-02,
        -4.34570312e-02,  1.69921875e-01,  1.85546875e-01,  2.96875000e-01,
        -1.06445312e-01, -2.15820312e-01,  4.85839844e-02, -1.51367188e-01,
        -7.37304688e-02, -1.19140625e-01, -1.34765625e-01,  2.79296875e-01,
         7.47070312e-02, -5.15136719e-02, -3.70788574e-03,  2.63671875e-01,
         1.06933594e-01,  1.26953125e-01, -3.00781250e-01,  3.16406250e-01,
        -2.26562500e-01,  1.25976562e-01, -1.86523438e-01, -8.05664062e-02,
        -1.65039062e-01,  1.17187500e-01,  5.49316406e-02, -2.85156250e-01,
        -3.32641602e-03, -1.35742188e-01,  1.22558594e-01,  3.32031250e-02,
         9.27734375e-02, -1.04003906e-01,  2.98828125e-01, -8.34960938e-02,
        -2.27539062e-01, -7.76367188e-02,  6.39648438e-02, -9.91210938e-02,
         2.21679688e-01,  1.16210938e-01, -6.93359375e-02,  1.75781250e-01,
        -4.78515625e-02,  9.27734375e-02, -5.17578125e-02,  2.09960938e-02,
        -8.39843750e-02, -2.02148438e-01, -2.04101562e-01,  1.11816406e-01,
         8.59375000e-02,  2.09960938e-01,  1.94335938e-01, -1.61132812e-01,
         1.04370117e-02, -1.17675781e-01, -2.59765625e-01, -7.37304688e-02,
         6.05468750e-02,  3.21960449e-03, -2.81250000e-01, -1.62109375e-01,
        -5.12695312e-02,  1.23535156e-01, -6.83593750e-02, -3.45703125e-01,
        -5.02929688e-02,  4.73632812e-02,  2.06298828e-02, -4.17480469e-02,
         5.17578125e-02,  1.40625000e-01,  4.54101562e-02,  6.39648438e-02,
         1.00097656e-01, -4.37011719e-02,  9.61914062e-02,  2.83203125e-01,
         1.69921875e-01,  1.51367188e-01,  8.34960938e-02, -2.03857422e-02,
        -1.31835938e-01,  2.64892578e-02, -1.92382812e-01, -1.08398438e-01,
         1.40625000e-01, -1.25122070e-02, -5.83496094e-02,  1.41601562e-01,
        -2.04101562e-01, -2.01171875e-01, -2.18750000e-01,  1.09252930e-02,
         1.85546875e-01, -4.07714844e-02, -1.30859375e-01,  1.79687500e-01],
       dtype=float32)
```

[ ]:
```python
w2v_model.most_similar('hot')
```

[ ]:
```
[('Hot', 0.6659734845161438),
 ('hottest', 0.6050904989242554),
 ('hotter', 0.5794501900672913),
 ('CHEFS_Chefs', 0.5725089907646179),
 ('sizzling', 0.5456805229187012),
 ('scorching', 0.5294975638389587),
 ('teardrop_shaped_VTA', 0.5230512619018555),
 ('cool', 0.5151149034500122),
 ('briquette_iron', 0.5066832304000854),
 ('heated', 0.5010007619857788)]
```

```
[ ]: w2v_model['hot']
```

```
[ ]: array([-0.00750732,  0.00817871,  0.07373047,  0.0859375 , -0.04418945,
       -0.13769531,  0.09619141, -0.20019531, -0.20214844,  0.19433594,
        0.03710938, -0.15429688,  0.13085938, -0.08203125, -0.10058594,
        0.15625   ,  0.05908203, -0.18261719,  0.06347656, -0.20410156,
        0.01843262,  0.13671875, -0.10205078, -0.09667969, -0.05493164,
        0.10107422,  0.03198242,  0.09960938, -0.1484375 ,  0.04174805,
        0.08007812,  0.171875  ,  0.22460938, -0.12695312, -0.22460938,
       -0.12207031, -0.11816406,  0.02868652,  0.14453125, -0.07324219,
        0.06079102, -0.41210938,  0.02819824, -0.1015625 ,  0.015625  ,
       -0.20605469,  0.109375  , -0.07421875,  0.20703125,  0.08544922,
        0.13574219,  0.421875  , -0.07763672,  0.1953125 , -0.25390625,
        0.0546875 ,  0.30664062, -0.03173828,  0.09814453,  0.03015137,
       -0.09716797,  0.00543213, -0.26367188, -0.09375   ,  0.14355469,
        0.04150391, -0.3125    , -0.50390625,  0.00127411,  0.04956055,
        0.30273438,  0.10009766,  0.22753906,  0.20410156, -0.19140625,
       -0.12353516,  0.21875   ,  0.11669922, -0.16601562,  0.35546875,
        0.03466797, -0.02819824,  0.23242188,  0.00326538, -0.15917969,
        0.15527344, -0.16796875,  0.14648438, -0.02404785,  0.05273438,
       -0.10888672,  0.08837891, -0.16015625, -0.12695312, -0.09472656,
        0.11181641,  0.18945312, -0.12597656,  0.27148438,  0.02893066,
       -0.09472656, -0.14746094, -0.09912109, -0.03637695, -0.04003906,
       -0.27148438,  0.11083984, -0.19335938,  0.00909424, -0.08300781,
        0.09326172, -0.16992188, -0.10058594, -0.1640625 ,  0.20117188,
        0.00622559,  0.2578125 ,  0.02746582,  0.00476074, -0.12792969,
       -0.19921875, -0.20410156, -0.02355957, -0.21777344, -0.28515625,
       -0.01361084, -0.06152344,  0.29882812,  0.03637695, -0.17871094,
        0.13476562, -0.06054688,  0.07177734,  0.10107422, -0.11474609,
       -0.01037598,  0.28515625, -0.21289062,  0.11962891, -0.07763672,
        0.06884766,  0.18554688,  0.28320312, -0.11767578,  0.25      ,
       -0.02416992,  0.05419922, -0.07666016, -0.05444336, -0.16992188,
        0.13964844,  0.04272461, -0.14453125,  0.20996094, -0.20605469,
       -0.02966309, -0.06396484, -0.24804688, -0.07080078, -0.22753906,
       -0.07226562,  0.04663086,  0.01989746, -0.0112915 ,  0.06054688,
        0.203125  , -0.04394531,  0.02709961, -0.03491211, -0.17480469,
       -0.21972656,  0.04003906,  0.11279297,  0.04443359, -0.22265625,
        0.04907227, -0.10498047,  0.04589844, -0.4296875 ,  0.02001953,
        0.09033203, -0.30664062,  0.03833008, -0.10302734,  0.13574219,
       -0.09716797, -0.07958984,  0.17773438, -0.10107422,  0.203125  ,
        0.05810547,  0.07910156,  0.13085938,  0.14355469, -0.19042969,
        0.40234375, -0.24023438, -0.13476562, -0.05126953, -0.10449219,
        0.1953125 ,  0.05834961, -0.03930664, -0.18261719,  0.02990723,
       -0.15332031, -0.05029297, -0.19042969,  0.00183868,  0.02392578,
       -0.36328125,  0.14453125,  0.19042969, -0.02819824,  0.31835938,
        0.03857422,  0.25976562, -0.06884766, -0.18359375, -0.10253906,
       -0.14746094,  0.16601562, -0.0859375 ,  0.17675781,  0.10253906,
         0.12353516,  0.10888672, -0.19335938, -0.03076172, -0.15332031,
        -0.12451172,  0.11035156, -0.09765625,  0.05615234, -0.12402344,
        -0.06494141, -0.02868652, -0.21777344,  0.09863281,  0.24804688,
         0.30273438,  0.11816406, -0.04980469, -0.05859375,  0.03759766,
         0.02258301,  0.10644531, -0.0078125 ,  0.29101562, -0.06835938,
        -0.14257812,  0.16113281,  0.15527344,  0.02880859,  0.15625   ,
        -0.11865234,  0.13574219, -0.11328125, -0.02453613, -0.24609375,
         0.08251953,  0.00448608, -0.15625   ,  0.18359375,  0.13867188,
        -0.03662109, -0.03686523, -0.11523438,  0.1484375 , -0.11035156,
         0.17480469,  0.15625   ,  0.07421875,  0.04956055, -0.07714844,
         0.08837891, -0.02966309, -0.25585938, -0.14941406,  0.24023438,
         0.44726562,  0.13476562,  0.17675781,  0.18261719,  0.08886719,
         0.04858398, -0.12988281,  0.21386719, -0.00202942, -0.24316406,
         0.29492188, -0.06298828,  0.125     ,  0.15039062,  0.1796875 ,
        -0.25195312, -0.07226562,  0.0324707 , -0.11816406,  0.07763672],
       dtype=float32)
```

[ ]:
```
w2v_model.most_similar('sweet')
```

[ ]:
```
[('sweetness', 0.6216812133789062),
 ('sweetest', 0.6207044720649719),
 ('caramelly', 0.6009937524795532),
 ('syrupy_sweet', 0.5979651808738708),
 ('yummy', 0.5963824391365051),
 ('buttery', 0.593994140625),
 ('tooth_achingly', 0.5892406105995178),
 ('fan_Rosangela_Pereira', 0.5891238451004028),
 ('delicious', 0.5866374969482422),
 ('fruity', 0.5804054737091064)]
```

[ ]:
```
w2v_model['sweet']
```

[ ]:
```
array([ 0.08691406,  0.0300293 , -0.04589844,  0.11767578,  0.02490234,
        -0.07421875,  0.31640625, -0.00331116, -0.07226562,  0.23925781,
        -0.2734375 , -0.1640625 , -0.24121094, -0.0201416 ,  0.06176758,
         0.2890625 , -0.06298828, -0.01599121, -0.01635742, -0.06787109,
        -0.06591797,  0.25976562,  0.10351562, -0.05224609,  0.11035156,
        -0.13671875, -0.11621094, -0.06640625, -0.03442383,  0.10058594,
         0.03271484,  0.12988281,  0.140625  ,  0.17773438, -0.19335938,
        -0.13378906,  0.05175781, -0.13574219,  0.02490234,  0.07910156,
         0.19628906, -0.3203125 ,  0.09423828, -0.02258301,  0.07519531,
        -0.24804688,  0.09179688,  0.24707031,  0.11621094,  0.10449219,
        -0.17871094,  0.2734375 ,  0.19335938,  0.18652344,  0.01953125,
         0.15917969, -0.04516602, -0.06494141, -0.109375  , -0.21582031,
        -0.31835938,  0.12597656, -0.21972656,  0.15039062, -0.00408936,
        -0.06591797, -0.02087402, -0.3515625 ,  0.07666016,  0.26757812,
         0.22363281, -0.11865234,  0.01123047,  0.0703125 , -0.05541992,
         0.05664062, -0.11132812,  0.11181641,  0.12695312,  0.37109375,
        -0.11181641,  0.14453125, -0.06030273, -0.07861328, -0.08007812,
        -0.20898438, -0.0625    ,  0.2421875 , -0.09667969,  0.1953125 ,
        -0.04125977,  0.08984375, -0.23046875,  0.13964844, -0.13183594,
         0.02648926, -0.03466797,  0.01464844, -0.20410156, -0.21582031,
        -0.22753906,  0.02111816, -0.05175781, -0.07373047,  0.10742188,
        -0.06152344,  0.17578125, -0.09130859,  0.01843262,  0.00787354,
        -0.13769531,  0.12890625, -0.23339844, -0.07910156,  0.37304688,
         0.13867188,  0.06225586,  0.09570312,  0.05810547, -0.02832031,
         0.08154297,  0.07226562, -0.05395508, -0.22558594, -0.31054688,
         0.14355469,  0.03295898,  0.11669922,  0.30664062, -0.21484375,
        -0.20507812,  0.23828125,  0.09082031,  0.06494141, -0.22167969,
         0.15722656,  0.15820312,  0.14453125,  0.09814453,  0.12109375,
         0.05786133, -0.0168457 , -0.03564453, -0.17382812, -0.27148438,
         0.04370117, -0.03637695,  0.08154297, -0.31054688, -0.19921875,
         0.05761719,  0.07128906, -0.10791016,  0.09765625, -0.28710938,
        -0.03344727,  0.03051758,  0.09423828, -0.3359375 , -0.05200195,
        -0.07617188,  0.12109375,  0.13183594,  0.04370117,  0.14941406,
        -0.10009766,  0.06542969, -0.08349609, -0.10986328,  0.02856445,
        -0.02282715,  0.00701904,  0.08154297, -0.18652344, -0.19824219,
         0.11132812,  0.078125  ,  0.0480957 , -0.17285156,  0.15722656,
         0.01843262, -0.23144531, -0.02038574,  0.09570312, -0.16699219,
         0.03393555, -0.15039062,  0.1171875 , -0.06591797,  0.12695312,
        -0.10546875,  0.01489258,  0.05639648, -0.23925781, -0.05053711,
         0.48242188, -0.1796875 , -0.13476562,  0.0267334 ,  0.12011719,
         0.00297546, -0.09863281, -0.08496094,  0.01422119, -0.01831055,
        -0.17578125, -0.25      , -0.01672363, -0.11621094,  0.08740234,
        -0.40625   ,  0.05883789, -0.14746094,  0.10449219,  0.02612305,
         0.00445557,  0.03344727, -0.04541016, -0.05151367,  0.13085938,
        -0.03833008,  0.37109375, -0.02209473, -0.1328125 ,  0.03442383,
         0.15136719, -0.11865234, -0.20214844, -0.03112793,  0.18945312,
        -0.11865234, -0.1015625 , -0.04296875,  0.34765625,  0.11816406,
        -0.08740234, -0.3359375 , -0.140625  , -0.07421875,  0.10693359,
         0.50390625, -0.00933838, -0.14550781, -0.17773438,  0.03417969,
         0.24902344,  0.05737305,  0.15234375, -0.03662109, -0.09912109,
        -0.15625   , -0.01300049,  0.31054688,  0.25390625, -0.04980469,
        -0.2421875 , -0.07958984, -0.26367188, -0.12011719, -0.13183594,
        -0.13867188,  0.01513672, -0.40820312,  0.171875  ,  0.06884766,
        -0.28515625, -0.17773438, -0.12988281, -0.07470703, -0.02990723,
         0.265625  ,  0.09912109,  0.05200195,  0.1015625 ,  0.0291748 ,
        -0.01940918, -0.05200195,  0.04345703, -0.22851562,  0.1796875 ,
         0.078125  , -0.11669922, -0.10400391,  0.19140625,  0.15332031,
        -0.15722656, -0.15332031, -0.11328125, -0.31445312,  0.16601562,
        -0.02600098,  0.03369141,  0.00288391, -0.3046875 , -0.10986328,
        -0.06396484,  0.14648438, -0.02001953,  0.0559082 ,  0.10986328],
       dtype=float32)
```

```
[ ]: w2v_model.most_similar('friend')
```

```
[ ]: [('pal', 0.7476358413696289),
      ('friends', 0.7098034620285034),
      ('buddy', 0.6972494125366211),
      ('dear_friend', 0.6960037350654602),
      ('acquaintance', 0.6843010187149048),
      ('cousin', 0.6713200807571411),
      ('girlfriend', 0.6226294040679932),
      ('colleague', 0.6204894185066223),
      ('uncle', 0.6120923161506653),
      ('roommate', 0.611858069896698)]
```

```
[ ]: w2v_model['friend']
```

```
[ ]: array([ 0.07080078, -0.21386719,  0.15332031,  0.09423828, -0.03442383,
             0.43359375, -0.16503906, -0.05786133,  0.17578125, -0.08203125,
             0.24511719, -0.19335938, -0.0255127 , -0.09619141, -0.125     ,
             0.02575684,  0.16796875, -0.03759766,  0.09472656, -0.04760742,
             0.20605469,  0.31835938,  0.15917969, -0.17089844,  0.09033203,
            -0.1640625 , -0.15234375,  0.3125    ,  0.06298828, -0.24902344,
             0.15625   , -0.04516602, -0.12890625, -0.00686646, -0.02160645,
             0.14453125,  0.2734375 ,  0.12695312,  0.10742188,  0.11376953,
             0.14355469, -0.00173187,  0.22851562, -0.03515625,  0.17089844,
             0.04516602, -0.07958984, -0.08886719, -0.01342773, -0.09667969,
            -0.12597656,  0.10595703,  0.15332031, -0.03808594,  0.02246094,
             0.01428223, -0.03295898,  0.20703125, -0.03417969,  0.02233887,
             0.00244141,  0.13476562, -0.01403809,  0.13378906,  0.0201416 ,
             0.14746094,  0.00759888, -0.18652344,  0.16113281,  0.109375  ,
             0.14355469,  0.01623535,  0.01867676,  0.09179688, -0.33789062,
             0.19335938, -0.29101562, -0.00860596,  0.10644531,  0.359375  ,
             0.25585938, -0.03320312,  0.15625   , -0.24316406, -0.06738281,
             0.09033203, -0.125     ,  0.21777344, -0.02380371, -0.06445312,
            -0.14355469,  0.05664062, -0.12597656,  0.02172852,  0.03833008,
            -0.17578125, -0.08349609,  0.21386719, -0.01855469, -0.23535156,
            -0.14746094, -0.16113281, -0.03125   , -0.10107422,  0.07080078,
             0.01135254, -0.04370117,  0.07666016,  0.16503906,  0.04541016,
            -0.13867188,  0.13085938,  0.13378906, -0.14453125,  0.12792969,
            -0.06787109, -0.04296875, -0.03369141,  0.10302734,  0.22949219,
             0.14160156, -0.01153564, -0.00086212, -0.10449219, -0.03710938,
             0.01928711,  0.16699219, -0.06079102,  0.09814453,  0.0703125 ,
            -0.39648438, -0.23242188, -0.04077148,  0.09570312, -0.0546875 ,
            -0.09814453,  0.09082031,  0.03588867,  0.09228516,  0.3125    ,
             0.10595703,  0.18847656, -0.11230469,  0.00842285,  0.08935547,
             0.04663086, -0.25      , -0.03369141,  0.03808594, -0.03710938,
             0.42773438,  0.10839844, -0.01391602, -0.01965332, -0.04296875,
            -0.11035156,  0.0390625 ,  0.04541016, -0.20019531, -0.14355469,
            -0.14257812,  0.03662109,  0.25      ,  0.3671875 , -0.12304688,
            -0.0859375 ,  0.24902344, -0.21582031,  0.02648926,  0.17871094,
             0.29296875,  0.21582031,  0.1015625 ,  0.00167084, -0.07177734,
             0.03686523,  0.22851562, -0.125     ,  0.17285156,  0.22265625,
             0.21191406,  0.03686523,  0.09570312, -0.00344849,  0.13183594,
            -0.23925781,  0.00576782,  0.27148438,  0.10400391,  0.0098877 ,
            -0.24511719,  0.21777344, -0.03027344,  0.23046875,  0.11816406,
             0.1640625 , -0.00109863,  0.00349426, -0.02197266, -0.09179688,
            -0.10351562,  0.06933594, -0.13476562, -0.06201172,  0.14355469,
            -0.10888672, -0.11328125,  0.2109375 , -0.10839844, -0.18261719,
            -0.06689453, -0.265625  , -0.13378906, -0.04296875, -0.17773438,
             0.00689697, -0.00982666, -0.00640869, -0.12792969,  0.08203125,
            -0.01367188,  0.02734375,  0.12597656, -0.00772095, -0.04614258,
            -0.12255859,  0.16210938,  0.28320312,  0.04296875, -0.05175781,
            -0.16210938,  0.14648438, -0.18359375, -0.24511719,  0.22167969,
             0.0546875 , -0.10302734, -0.07763672, -0.33984375, -0.05908203,
            -0.0022583 , -0.11962891, -0.3046875 ,  0.02233887,  0.02941895,
             0.37695312, -0.01721191, -0.05932617,  0.30273438, -0.13574219,
             0.14746094,  0.17089844,  0.16015625,  0.21484375,  0.01013184,
             0.06738281, -0.12109375, -0.12304688, -0.20117188,  0.02880859,
            -0.00662231, -0.20410156,  0.02001953, -0.15136719,  0.16699219,
             0.14160156, -0.02331543,  0.14550781, -0.13476562,  0.04785156,
             0.14160156,  0.03808594, -0.12109375,  0.02770996, -0.0123291 ,
            -0.20410156, -0.06445312,  0.06079102, -0.07519531, -0.28125   ,
             0.18261719, -0.25390625, -0.0456543 ,  0.14160156, -0.0546875 ,
            -0.01477051, -0.38085938,  0.14355469,  0.12255859,  0.14941406,
            -0.03320312,  0.19433594, -0.34375   , -0.24902344, -0.00331116,
            -0.05639648, -0.00079727, -0.21679688, -0.01977539,  0.10644531],
           dtype=float32)
```

9 TASK-2

[6]:
```python
sent1 = 'Hello this is Millie'
sent2 = 'Millie and Shreyas are friends'
sent3 = 'Millie likes Physics'
sent4 = 'Shreyas likes NLP'

data = [sent1.split(), sent2.split(), sent3.split(), sent4.split()]
values = data[0]+data[1]+data[2]+data[3]
our_data = values
print(values)
print("The data: ",values)
```

```
['Hello', 'this', 'is', 'Millie', 'Millie', 'and', 'Shreyas', 'are', 'friends',
'Millie', 'likes', 'Physics', 'Shreyas', 'likes', 'NLP']
The data:  ['Hello', 'this', 'is', 'Millie', 'Millie', 'and', 'Shreyas', 'are',
'friends', 'Millie', 'likes', 'Physics', 'Shreyas', 'likes', 'NLP']
```

[7]:
```
import re
import pandas as pd
import nltk
import string
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')


def clean(doc): # doc is a string of text
    #doc = [x.replace("</br>", " ") for x in doc] # This text contains a lot of␣
 ↪<br/> tags.      [x.replace("</br>", " ") for x in doc] doc.replace("</br>",␣
 ↪" ")
    #doc = "".join([char for char in doc if char not in string.punctuation and␣
 ↪not char.isdigit()])
    #doc = " ".join([token for token in doc.split() if token not in stopwords])
    # remove punctuation and numbers
    #initialize stopwords
    stop_words_nltk = set(stopwords.words('english'))
    #print(stop_words_nltk)

    #stopword removal
    corpus2 = [i for i in doc if not i in stop_words_nltk]
    print("Tokenized corpus without stopwords:",corpus2)
    return corpus2
our_data = clean(our_data)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/feature_extraction/image.py:167:
DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To
silence this warning, use `int` by itself. Doing this will not modify any
behavior and is safe. When replacing `np.int`, you may wish to use e.g.
`np.int64` or `np.int32` to specify the precision. If you wish to review your
current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  dtype=np.int):
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:30:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  method='lar', copy_X=True, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:167:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
```

```
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  method='lar', copy_X=True, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:284:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  eps=np.finfo(np.float).eps, copy_Gram=True, verbose=0,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:862:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  eps=np.finfo(np.float).eps, copy_X=True, fit_path=True,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1101:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  eps=np.finfo(np.float).eps, copy_X=True, fit_path=True,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1127:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  eps=np.finfo(np.float).eps, positive=False):
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1362:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  max_n_alphas=1000, n_jobs=None, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1602:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
```

```
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  max_n_alphas=1000, n_jobs=None, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1738:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  eps=np.finfo(np.float).eps, copy_X=True, positive=False):
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.

Tokenized corpus without stopwords: ['Hello', 'Millie', 'Millie', 'Shreyas',
'friends', 'Millie', 'likes', 'Physics', 'Shreyas', 'likes', 'NLP']
```

[8]:
```
def extractL(lst):
    return list(map(lambda el:[el], lst))

our_data = extractL(our_data)
our_data
```

[8]:
```
[['Hello'],
 ['Millie'],
 ['Millie'],
 ['Shreyas'],
 ['friends'],
 ['Millie'],
 ['likes'],
 ['Physics'],
 ['Shreyas'],
 ['likes'],
 ['NLP']]
```

[9]:
```
model = Word2Vec(our_data, min_count=1,size= 50,workers=3, window =3, sg = 0)␣
 ↪#sg=0 default CBOW for skipgram set to 1

model2 = Word2Vec(our_data, min_count=1,size= 50,workers=5, window =2, sg = 1)
```

```
WARNING:gensim.models.base_any2vec:consider setting layer size to a multiple of
4 for greater performance
WARNING:gensim.models.base_any2vec:under 10 jobs per worker: consider setting a
```

```
smaller `batch_words' for smoother alpha decay
WARNING:gensim.models.base_any2vec:consider setting layer size to a multiple of
4 for greater performance
WARNING:gensim.models.base_any2vec:under 10 jobs per worker: consider setting a
smaller `batch_words' for smoother alpha decay
```

```
[ ]: model['NLP']
```

```
[ ]: array([ 1.3146516e-03,  8.7559475e-03, -2.5728724e-03,  1.1237229e-04,
            1.8875578e-03, -5.0069867e-03,  6.7687416e-03, -9.9992510e-03,
           -4.1280923e-04, -2.2862162e-03, -9.7010909e-03,  6.9062519e-03,
            9.5691029e-03, -3.0840512e-03,  4.4608414e-03, -1.7764017e-03,
           -8.2435962e-03, -3.2646020e-03, -9.9094855e-03, -9.4692912e-03,
            8.7401075e-03,  2.2220907e-03, -8.7813307e-03, -4.3473379e-03,
           -3.9097221e-04,  4.8672589e-03,  1.5777374e-03, -2.6316158e-04,
           -8.8995704e-03,  9.8497635e-03,  4.2200121e-03,  3.4971384e-03,
            2.8517384e-03,  8.2129147e-03,  8.1832148e-03,  1.3322410e-03,
            1.2316037e-05,  5.6195138e-03,  2.0136524e-03, -4.8742563e-04,
           -7.2859200e-03, -9.2956088e-03, -6.6227694e-03,  1.2034532e-03,
            5.1734657e-03,  5.3558438e-03,  6.4542238e-03, -5.8507882e-03,
            2.3829192e-03,  1.3914639e-03], dtype=float32)
```

```
[ ]: model['Hello']
```

```
[ ]: array([-7.7724608e-04, -8.9907041e-03,  7.3977425e-03, -7.7399369e-03,
            1.3372888e-03,  8.3685564e-03, -5.5546416e-03, -6.6069402e-03,
           -5.7461751e-03,  9.4375963e-04, -9.0449052e-03,  3.1865919e-03,
            3.0637851e-03, -6.6448567e-03,  8.2862442e-03,  6.0003200e-03,
            5.0570504e-03, -8.4179537e-03, -2.0104467e-03, -5.4620481e-03,
           -4.0113931e-03, -2.8337825e-05, -7.0015886e-03, -4.2749051e-04,
           -1.3846685e-03, -5.6160851e-03,  2.1336376e-04,  1.8454179e-03,
           -2.9860779e-03,  6.1147953e-03,  1.4553561e-03, -3.2206960e-03,
            5.7934150e-03, -2.4389478e-03,  2.5247803e-03, -5.2044098e-03,
           -4.4597303e-03,  4.4761668e-03, -8.7460503e-05, -5.0890315e-03,
            3.7272116e-03,  8.4023876e-03, -8.7595712e-03,  2.7472220e-04,
           -4.1439533e-03, -5.2641677e-03,  5.2119493e-03, -9.1280788e-03,
           -1.8064809e-03,  2.2712999e-03], dtype=float32)
```

```
[ ]: model['Millie']
```

```
[ ]: array([ 1.8134555e-05, -3.9992682e-04,  6.0865846e-03,  4.1675293e-03,
            3.6436340e-03, -8.8238623e-03, -8.3572650e-03,  5.8367532e-03,
            5.8944155e-03, -4.9712374e-03, -3.1488489e-03, -1.8000426e-03,
           -9.9649290e-03, -5.7536447e-03, -2.1811598e-03,  8.4818210e-03,
           -6.0961382e-03,  1.5411370e-03, -4.9108048e-03,  7.9836808e-03,
            2.2749444e-03,  6.8851812e-03,  6.2341075e-03, -4.4021336e-03,
            5.3096451e-03,  1.8955993e-03,  2.5525915e-03, -9.6495869e-03,
        3.0002154e-03,  9.3029039e-03,  7.9389010e-03, -8.5610524e-03,
       -4.6501551e-03, -7.5360546e-03,  5.1105209e-04, -7.3738475e-03,
       -7.4059195e-03, -2.8706687e-03, -7.4814255e-03,  3.2218655e-03,
        8.9386152e-03, -5.9583876e-03,  7.4548036e-04, -4.1315001e-03,
       -9.9905375e-03,  1.0565391e-03, -2.9333830e-03, -7.6596471e-03,
        3.9764252e-03,  2.0230883e-03], dtype=float32)
```

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

[ ]:
```
[('likes', 0.04230295866727829),
 ('Hello', 0.0013784740585833788),
 ('Physics', -0.0656752660870552),
 ('friends', -0.08807007968425751),
 ('NLP', -0.11610457301139832),
 ('Shreyas', -0.17405158281326294)]
```

[ ]:
```
model['Shreyas']
```

[ ]:
```
array([-3.1007377e-03,  8.8250246e-03, -1.7164573e-03, -2.8489414e-04,
       -6.6798795e-03,  4.2225556e-03,  1.2480994e-03, -8.7552397e-03,
       -1.9373518e-03,  2.5600396e-04,  8.9714229e-03,  5.1522618e-03,
        9.8919717e-04,  6.7502185e-04, -9.8543344e-03,  3.4552913e-03,
       -2.5372670e-03, -6.0131126e-03,  4.7803996e-03, -5.5559492e-03,
        5.3254263e-03,  7.1089347e-03,  2.4110294e-04,  6.4840536e-03,
       -4.1129002e-03,  6.4601186e-03, -5.4510897e-03,  5.4066246e-03,
       -9.9154869e-03,  6.6665183e-03, -4.2058756e-03, -9.9315345e-03,
        1.2303403e-03,  2.4500587e-03, -4.2300988e-03, -7.2377077e-03,
        5.5619841e-03,  8.2554650e-03,  9.0061547e-03,  5.5886339e-03,
       -1.4477929e-03,  5.4206932e-03, -2.5495712e-05, -1.1280694e-06,
        8.3391694e-03, -1.1670056e-03, -4.5022862e-03, -4.2864438e-03,
        8.2748979e-03,  8.9964010e-03], dtype=float32)
```

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

[ ]:
```
[('friends', 0.2498396635055542),
 ('NLP', 0.18875345587730408),
 ('Physics', 0.05735952407121658),
 ('Hello', 0.016872897744178772),
 ('likes', 0.010980001650750637),
 ('Millie', -0.17405156791210175)]
```

[ ]:
```
model['NLP']
```

[ ]:
```
array([ 1.3146516e-03,  8.7559475e-03, -2.5728724e-03,  1.1237229e-04,
        1.8875578e-03, -5.0069867e-03,  6.7687416e-03, -9.9992510e-03,
       -4.1280923e-04, -2.2862162e-03, -9.7010909e-03,  6.9062519e-03,
        9.5691029e-03, -3.0840512e-03,  4.4608414e-03, -1.7764017e-03,
        -8.2435962e-03, -3.2646020e-03, -9.9094855e-03, -9.4692912e-03,
         8.7401075e-03,  2.2220907e-03, -8.7813307e-03, -4.3473379e-03,
        -3.9097221e-04,  4.8672589e-03,  1.5777374e-03, -2.6316158e-04,
        -8.8995704e-03,  9.8497635e-03,  4.2200121e-03,  3.4971384e-03,
         2.8517384e-03,  8.2129147e-03,  8.1832148e-03,  1.3322410e-03,
         1.2316037e-05,  5.6195138e-03,  2.0136524e-03, -4.8742563e-04,
        -7.2859200e-03, -9.2956088e-03, -6.6227694e-03,  1.2034532e-03,
         5.1734657e-03,  5.3558438e-03,  6.4542238e-03, -5.8507882e-03,
         2.3829192e-03,  1.3914639e-03], dtype=float32)
```

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

[ ]:
```
[('Shreyas', 0.18875345587730408),
 ('Hello', 0.1579628437757492),
 ('likes', 0.08836456388235092),
 ('friends', 0.06421137601137161),
 ('Millie', -0.11610454320907593),
 ('Physics', -0.33987927436828613)]
```

[18]:
```
temp2=our_data
res = [''.join(ele) for ele in temp2]
str1 =''
for ele in res:
        str1 += ele
str1
```

[18]:
```
'HelloMillieMillieShreyasfriendsMillielikesPhysicsShreyaslikesNLP'
```

[15]:
```
import numpy as np
def get_mean_vector(word2vec_model, words):
    # remove out-of-vocabulary words
    words = [word for word in words if word in word2vec_model]
    if len(words) >= 1:
        return np.mean(word2vec_model[words], axis=0)
    else:
        return []
```

[56]:
```
word_embedding_avg_collectivei = []   #Averaged vector for the entire sentence␣
 ↪for CBOW
for doc in our_data:
    vec = get_mean_vector(model, doc)
    word_embedding_avg_collectivei.append(vec)
    if len(vec) > 0:
      print('Dimension of vector for each word:',vec.shape)

word_embedding_avgi = np.mean(word_embedding_avg_collectivei, axis=0)
print()
print('Dimension of vector for overall corpus:',word_embedding_avgi.shape)

print('Vector for overall corpus:',word_embedding_avgi)
```

```
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)

Dimension of vector for overall corpus: (50,)
Vector for overall corpus: [-0.00042292 -0.00128124  0.0016713  -0.00100073
0.00264984 -0.00056274
 -0.0018508   0.00528235  0.00074241 -0.00030788  0.00358317  0.00211925
  0.00028895  0.00125559  0.00045029 -0.00221913 -0.00109371 -0.00282404
 -0.00254699 -0.00171106 -0.00148378 -0.00040002 -0.00417252  0.0016632
 -0.00242404 -0.00181534  0.00011485  0.00287269  0.00050896  0.00212136
  0.00023629  0.00192432  0.00266776 -0.00164606  0.00342413  0.00275134
  0.00014774 -0.00225084  0.0016815   0.00242732  0.00293793 -0.00170815
 -0.00655518 -0.00020772  0.003925   -0.00628107  0.00131902 -0.00062684
  0.00140303  0.00632593]
```

```
[ ]: #for skipgrams
```

```
[ ]: model2['NLP']
```

```
[ ]: array([ 1.3146516e-03,  8.7559475e-03, -2.5728724e-03,  1.1237229e-04,
         1.8875578e-03, -5.0069867e-03,  6.7687416e-03, -9.9992510e-03,
        -4.1280923e-04, -2.2862162e-03, -9.7010909e-03,  6.9062519e-03,
         9.5691029e-03, -3.0840512e-03,  4.4608414e-03, -1.7764017e-03,
        -8.2435962e-03, -3.2646020e-03, -9.9094855e-03, -9.4692912e-03,
         8.7401075e-03,  2.2220907e-03, -8.7813307e-03, -4.3473379e-03,
        -3.9097221e-04,  4.8672589e-03,  1.5777374e-03, -2.6316158e-04,
        -8.8995704e-03,  9.8497635e-03,  4.2200121e-03,  3.4971384e-03,
         2.8517384e-03,  8.2129147e-03,  8.1832148e-03,  1.3322410e-03,
         1.2316037e-05,  5.6195138e-03,  2.0136524e-03, -4.8742563e-04,
        -7.2859200e-03, -9.2956088e-03, -6.6227694e-03,  1.2034532e-03,
         5.1734657e-03,  5.3558438e-03,  6.4542238e-03, -5.8507882e-03,
         2.3829192e-03,  1.3914639e-03], dtype=float32)
```

```
[ ]: model2.most_similar('NLP')
[ ]: [('Shreyas', 0.18875345587730408),
      ('Hello', 0.1579628437757492),
      ('likes', 0.08836456388235092),
      ('friends', 0.06421137601137161),
      ('Millie', -0.11610454320907593),
      ('Physics', -0.33987927436828613)]
```

```
[ ]: model2['Shreyas']
```

```
[ ]: array([-3.1007377e-03,  8.8250246e-03, -1.7164573e-03, -2.8489414e-04,
            -6.6798795e-03,  4.2225556e-03,  1.2480994e-03, -8.7552397e-03,
            -1.9373518e-03,  2.5600396e-04,  8.9714229e-03,  5.1522618e-03,
             9.8919717e-04,  6.7502185e-04, -9.8543344e-03,  3.4552913e-03,
            -2.5372670e-03, -6.0131126e-03,  4.7803996e-03, -5.5559492e-03,
             5.3254263e-03,  7.1089347e-03,  2.4110294e-04,  6.4840536e-03,
            -4.1129002e-03,  6.4601186e-03, -5.4510897e-03,  5.4066246e-03,
            -9.9154869e-03,  6.6665183e-03, -4.2058756e-03, -9.9315345e-03,
             1.2303403e-03,  2.4500587e-03, -4.2300988e-03, -7.2377077e-03,
             5.5619841e-03,  8.2554650e-03,  9.0061547e-03,  5.5886339e-03,
            -1.4477929e-03,  5.4206932e-03, -2.5495712e-05, -1.1280694e-06,
             8.3391694e-03, -1.1670056e-03, -4.5022862e-03, -4.2864438e-03,
             8.2748979e-03,  8.9964010e-03], dtype=float32)
```

```
[ ]: model2.most_similar('Shreyas')
```

```
[ ]: [('friends', 0.2498396635055542),
      ('NLP', 0.18875345587730408),
      ('Physics', 0.05735952407121658),
      ('Hello', 0.016872897744178772),
      ('likes', 0.010980001650750637),
      ('Millie', -0.17405156791210175)]
```

```
[ ]: model2['Millie']
```

```
[ ]: array([ 1.8134555e-05, -3.9992682e-04,  6.0865846e-03,  4.1675293e-03,
             3.6436340e-03, -8.8238623e-03, -8.3572650e-03,  5.8367532e-03,
             5.8944155e-03, -4.9712374e-03, -3.1488489e-03, -1.8000426e-03,
            -9.9649290e-03, -5.7536447e-03, -2.1811598e-03,  8.4818210e-03,
            -6.0961382e-03,  1.5411370e-03, -4.9108048e-03,  7.9836808e-03,
             2.2749444e-03,  6.8851812e-03,  6.2341075e-03, -4.4021336e-03,
             5.3096451e-03,  1.8955993e-03,  2.5525915e-03, -9.6495869e-03,
             3.0002154e-03,  9.3029039e-03,  7.9389010e-03, -8.5610524e-03,
            -4.6501551e-03, -7.5360546e-03,  5.1105209e-04, -7.3738475e-03,
            -7.4059195e-03, -2.8706687e-03, -7.4814255e-03,  3.2218655e-03,
             8.9386152e-03, -5.9583876e-03,  7.4548036e-04, -4.1315001e-03,
            -9.9905375e-03,  1.0565391e-03, -2.9333830e-03, -7.6596471e-03,
             3.9764252e-03,  2.0230883e-03], dtype=float32)
```

[57]:
```
word_embedding_avg_collectiveii = []          #Averaged vector for the entire␣
 ↪sentence-skipgram
for doc in our_data:
    vec = get_mean_vector(model2, doc)
    word_embedding_avg_collectiveii.append(vec)
    if len(vec) > 0:
      print('Dimension of vector for each word:',vec.shape)

word_embedding_avgii = np.mean(word_embedding_avg_collectiveii, axis=0)
print()
print('Dimension of vector for overall corpus:',word_embedding_avgii.shape)

print('Vector for overall corpus:',word_embedding_avgii)
```

```
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)

Dimension of vector for overall corpus: (50,)
Vector for overall corpus: [-0.00042292 -0.00128124  0.0016713  -0.00100073
0.00264984 -0.00056274
 -0.0018508   0.00528235  0.00074241 -0.00030788  0.00358317  0.00211925
  0.00028895  0.00125559  0.00045029 -0.00221913 -0.00109371 -0.00282404
 -0.00254699 -0.00171106 -0.00148378 -0.00040002 -0.00417252  0.0016632
 -0.00242404 -0.00181534  0.00011485  0.00287269  0.00050896  0.00212136
  0.00023629  0.00192432  0.00266776 -0.00164606  0.00342413  0.00275134
  0.00014774 -0.00225084  0.0016815   0.00242732  0.00293793 -0.00170815
 -0.00655518 -0.00020772  0.003925   -0.00628107  0.00131902 -0.00062684
  0.00140303  0.00632593]
```

[46]:
```
#Doc2Vec
temp = our_data
temp2 = []
for i in temp:
    for j in i:
        temp2.append(j)
import gensim
docmodel = gensim.models.doc2vec.Doc2Vec(vector_size=50, min_count=2, epochs=40)
```

```
#building vocab
#docmodel.build_vocab([s.encode('utf-8').split() for s in temp2])


sentences = [gensim.models.doc2vec.TaggedDocument(sentence, 'tag') for sentence␣
 ↪in temp2]
docmodel.build_vocab(sentences)
```

```
WARNING:gensim.models.base_any2vec:consider setting layer size to a multiple of
4 for greater performance
WARNING:gensim.models.doc2vec:Each 'words' should be a list of words (usually
unicode strings). First 'words' here is instead plain <class 'str'>.
```

[44]:
```
#temp3 = []
#temp3.append(temp2)
print(temp3)
print(temp2)
```

```
[['Hello', 'Millie', 'Millie', 'Shreyas', 'friends', 'Millie', 'likes',
'Physics', 'Shreyas', 'likes', 'NLP']]
['Hello', 'Millie', 'Millie', 'Shreyas', 'friends', 'Millie', 'likes',
'Physics', 'Shreyas', 'likes', 'NLP']
```

[48]:
```
docmodel.train(sentences, total_examples=docmodel.corpus_count, epochs=docmodel.
 ↪epochs)
```

[58]:
```
vector = docmodel.infer_vector(temp2)
print(vector)                                #dim 50
```

```
[-0.00893469  0.00201269 -0.00633589 -0.00504674 -0.00998972 -0.00796403
 -0.00760187 -0.00552995  0.00727674 -0.00148645 -0.00219974 -0.00134809
  0.00432421 -0.00876796  0.00342459  0.00795252 -0.00072752  0.00848563
  0.00196946  0.00776178 -0.00220877  0.00096645  0.00977787 -0.00434421
 -0.00197775 -0.00749059  0.00766959  0.00336002  0.00862157  0.00630099
  0.00557941 -0.00040277  0.00151306  0.00681429  0.00199014  0.00178147
  0.00159094  0.00194351 -0.00194607  0.00822841 -0.0087912  -0.00631416
  0.00248155  0.0084832  -0.00275759  0.00308281 -0.00087546  0.00982781
  0.00596808 -0.00100695]
```

[61]:
```
from scipy import spatial

dataSetI = word_embedding_avgi
dataSetII = word_embedding_avgii                                               ␣
 ↪     #-1  value will indicate strongly opposite vectors
dataSetIII = vector                                                            ␣
 ↪     #0 independent (orthogonal) vectors
```

```
result1 = 1 - spatial.distance.cosine(word_embedding_avgi,␣
 ↪word_embedding_avgii)    #1 similar (positive co-linear) vectors.␣
 ↪Intermediate values are used to assess
result1                                                                  ␣
 ↪      #the degree of similarity.
```

[61]: 1

[62]:
```
result2 = 1 - spatial.distance.cosine(word_embedding_avgi, vector)
result2
```

[62]: -0.22039258480072021

[25]:
```
#TASK-3
corpusn = "NLP drives computer programs that translate text from one language␣
 ↪to another, respond to spoken commands, and summarize large volumes of text␣
 ↪rapidly-even in real time. There's a good chance you've interacted with NLP␣
 ↪in the form of voice-operated GPS systems, digital assistants,␣
 ↪speech-to-text dictation software, customer service chatbots, and other␣
 ↪consumer conveniences. But NLP also plays a growing role in enterprise␣
 ↪solutions that help streamline business operations, increase employee␣
 ↪productivity, and simplify mission-critical business processes."
values = corpusn.split()
stop_words_nltk = set(stopwords.words('english'))
values = [ i for i in values if not i in stop_words_nltk]

values
```

[25]:
```
['NLP',
 'drives',
 'computer',
 'programs',
 'translate',
 'text',
 'one',
 'language',
 'another,',
 'respond',
 'spoken',
 'commands,',
 'summarize',
 'large',
 'volumes',
 'text',
 'rapidly-even',
 'real',
 'time.',
 'There’s',
 'good',
 'chance',
 'you’ve',
 'interacted',
 'NLP',
 'form',
 'voice-operated',
 'GPS',
 'systems,',
 'digital',
 'assistants,',
 'speech-to-text',
 'dictation',
 'software,',
 'customer',
 'service',
 'chatbots,',
 'consumer',
 'conveniences.',
 'But',
 'NLP',
 'also',
 'plays',
 'growing',
 'role',
 'enterprise',
 'solutions',
 'help',
 'streamline',
 'business',
 'operations,',
 'increase',
 'employee',
 'productivity,',
 'simplify',
 'mission-critical',
 'business',
 'processes.']
```

[26]:
```
values = extractL(values)
```

[27]:
```
values
```

[27]:
```
[['NLP'],
 ['drives'],
 ['computer'],
 ['programs'],
 ['translate'],
 ['text'],
 ['one'],
 ['language'],
 ['another,'],
 ['respond'],
 ['spoken'],
 ['commands,'],
 ['summarize'],
 ['large'],
 ['volumes'],
 ['text'],
 ['rapidly-even'],
 ['real'],
 ['time.'],
 ['There’s'],
 ['good'],
 ['chance'],
 ['you’ve'],
 ['interacted'],
 ['NLP'],
 ['form'],
 ['voice-operated'],
 ['GPS'],
 ['systems,'],
 ['digital'],
 ['assistants,'],
 ['speech-to-text'],
 ['dictation'],
 ['software,'],
 ['customer'],
 ['service'],
 ['chatbots,'],
 ['consumer'],
 ['conveniences.'],
 ['But'],
 ['NLP'],
 ['also'],
 ['plays'],
 ['growing'],
 ['role'],
 ['enterprise'],
 ['solutions'],
 ['help'],
 ['streamline'],
 ['business'],
  ['operations,'],
  ['increase'],
  ['employee'],
  ['productivity,'],
  ['simplify'],
  ['mission-critical'],
  ['business'],
  ['processes.']]
```

[28]:
```
model3 = Word2Vec(values, min_count=1,size= 50,workers=3, window =3, sg = 0)␣
↪#sg=0 default CBOW for skipgram set to 1

model4 = Word2Vec(values, min_count=1,size= 50,workers=7, window =2, sg = 1)
```

```
WARNING:gensim.models.base_any2vec:consider setting layer size to a multiple of
4 for greater performance
WARNING:gensim.models.base_any2vec:under 10 jobs per worker: consider setting a
smaller `batch_words' for smoother alpha decay
WARNING:gensim.models.base_any2vec:consider setting layer size to a multiple of
4 for greater performance
WARNING:gensim.models.base_any2vec:under 10 jobs per worker: consider setting a
smaller `batch_words' for smoother alpha decay
```

[ ]:
```
model3['NLP']
```

[ ]:
```
array([ 1.3146516e-03,  8.7559475e-03, -2.5728724e-03,  1.1237229e-04,
        1.8875578e-03, -5.0069867e-03,  6.7687416e-03, -9.9992510e-03,
       -4.1280923e-04, -2.2862162e-03, -9.7010909e-03,  6.9062519e-03,
        9.5691029e-03, -3.0840512e-03,  4.4608414e-03, -1.7764017e-03,
       -8.2435962e-03, -3.2646020e-03, -9.9094855e-03, -9.4692912e-03,
        8.7401075e-03,  2.2220907e-03, -8.7813307e-03, -4.3473379e-03,
       -3.9097221e-04,  4.8672589e-03,  1.5777374e-03, -2.6316158e-04,
       -8.8995704e-03,  9.8497635e-03,  4.2200121e-03,  3.4971384e-03,
        2.8517384e-03,  8.2129147e-03,  8.1832148e-03,  1.3322410e-03,
        1.2316037e-05,  5.6195138e-03,  2.0136524e-03, -4.8742563e-04,
       -7.2859200e-03, -9.2956088e-03, -6.6227694e-03,  1.2034532e-03,
        5.1734657e-03,  5.3558438e-03,  6.4542238e-03, -5.8507882e-03,
        2.3829192e-03,  1.3914639e-03], dtype=float32)
```

[ ]:
```
model4['NLP']
```

[ ]:
```
array([ 1.3146516e-03,  8.7559475e-03, -2.5728724e-03,  1.1237229e-04,
        1.8875578e-03, -5.0069867e-03,  6.7687416e-03, -9.9992510e-03,
       -4.1280923e-04, -2.2862162e-03, -9.7010909e-03,  6.9062519e-03,
        9.5691029e-03, -3.0840512e-03,  4.4608414e-03, -1.7764017e-03,
       -8.2435962e-03, -3.2646020e-03, -9.9094855e-03, -9.4692912e-03,
        8.7401075e-03,  2.2220907e-03, -8.7813307e-03, -4.3473379e-03,
       -3.9097221e-04,  4.8672589e-03,  1.5777374e-03, -2.6316158e-04,
       -8.8995704e-03,  9.8497635e-03,  4.2200121e-03,  3.4971384e-03,
        2.8517384e-03,  8.2129147e-03,  8.1832148e-03,  1.3322410e-03,
        1.2316037e-05,  5.6195138e-03,  2.0136524e-03, -4.8742563e-04,
       -7.2859200e-03, -9.2956088e-03, -6.6227694e-03,  1.2034532e-03,
        5.1734657e-03,  5.3558438e-03,  6.4542238e-03, -5.8507882e-03,
        2.3829192e-03,  1.3914639e-03], dtype=float32)
```

```
[ ]: model3['business']
```

```
[ ]: array([ 0.00311132,  0.00734582, -0.0038429 ,  0.00668994, -0.00175607,
       -0.00351088,  0.00892915, -0.00773621, -0.00191435,  0.00024624,
       -0.00823071, -0.00186822,  0.00824068, -0.0065991 , -0.00107715,
       -0.00354975,  0.00752701, -0.00794327, -0.00778868, -0.00164652,
        0.0068918 ,  0.00291673,  0.0021475 , -0.0081624 , -0.00310096,
        0.00414062,  0.00169684,  0.00954767,  0.0094126 , -0.00751523,
        0.00489935,  0.00634651,  0.00778395,  0.00034678, -0.00514487,
        0.00600662,  0.00944785, -0.00905489,  0.00700527,  0.00811886,
        0.00404599,  0.00191028, -0.0073443 ,  0.00705282, -0.00915282,
       -0.00484462, -0.00525144, -0.00837096, -0.00492031,  0.00252984],
      dtype=float32)
```

```
[ ]: model4['GPS']
```

```
[ ]: array([-0.00010812,  0.00654889, -0.0096795 ,  0.00419711,  0.00881517,
        0.00870045, -0.00085944,  0.00877158, -0.0076904 , -0.00458279,
        0.00690111,  0.00925323,  0.00221632, -0.00508633, -0.00584864,
        0.00062931,  0.00525857,  0.00349085, -0.00545431, -0.00413292,
        0.00664355, -0.00995706, -0.00188874,  0.00823313,  0.00427296,
        0.0067082 , -0.00952195, -0.00061338, -0.00569869, -0.0013225 ,
       -0.00407934,  0.00688981,  0.00871297, -0.00707884, -0.00306569,
        0.00214769, -0.00354203,  0.00349761, -0.00123983, -0.00031144,
        0.00984737, -0.00711029,  0.00292753,  0.00291804, -0.00564377,
       -0.00509669, -0.00781621,  0.00239007,  0.0001292 ,  0.00866508],
      dtype=float32)
```

```
[ ]: model3.most_similar('text')
```

```
[ ]: [('There's', 0.2794710099697113),
      ('commands,', 0.26954254508018494),
      ('mission-critical', 0.25699689984321594),
      ('consumer', 0.19439628720283508),
      ('dictation', 0.18717510998249054),
      ('employee', 0.18183262646198273),
      ('another,', 0.16919739544391632),
      ('large', 0.15288642048835754),
      ('simplify', 0.12655995786190033),
 ('time.', 0.12331050634384155)]
```

[ ]:
```
model4.most_similar('NLP')
```

[ ]:
```
[('help', 0.25218427181243896),
 ('one', 0.213368222117424),
 ('business', 0.18884296715259552),
 ('dictation', 0.1857631653547287),
 ('rapidly-even', 0.16017085313796997),
 ('computer', 0.15341691672801971),
 ('But', 0.14902864396572113),
 ('summarize', 0.14282067120075226),
 ('commands,', 0.13742339611053467),
 ('digital', 0.13535983860492706)]
```

[33]:
```
word_embedding_avg_collectiveiii = []            #Averaged vector for the␣
 ↪entire sentence-skipgram
counter = 0
for doc in values:
    vec = get_mean_vector(model3, doc)
    word_embedding_avg_collectiveiii.append(vec)
    if len(vec) > 0:
      counter += 1
      print('Dimension of vector for each word:',vec.shape)

word_embedding_avgiii = np.mean(word_embedding_avg_collectiveiii, axis=0)
print()
print('Total words in corpus:',counter)
print('Dimension of vector for overall corpus:',word_embedding_avgiii.shape)

print('Vector for overall corpus:',word_embedding_avgiii)
```

```
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)

Total words in corpus: 58
Dimension of vector for overall corpus: (50,)
Vector for overall corpus: [ 1.8505657e-03 -5.7472562e-04  3.0958108e-05
-1.7864672e-03
```

```
  7.5191143e-04  5.5259484e-04 -8.7608385e-04  1.0918005e-03
 -9.4184658e-04 -1.1760849e-03  2.2986478e-03 -9.9459408e-05
 -1.0390236e-03 -9.1631658e-04 -5.3231308e-04 -6.3729647e-04
  1.4189096e-03 -2.1073164e-04  1.9913938e-04 -2.3064798e-05
  1.5755862e-04 -7.3771534e-04 -7.8160356e-04  1.1761512e-03
 -4.6985960e-04  1.5277104e-03  2.2414820e-03  3.6727515e-04
 -2.7354152e-04 -4.1866759e-04  1.0230073e-03 -1.3033379e-04
 -2.3361498e-03 -1.0916896e-03 -1.2578244e-03 -1.4172550e-04
 -6.2927307e-04  4.7993584e-04  7.7455543e-04 -4.8710566e-04
  5.1863655e-04 -5.3540576e-04 -2.7897570e-04  1.4048805e-03
  3.9726321e-04 -3.0828876e-05 -6.4594824e-05  9.7272446e-04
 -1.7769572e-04 -4.1517388e-04]
```

[63]:
```
word_embedding_avg_collectiveiv = []          #Averaged vector for the entire␣
 ↪sentence-skipgram
counter = 0
for doc in values:
    vec = get_mean_vector(model4, doc)
    word_embedding_avg_collectiveiv.append(vec)
    if len(vec) > 0:
      counter += 1
      print('Dimension of vector for each word:',vec.shape)

word_embedding_avgiv = np.mean(word_embedding_avg_collectiveiv, axis=0)
print()
print('Total words in corpus:',counter)
print('Dimension of vector for overall corpus:',word_embedding_avgiv.shape)

print('Vector for overall corpus:',word_embedding_avgiv)
```

```
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)
Dimension of vector for each word: (50,)

Total words in corpus: 58
Dimension of vector for overall corpus: (50,)
Vector for overall corpus: [ 1.8505657e-03 -5.7472562e-04  3.0958108e-05
-1.7864672e-03
  7.5191143e-04  5.5259484e-04 -8.7608385e-04  1.0918005e-03
 -9.4184658e-04 -1.1760849e-03  2.2986478e-03 -9.9459408e-05
 -1.0390236e-03 -9.1631658e-04 -5.3231308e-04 -6.3729647e-04
```

```
  1.4189096e-03 -2.1073164e-04  1.9913938e-04 -2.3064798e-05
  1.5755862e-04 -7.3771534e-04 -7.8160356e-04  1.1761512e-03
 -4.6985960e-04  1.5277104e-03  2.2414820e-03  3.6727515e-04
 -2.7354152e-04 -4.1866759e-04  1.0230073e-03 -1.3033379e-04
 -2.3361498e-03 -1.0916896e-03 -1.2578244e-03 -1.4172550e-04
 -6.2927307e-04  4.7993584e-04  7.7455543e-04 -4.8710566e-04
  5.1863655e-04 -5.3540576e-04 -2.7897570e-04  1.4048805e-03
  3.9726321e-04 -3.0828876e-05 -6.4594824e-05  9.7272446e-04
 -1.7769572e-04 -4.1517388e-04]
```

[53]:
```
#Doc2Vec                        #Averaged vector for the entire sentence DOC2VEC
temp = values
temp2ii = []
for i in temp:
    for j in i:
        temp2ii.append(j)
import gensim
docmodelii = gensim.models.doc2vec.Doc2Vec(vector_size=50, min_count=2,␣
 ↪epochs=50)

#building vocab
#docmodel.build_vocab([s.encode('utf-8').split() for s in temp2ii])


sentences = [gensim.models.doc2vec.TaggedDocument(sentence, 'tag') for sentence␣
 ↪in temp2ii]
docmodelii.build_vocab(sentences)
```

```
WARNING:gensim.models.base_any2vec:consider setting layer size to a multiple of
4 for greater performance
WARNING:gensim.models.doc2vec:Each 'words' should be a list of words (usually
unicode strings). First 'words' here is instead plain <class 'str'>.
```

[54]:
```
docmodelii.train(sentences, total_examples=docmodelii.corpus_count,␣
 ↪epochs=docmodelii.epochs)
```

[55]:
```
vectorii = docmodelii.infer_vector(temp2ii)    #dim 50
print(vectorii)
```

```
[ 0.00088993 -0.00712357  0.00204865 -0.00830369  0.00308902  0.00716895
  0.00359111 -0.00634366 -0.00768017  0.00855012  0.00871506  0.00488005
 -0.00846803 -0.00307154 -0.00663422 -0.00210881  0.00351603 -0.00541055
  0.00253019 -0.00269983  0.00804262  0.00401383  0.00844861  0.00689903
  0.00850768  0.00527708  0.00499489  0.00880536  0.00981555  0.00917694
 -0.00877941  0.00019085 -0.00945838 -0.00265259  0.00590502  0.00819771
  0.0083445   0.00250356 -0.00436055 -0.00926238 -0.00361688 -0.00864589
 -0.00965511 -0.00711285  0.00168902  0.00193385 -0.00681643 -0.00939186
 -0.00882162  0.00536393]
```

```
[66]: result2 = 1 - spatial.distance.cosine(word_embedding_avgiv, vectorii)
      result2

[66]: 0.18631403148174286
```

10 Conclusion:

1) Word embedding is one of the most important techniques in natural language processing(NLP), where words are mapped to vectors of real numbers.

2) Word embedding is capable of capturing the meaning of a word in a document, semantic and syntactic similarity, relation with other words.

3) For both task 2&3 we can observe that CBOW & Skipgram feature vector formed for the words are same i.e. there is no difference.

4) When we check the cosine similarity between the avg w2v and d2v vector for Task 2 we get -0.22 meaning the vectors are dissimilar and have negative direction/correlation.
   Whereas in task 3 doing the same on the new vectors, we got a cosine similarity of +0.186 meaning the vectors are similar and have positive correlation.
   This suggests that having large enough corpus (which we had for task3) will help to make better models of CBOW/Skipgram/D2V since they all showed some positive correlation amongst themselves meaning they were able to train better on this corpus than the one in task2.