```
Assignment No.5
Name : Shivani Gaikwad
Roll No. 43315
Batch : P-11
```

1. Import following libraries gensim and numpy set i.e. text file created . It should be preprocessed.

In [3]:
```
pip install -U gensim
```

```
Collecting gensim
Note: you may need to restart the kernel to use updated packages.  Using cached
gensim-4.2.0-cp38-cp38-win_amd64.whl (24.0 MB)
Requirement already satisfied: numpy>=1.17.0 in f:\anacondasetup\lib\site-packa
ges (from gensim) (1.20.1)
Requirement already satisfied: Cython==0.29.28 in f:\anacondasetup\lib\site-pac
kages (from gensim) (0.29.28)
Requirement already satisfied: smart-open>=1.8.1 in f:\anacondasetup\lib\site-p
ackages (from gensim) (6.2.0)
Requirement already satisfied: scipy>=0.18.1 in f:\anacondasetup\lib\site-packa
ges (from gensim) (1.6.2)
Installing collected packages: gensim
Successfully installed gensim-4.2.0
```

In [4]:
```
import numpy as np
import keras.backend as K
from keras.models import Sequential
from keras.layers import Dense, Embedding, Lambda
from keras.utils import np_utils
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
import gensim
```

2. Tokenize the every word from the paragraph . You can call in built tokenizer present in Gensim

In [7]:
```
data = open('C:/Users/hp/Downloads/LP-IV Lab (Deep Learning)/covid.txt', 'r')
corona_data = [text for text in data if text.count(' ') >= 2]
vectorize = Tokenizer()
```

3. Fit the data to tokenizer

In [8]:
```
vectorize.fit_on_texts(corona_data)
corona_data = vectorize.texts_to_sequences(corona_data)
```

4. Find total no of words and total no of sentences

In [9]:
```
total_vocab = sum(len(s) for s in corona_data)
word_count = len(vectorize.word_index) + 1
window_size = 2
```

5. Generate the pairs of Context words and target words

In [10]:
```
def cbow_model(data, window_size, total_vocab):
    total_length = window_size * 2
    for text in data:
        text_len = len(text)
        for idx, word in enumerate(text):
            context_word = []
            target = []
            begin = idx - window_size
            end = idx + window_size + 1
            context_word.append([text[i] for i in range(begin, end) if 0 <= i < t
            target.append(word)
            contextual = sequence.pad_sequences(context_word, total_length=total_
            final_target = np_utils.to_categorical(target, total_vocab)
            yield(contextual, final_target)
```

6. Create Neural Network model with following parameters:

Model type : sequential Layers : Dense , Lambda , embedding. Compile Options : (loss='categorical_crossentropy', optimizer='adam')

In [11]:
```
model = Sequential()
model.add(Embedding(input_dim=total_vocab, output_dim=100, input_length=window_si
model.add(Lambda(lambda x : K.mean(x, axis=1), output_shape=(100,)))
model.add(Dense(total_vocab, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer = 'adam')
for i in range(10):
    cost = 0
    for x, y in cbow_model(data, window_size, total_vocab):
        cost += model.train_on_batch(contextual, final_target)
    print(i, cost)
```

```
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
```

7. Create vector file of some word for testing

In [13]:
```
dimensions = 100
vect_file = open('C:/Users/hp/Downloads/LP-IV Lab (Deep Learning)/vectors.txt', '
vect_file.write('{} {}\n'.format(total_vocab, dimensions))
```

Out[13]:
```
8
```

8. Assign weights to your trained model

In [14]:
```
weigths = model.get_weights()[0]
for text, i in vectorize.word_index.items():
    final_vec = ' '.join(map(str, list(weigths[i, :])))
    vect_file.write('{} {}\n'.format(text, final_vec))
vect_file.close()
```

9. Use the vectors created in Gemsim

In [16]:
```
cbow_output = gensim.models.KeyedVectors.load_word2vec_format('C:/Users/hp/Downlo
```

10. choose the word to get similar type of words

In [19]:
```
cbow_output.most_similar(positive=['virus'])
```

Out[19]:
```
[('can', 0.28823187947273254),
 ('context', 0.2404625564813614),
 ('one', 0.2138516753911972),
 ('a', 0.21097348630428314),
 ('specific', 0.1929675191640854),
 ('viruses', 0.1822327971458435),
 ('of', 0.1768723428249359),
 ('contrast', 0.16680879890918732),
 ('serial', 0.16573864221572876),
 ('important', 0.1632171869277954)]
```

In [ ]:
```

```