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

```
In [ ]: 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 [ ]: data=open('/content/covid.txt','r')
        corona_data = [text for text in data if text.count(' ') >= 2]
        vectorize = Tokenizer()
```

3. Fit the data to tokenizer

```
In [ ]: 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 [ ]: 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 [ ]:  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 [ ]:
```
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 [ ]:
```
dimensions=100
vect_file = open('/content/vectors.txt' ,'w')
vect_file.write('{} {}\n'.format(total_vocab,dimensions))
```

Out[30]:
```
8
```

8. Assign weights to your trained model

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

9. Use the vectors created in Gemsim

In [ ]:
```
cbow_output = gensim.models.KeyedVectors.load_word2vec_format('/content/vectors.t
```

10. choose the word to get similar type of words

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

Out[35]:
```
[('higher', 0.28014785051345825),
 ('difference', 0.2435266375541687),
 ('5', 0.20107674598693848),
 ('context', 0.1770254671573639),
 ('while', 0.17489957809448242),
 ('pre', 0.1683725118637085),
 ('who', 0.16733811795711517),
 ('is', 0.14425021409988403),
 ('however', 0.14360009133815765),
 ('influenza', 0.1398591697216034)]
```

In [ ]: