Assignment 5

1. Name: Vaibhav Bichave
2. Batch: P-10
3. Roll No.: 43209

Problem Statement :

Implement the Continuous Bag of Words (CBOW) Model

Importing libraries

In [1]:

```
from keras.preprocessing import text
from keras.utils import np_utils
from keras.preprocessing import sequence
from keras_preprocessing.sequence import pad_sequences
import numpy as np
import pandas as pd
```

Taking random sentences as data

In [2]:

```
data = """Deep learning (also known as deep structured learning) is part of a broad
Deep-learning architectures such as deep neural networks, deep belief networks, dee
"""
dl_data = data.split()
```

a. Data preparation

Tokenization

In [3]:

```
tokenizer = text.Tokenizer()
tokenizer.fit_on_texts(dl_data)
word2id = tokenizer.word_index

word2id['PAD'] = 0
id2word = {v:k for k, v in word2id.items()}
wids = [[word2id[w] for w in text.text_to_word_sequence(doc)] for doc in dl_data]

vocab_size = len(word2id)
embed_size = 100
window_size = 2

print('Vocabulary Size:', vocab_size)
print('Vocabulary Sample:', list(word2id.items())[:10])
```

```
Vocabulary Size: 75
Vocabulary Sample: [('learning', 1), ('deep', 2), ('networks', 3), ('n
eural', 4), ('and', 5), ('as', 6), ('of', 7), ('machine', 8), ('superv
ised', 9), ('have', 10)]
```

b. Generate training data

Generating (context word, target/label word) pairs

In [4]:

```
def generate_context_word_pairs(corpus, window_size, vocab_size):
    context_length = window_size*2
    for words in corpus:
        sentence_length = len(words)
        for index, word in enumerate(words):
            context_words = []
            label_word   = []
            start = index - window_size
            end = index + window_size + 1

            context_words.append([words[i]
                                  for i in range(start, end)
                                  if 0 <= i < sentence_length
                                  and i != index])
            label_word.append(word)

            x = pad_sequences(context_words, maxlen=context_length)
            y = np_utils.to_categorical(label_word, vocab_size)
            yield (x, y)

i = 0
for x, y in generate_context_word_pairs(corpus=wids, window_size=window_size, vocab
    if 0 not in x[0]:
        print('Context (X):', [id2word[w] for w in x[0]], '-> Target (Y):', id2word

        if i == 10:
            break
        i += 1
```

c. Train model

Model building

In [5]:

```
import keras.backend as K
from keras.models import Sequential
from keras.layers import Dense, Embedding, Lambda

cbow = Sequential()
cbow.add(Embedding(input_dim=vocab_size, output_dim=embed_size, input_length=window
cbow.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(embed_size,)))
cbow.add(Dense(vocab_size, activation='softmax'))
cbow.compile(loss='categorical_crossentropy', optimizer='rmsprop')

print(cbow.summary())

# from IPython.display import SVG
# from keras.utils.vis_utils import model_to_dot

# SVG(model_to_dot(cbow, show_shapes=True, show_layer_names=False, rankdir='TB').cr
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 4, 100)            7500

 lambda (Lambda)             (None, 100)               0

 dense (Dense)               (None, 75)                7575

=================================================================
Total params: 15,075
Trainable params: 15,075
Non-trainable params: 0
_________________________________________________________________
None
```

In [6]:

```
for epoch in range(1, 6):
    loss = 0.
    i = 0
    for x, y in generate_context_word_pairs(corpus=wids, window_size=window_size, v
        i += 1
        loss += cbow.train_on_batch(x, y)
        if i % 100000 == 0:
            print('Processed {} (context, word) pairs'.format(i))

    print('Epoch:', epoch, '\tLoss:', loss)
    print()
```

```
Epoch: 1        Loss: 433.61818504333496

Epoch: 2        Loss: 428.8695614337921

Epoch: 3        Loss: 425.2637906074524

Epoch: 4        Loss: 421.93233609199524

Epoch: 5        Loss: 419.51635098457336
```

In [7]:

```
weights = cbow.get_weights()[0]
weights = weights[1:]
print(weights.shape)

pd.DataFrame(weights, index=list(id2word.values())[1:]).head()
```

```
(74, 100)
```

Out[7]:

	0	1	2	3	4	5	6	7
deep	-0.034130	0.024219	0.032866	0.053057	-0.015359	0.024081	-0.027761	0.015272
networks	-0.015954	-0.040530	0.016333	0.065731	-0.064257	-0.008575	-0.043316	0.004140
neural	-0.015125	-0.016590	-0.026489	-0.046720	0.038668	-0.012035	-0.045278	0.046965
and	0.029279	0.033890	0.049657	-0.037406	-0.049706	-0.005566	0.040193	0.014699
as	-0.009610	0.026094	-0.016352	0.039663	0.004246	-0.007173	-0.008121	-0.004822

5 rows × 100 columns

d. Output

In [8]:

```
from sklearn.metrics.pairwise import euclidean_distances

distance_matrix = euclidean_distances(weights)
print(distance_matrix.shape)

similar_words = {search_term: [id2word[idx] for idx in distance_matrix[word2id[sear
                  for search_term in ['deep']}

similar_words
```

```
(74, 74)
```

Out[8]:

```
{'deep': ['material', 'based', 'can', 'of', 'reinforcement']}
```

In [ ]: