In [1]:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize,sent_tokenize
```

In [2]:

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\varsh\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\varsh\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[2]:

```
True
```

In [3]:

```
text=""
```

In [4]:

```
stopWords=set(stopwords.words("english"))
print(stopWords)
```

```
{'s', 'are', "hadn't", "weren't", 'wouldn', 'there', "mightn't", 'no
r', 'ain', 'herself', 'is', 'doing', "mustn't", 'once', 'i', 'no', 'a
t', 'be', 'shan', 'but', "shouldn't", 'm', 'theirs', 'some', 'own', 'h
ave', 'being', 'under', 'when', 'which', 'against', "couldn't", 'itsel
f', 'haven', 'why', 'mightn', 'up', "aren't", 'did', 'then', 'off', 'm
yself', 'from', 'other', 've', 'd', 'them', 'yourself', 'has', 'me',
'does', 'ma', 'ours', 'during', 'over', 'mustn', 'too', 'hadn', 'befor
e', 'wasn', 'by', 'about', 'while', 'isn', 'out', "it's", 'so', 'its',
'most', "hasn't", 'here', 'all', 'only', 'hers', "shan't", 'am', 'mor
e', 'yourselves', 'those', 'your', 'should', 'whom', 'themselves', 'do
wn', 'she', 'the', 'as', "didn't", 'himself', 'him', 'until', 'now',
'his', 'o', 't', 'hasn', 'because', 'above', 'on', 'you', 'with', 'm
y', 'shouldn', 'into', 'their', 'same', 'we', 'and', "you'll", 'each',
'what', "needn't", 'few', 'didn', 'our', "won't", 'between', "you're",
'these', "wouldn't", 'her', 'after', 'a', 'very', 'can', 'he', 'it',
'couldn', "wasn't", 'if', 'any', 'to', 'aren', 'was', 'don', 'doesn',
"you'd", "don't", 'had', "doesn't", 'for', "that'll", 'do', 'y', 'thi
s', 'or', "haven't", 'having', 'than', 'been', 'in', 're', 'further',
'weren', 'below', 'that', 'needn', 'through', 'how', "isn't", 'they',
'where', 'were', 'an', 'who', 'of', "you've", "she's", 'both', 'oursel
ves', 'again', "should've", 'such', 'yours', 'won', 'just', 'll', 'no
t', 'will'}
```

In [6]:

```
words=word_tokenize(text)
print(words)
```

```
['Deep', 'multi-layer', 'neural', 'networks', 'have', 'many', 'level
s', 'of', 'non-linearities', 'allowing', 'them', 'to', 'compactly', 'r
epresent', 'highly', 'non-linear', 'and', 'highly-varying', 'function
s', '.', 'However', ',', 'until', 'recently', 'it', 'was', 'not', 'cle
ar', 'how', 'to', 'train', 'such', 'deep', 'networks', ',', 'since',
'gradient-based', 'optimization', 'starting', 'from', 'random', 'initi
alization', 'often', 'appears', 'to', 'get', 'stuck', 'in', 'poor', 's
olutions', '.', 'Hinton', 'et', 'al', '.', 'recently', 'proposed',
'a', 'greedy', 'layer-wise', 'unsupervised', 'learning', 'procedure',
'relying', 'on', 'the', 'training', 'algorithm', 'of', 'restricted',
'Boltzmann', 'machines', '(', 'RBM', ')', 'to', 'initialize', 'the',
'parameters', 'of', 'a', 'deep', 'belief', 'network', '(', 'DBN', ')',
',', 'a', 'generative', 'model', 'with', 'many', 'layers', 'of', 'hidd
en', 'causal', 'variables', '.', 'This', 'was', 'followed', 'by', 'th
e', 'proposal', 'of', 'another', 'greedy', 'layer-wise', 'procedure',
',', 'relying', 'on', 'the', 'usage', 'of', 'autoassociator', 'network
s', '.', 'In', 'the', 'context', 'of', 'the', 'above', 'optimization',
'problem', ',', 'we', 'study', 'these', 'algorithms', 'empirically',
'to', 'better', 'understand', 'their', 'success', '.', 'Our', 'experim
ents', 'confirm', 'the', 'hypothesis', 'that', 'the', 'greedy', 'layer
-wise', 'unsupervised', 'training', 'strategy', 'helps', 'the', 'optim
ization', 'by', 'initializing', 'weights', 'in', 'a', 'region', 'nea
r', 'a', 'good', 'local', 'minimum', ',', 'but', 'also', 'implicitly',
'acts', 'as', 'a', 'sort', 'of', 'regularization', 'that', 'brings',
'better', 'generalization', 'and', 'encourages', 'internal', 'distribu
ted', 'representations', 'that', 'are', 'high-level', 'abstractions',
'of', 'the', 'input', '.', 'We', 'also', 'present', 'a', 'series', 'o
f', 'experiments', 'aimed', 'at', 'evaluating', 'the', 'link', 'betwee
n', 'the', 'performance', 'of', 'deep', 'neural', 'networks', 'and',
'practical', 'aspects', 'of', 'their', 'topology', ',', 'for', 'exampl
e', ',', 'demonstrating', 'cases', 'where', 'the', 'addition', 'of',
'more', 'depth', 'helps', '.', 'Finally', ',', 'we', 'empirically', 'e
xplore', 'simple', 'variants', 'of', 'these', 'training', 'algorithm
s', ',', 'such', 'as', 'the', 'use', 'of', 'different', 'RBM', 'inpu
t', 'unit', 'distributions', ',', 'a', 'simple', 'way', 'of', 'combini
ng', 'gradient', 'estimators', 'to', 'improve', 'performance', ',', 'a
s', 'well', 'as', 'on-line', 'versions', 'of', 'those', 'algorithms',
'.', 'Keywords', ':', 'artificial', 'neural', 'networks', ',', 'deep',
'belief', 'networks', ',', 'restricted', 'Boltzmann', 'machines', ',',
'autoassociators', ',', 'unsupervised', 'learning']
```

In [7]:

```
freqTable=dict()
```

In [8]:

```python
freqTable=dict()
for word in words:
    word=word.lower()
    if word in stopWords:
        continue
    if word in freqTable:
        freqTable[word]+=1
    else:
        freqTable[word]=1
print(freqTable)
```

```
{'deep': 5, 'multi-layer': 1, 'neural': 3, 'networks': 6, 'many': 2,
'levels': 1, 'non-linearities': 1, 'allowing': 1, 'compactly': 1, 'rep
resent': 1, 'highly': 1, 'non-linear': 1, 'highly-varying': 1, 'functi
ons': 1, '.': 9, 'however': 1, ',': 16, 'recently': 2, 'clear': 1, 'tr
ain': 1, 'since': 1, 'gradient-based': 1, 'optimization': 3, 'startin
g': 1, 'random': 1, 'initialization': 1, 'often': 1, 'appears': 1, 'ge
t': 1, 'stuck': 1, 'poor': 1, 'solutions': 1, 'hinton': 1, 'et': 1, 'a
l': 1, 'proposed': 1, 'greedy': 3, 'layer-wise': 3, 'unsupervised': 3,
'learning': 2, 'procedure': 2, 'relying': 2, 'training': 3, 'algorith
m': 1, 'restricted': 2, 'boltzmann': 2, 'machines': 2, '(': 2, 'rbm':
2, ')': 2, 'initialize': 1, 'parameters': 1, 'belief': 2, 'network':
1, 'dbn': 1, 'generative': 1, 'model': 1, 'layers': 1, 'hidden': 1, 'c
ausal': 1, 'variables': 1, 'followed': 1, 'proposal': 1, 'another': 1,
'usage': 1, 'autoassociator': 1, 'context': 1, 'problem': 1, 'study':
1, 'algorithms': 3, 'empirically': 2, 'better': 2, 'understand': 1, 's
uccess': 1, 'experiments': 2, 'confirm': 1, 'hypothesis': 1, 'strateg
y': 1, 'helps': 2, 'initializing': 1, 'weights': 1, 'region': 1, 'nea
r': 1, 'good': 1, 'local': 1, 'minimum': 1, 'also': 2, 'implicitly':
1, 'acts': 1, 'sort': 1, 'regularization': 1, 'brings': 1, 'generaliza
tion': 1, 'encourages': 1, 'internal': 1, 'distributed': 1, 'represent
ations': 1, 'high-level': 1, 'abstractions': 1, 'input': 2, 'present':
1, 'series': 1, 'aimed': 1, 'evaluating': 1, 'link': 1, 'performance':
2, 'practical': 1, 'aspects': 1, 'topology': 1, 'example': 1, 'demonst
rating': 1, 'cases': 1, 'addition': 1, 'depth': 1, 'finally': 1, 'expl
ore': 1, 'simple': 2, 'variants': 1, 'use': 1, 'different': 1, 'unit':
1, 'distributions': 1, 'way': 1, 'combining': 1, 'gradient': 1, 'estim
ators': 1, 'improve': 1, 'well': 1, 'on-line': 1, 'versions': 1, 'keyw
ords': 1, ':': 1, 'artificial': 1, 'autoassociators': 1}
```

In [9]:

```
sentences=sent_tokenize(text)
print(sentences)
```

```
['Deep multi-layer neural networks have many levels of non-linearities
allowing them to compactly\nrepresent highly non-linear and highly-var
ying functions.', 'However, until recently it was not clear\nhow to tr
ain such deep networks, since gradient-based optimization starting fro
m random initialization often appears to get stuck in poor solution
s.', 'Hinton et al.', 'recently proposed a greedy\nlayer-wise unsuperv
ised learning procedure relying on the training algorithm of restricte
d Boltzmann machines (RBM) to initialize the parameters of a deep beli
ef network (DBN), a generative\nmodel with many layers of hidden causa
l variables.', 'This was followed by the proposal of another\ngreedy l
ayer-wise procedure, relying on the usage of autoassociator network
s.', 'In the context of\nthe above optimization problem, we study thes
e algorithms empirically to better understand their\nsuccess.', 'Our e
xperiments confirm the hypothesis that the greedy layer-wise unsupervi
sed training\nstrategy helps the optimization by initializing weights
in a region near a good local minimum, but\nalso implicitly acts as a
sort of regularization that brings better generalization and encourage
s internal distributed representations that are high-level abstraction
s of the input.', 'We also present a series\nof experiments aimed at e
valuating the link between the performance of deep neural networks and
\npractical aspects of their topology, for example, demonstrating case
s where the addition of more\ndepth helps.', 'Finally, we empirically
explore simple variants of these training algorithms, such as\nthe use
of different RBM input unit distributions, a simple way of combining g
radient estimators to\nimprove performance, as well as on-line version
s of those algorithms.', 'Keywords: artificial neural networks, deep b
elief networks, restricted Boltzmann machines, autoassociators, unsupe
rvised learning']
```

In [10]:

```
sentences[0]
```

Out[10]:

```
'Deep multi-layer neural networks have many levels of non-linearities
allowing them to compactly\nrepresent highly non-linear and highly-var
ying functions.'
```

In [11]:

```
def getsentenceValue():
    sentenceValue=dict()
    for sentence in sentences:
        for word,freq in freqTable.items():
            if word in sentence.lower():
                if sentence in sentenceValue:
                    sentenceValue[sentence]+=freq
                else:
                    sentenceValue[sentence]=freq
    return sentenceValue
    print(sentenceValue)
sentenceValue=getsentenceValue()
print(sentenceValue)
```

```
{'Deep multi-layer neural networks have many levels of non-linearities
allowing them to compactly\nrepresent highly non-linear and highly-var
ying functions.': 41, 'However, until recently it was not clear\nhow t
o train such deep networks, since gradient-based optimization starting
from random initialization often appears to get stuck in poor solution
s.': 59, 'Hinton et al.': 12, 'recently proposed a greedy\nlayer-wise
unsupervised learning procedure relying on the training algorithm of r
estricted Boltzmann machines (RBM) to initialize the parameters of a d
eep belief network (DBN), a generative\nmodel with many layers of hidd
en causal variables.': 81, 'This was followed by the proposal of anoth
er\ngreedy layer-wise procedure, relying on the usage of autoassociato
r networks.': 49, 'In the context of\nthe above optimization problem,
we study these algorithms empirically to better understand their\nsucc
ess.': 43, 'Our experiments confirm the hypothesis that the greedy lay
er-wise unsupervised training\nstrategy helps the optimization by init
ializing weights in a region near a good local minimum, but\nalso impl
icitly acts as a sort of regularization that brings better generalizat
ion and encourages internal distributed representations that are high-
level abstractions of the input.': 77, 'We also present a series\nof e
xperiments aimed at evaluating the link between the performance of dee
p neural networks and\npractical aspects of their topology, for exampl
e, demonstrating cases where the addition of more\ndepth helps.': 63,
'Finally, we empirically explore simple variants of these training alg
orithms, such as\nthe use of different RBM input unit distributions, a
simple way of combining gradient estimators to\nimprove performance, a
s well as on-line versions of those algorithms.': 59, 'Keywords: artif
icial neural networks, deep belief networks, restricted Boltzmann mach
ines, autoassociators, unsupervised learning': 51}
```

In [12]:

```
def getsumValues():
    sumValues=0
    for sentence in sentenceValue:
        sumValues+=sentenceValue[sentence]
    average=int(sumValues/len(sentenceValue))
    return average
average=getsumValues()
print(average)
```

```
53
```

In [13]:

```
summary=''
for sentence in sentences:
    if(sentence in sentenceValue) and (sentenceValue[sentence]>(1.2*aver
        summary+=" "+sentence
print(summary)
```

```
 recently proposed a greedy
layer-wise unsupervised learning procedure relying on the training alg
orithm of restricted Boltzmann machines (RBM) to initialize the parame
ters of a deep belief network (DBN), a generative
model with many layers of hidden causal variables. Our experiments con
firm the hypothesis that the greedy layer-wise unsupervised training
strategy helps the optimization by initializing weights in a region ne
ar a good local minimum, but
also implicitly acts as a sort of regularization that brings better ge
neralization and encourages internal distributed representations that
are high-level abstractions of the input.
```

In [ ]:

```
```