IMPORTING REQUIRED LIBRARIES

In [1]:
```
import sagemaker
import boto3
import io
import os
from sagemaker import Session
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
sagemaker_session = sagemaker.Session()
bucket = 'mohan-bucket-cognizer'
subfolder = 'Assignment1/Data'
role = sagemaker.get_execution_role()
s3 = boto3.client('s3')
contents = s3.list_objects(Bucket=bucket, Prefix=subfolder)['Contents']
files=[]
for f in contents:
    files.append(f['Key'])
print(files)
```

```
['Assignment1/Data/', 'Assignment1/Data/data1.txt', 'Assignment1/Data/data2.txt', 'Assignment1/Data/data3.txt', 'Assignment1/Data/data4.txt', 'Assignment1/Data/data5.txt']
```

In [3]:
```
data = ''
for file in files[1:]:
    lines = s3.get_object(Bucket = bucket,Key = str(file))
    lines['Body']
    with io.FileIO('s.txt', 'w') as sample:
        for i in lines['Body']:
            sample.write(i)
    f = open("s.txt", "r")
    data = data + f.read()
```

In [4]:
```
print(data)
```

```
A reactive machine follows the most basic of AI principles and, as its name implies, is capable of only using i
ts intelligence to perceive and react to the world in front of it. A reactive machine cannot store a memory and
as a result cannot rely on past experiences to inform decision making in real-time.Perceiving the world directl
y means that reactive machines are designed to complete only a limited number of specialized duties. Intentiona
lly narrowing a reactive machine's worldview is not any sort of cost-cutting measure, however, and instead mean
s that this type of AI will be more trustworthy and reliable — it will react the same way to the same stimuli e
very time. A famous example of a reactive machine is Deep Blue, which was designed by IBM in the 1990's as a ch
ess-playing supercomputer and defeated international grandmaster Gary Kasparov in a game. Deep Blue was only ca
pable of identifying the pieces on a chess board and knowing how each moves based on the rules of chess, acknow
ledging each piece's present position, and determining what the most logical move would be at that moment. The
computer was not pursuing future potential moves by its opponent or trying to put its own pieces in better posi
tion. Every turn was viewed as its own reality, separate from any other movement that was made beforehand.Anoth
er example of a game-playing reactive machine is Google's AlphaGo. AlphaGo is also incapable of evaluating futu
re moves but relies on its own neural network to evaluate developments of the present game, giving it an edge o
ver Deep Blue in a more complex game. AlphaGo also bested world-class competitors of the game, defeating champi
on Go player Lee Sedol in 2016.Though limited in scope and not easily altered, reactive machine artificial inte
lligence can attain a level of complexity, and offers reliability when created to fulfill repeatable tasks.
```

In [5]:
```
l=data.split('.')
print(l)
print(len(l))
```

```
['A reactive machine follows the most basic of AI principles and, as its name implies, is capable of only using
its intelligence to perceive and react to the world in front of it', ' A reactive machine cannot store a memory
and as a result cannot rely on past experiences to inform decision making in real-time', 'Perceiving the world
directly means that reactive machines are designed to complete only a limited number of specialized duties', '
Intentionally narrowing a reactive machine's worldview is not any sort of cost-cutting measure, however, and in
stead means that this type of AI will be more trustworthy and reliable — it will react the same way to the same
stimuli every time', ' A famous example of a reactive machine is Deep Blue, which was designed by IBM in the 19
90's as a chess-playing supercomputer and defeated international grandmaster Gary Kasparov in a game', ' Deep B
lue was only capable of identifying the pieces on a chess board and knowing how each moves based on the rules o
f chess, acknowledging each piece's present position, and determining what the most logical move would be at th
at moment', ' The computer was not pursuing future potential moves by its opponent or trying to put its own pie
ces in better position', ' Every turn was viewed as its own reality, separate from any other movement that was
made beforehand', 'Another example of a game-playing reactive machine is Google's AlphaGo', ' AlphaGo is also i
ncapable of evaluating future moves but relies on its own neural network to evaluate developments of the presen
t game, giving it an edge over Deep Blue in a more complex game', ' AlphaGo also bested world-class competitors
of the game, defeating champion Go player Lee Sedol in 2016', 'Though limited in scope and not easily altered,
reactive machine artificial intelligence can attain a level of complexity, and offers reliability when created
to fulfill repeatable tasks', '']
13
```

In [6]:
```
fstring = []
for i in range(0,len(l)):
    s = re.sub('[^a-zA-Z]',' ',str(l[i]))
    s = s.lower()
    s = s.split()
    ps = PorterStemmer()
    swords = stopwords.words('english')
    s = [ps.stem(word) for word in s if not word in set(swords)]
    s = ' '.join(s)
    fstring.append(s)
fstring
```

Out[6]:
```
['reactiv machin follow basic ai principl name impli capabl use intellig perceiv react world front',
 'reactiv machin cannot store memori result cannot reli past experi inform decis make real time',
 'perceiv world directli mean reactiv machin design complet limit number special duti',
 'intent narrow reactiv machin worldview sort cost cut measur howev instead mean type ai trustworthi reliabl re
act way stimuli everi time',
 'famous exampl reactiv machin deep blue design ibm chess play supercomput defeat intern grandmast gari kasparov
game',
 'deep blue capabl identifi piec chess board know move base rule chess acknowledg piec present posit determin l
ogic move would moment',
 'comput pursu futur potenti move oppon tri put piec better posit',
 'everi turn view realiti separ movement made beforehand',
 'anoth exampl game play reactiv machin googl alphago',
 'alphago also incap evalu futur move reli neural network evalu develop present game give edg deep blue complex
game',
 'alphago also best world class competitor game defeat champion go player lee sedol',
 'though limit scope easili alter reactiv machin artifici intellig attain level complex offer reliabl creat ful
fil repeat task',
 '']
```

In [7]:
```
result=''.join(fstring)
result
```

Out[7]:
```
'reactiv machin follow basic ai principl name impli capabl use intellig perceiv react world frontreactiv machin
cannot store memori result cannot reli past experi inform decis make real timeperceiv world directli mean react
iv machin design complet limit number special dutiintent narrow reactiv machin worldview sort cost cut measur h
owev instead mean type ai trustworthi reliabl react way stimuli everi timefamou exampl reactiv machin deep blue
design ibm chess play supercomput defeat intern grandmast gari kasparov gamedeep blue capabl identifi piec ches
s board know move base rule chess acknowledg piec present posit determin logic move would momentcomput pursu fu
tur potenti move oppon tri put piec better positeveri turn view realiti separ movement made beforehandanoth exa
mpl game play reactiv machin googl alphagoalphago also incap evalu futur move reli neural network evalu develop
present game give edg deep blue complex gamealphago also best world class competitor game defeat champion go pl
ayer lee sedolthough limit scope easili alter reactiv machin artifici intellig attain level complex offer relia
bl creat fulfil repeat task'
```

In [8]:
```
f = open("result.txt", "a")
f.write(result)
boto3.resource('s3').Bucket(bucket).Object(os.path.join('Assignment1/Results','result.txt')).upload_file('resul
```

In [9]:
```
contents = s3.list_objects(Bucket=bucket, Prefix='Assignment1/Results')['Contents']
for f in contents:
    print(f['Key'])
```

```
Assignment1/Results/result.txt
```

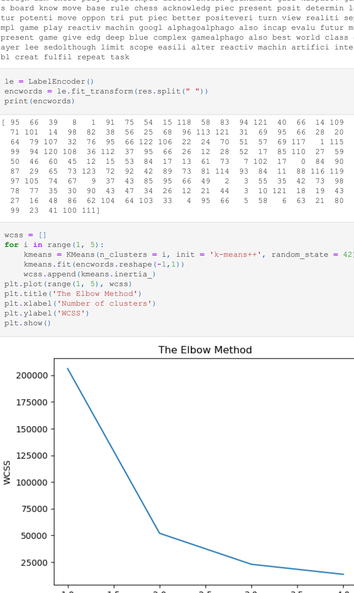
In [10]:
```
lines = s3.get_object(Bucket = bucket,Key = 'Assignment1/Results/result.txt')
lines['Body']
res=''
with io.FileIO('s.txt', 'w') as sample:
    for i in lines['Body']:
        sample.write(i)
f = open("s.txt", "r")
res = res + f.read()
print(res)
```

```
reactiv machin follow basic ai principl name impli capabl use intellig perceiv react world frontreactiv machin
cannot store memori result cannot reli past experi inform decis make real timeperceiv world directli mean react
iv machin design complet limit number special dutiintent narrow reactiv machin worldview sort cost cut measur h
owev instead mean type ai trustworthi reliabl react way stimuli everi timefamou exampl reactiv machin deep blue
design ibm chess play supercomput defeat intern grandmast gari kasparov gamedeep blue capabl identifi piec ches
s board know move base rule chess acknowledg piec present posit determin logic move would momentcomput pursu fu
tur potenti move oppon tri put piec better positeveri turn view realiti separ movement made beforehandanoth exa
mpl game play reactiv machin googl alphagoalphago also incap evalu futur move reli neural network evalu develop
present game give edg deep blue complex gamealphago also best world class competitor game defeat champion go pl
ayer lee sedolthough limit scope easili alter reactiv machin artifici intellig attain level complex offer relia
bl creat fulfil repeat task
```

In [11]:
```
le = LabelEncoder()
encwords = le.fit_transform(res.split(" "))
print(encwords)
```

```
[ 95  66  39   8   1  91  75  54  15 118  58  83  94 121  40  66  14 109
  71 101  14  98  82  38  56  25  68  96 113 121  31  69  95  66  28  20
  64  19 107  32  76  95  66 122 106  22  24  70  51  57  69 117   1 115
  99  94 120 108  36 112  37  95  66  26  12  28  52  17  85 110  27  59
  50  46  60  45  12  15  53  84  17  13  61  73   7 102  17   0  84  90
  87  29  65  73 123  72  92  42  89  73  81 114  93  84  11  88 116 119
  97 105  74  67   9  37  43  85  95  66  49   2   3  55  35  42  74  98
  78  77  35  30  90  43  47  34  26  12  21  44   3  10 121  18  19  43
  27  16  48  86  62 104  64 103  33   4  95  66   5  58   6  63  21  80
  99  23  41 100 111]
```

In [12]:
```
wcss = []
for i in range(1, 5):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(encwords.reshape(-1,1))
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 5), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [13]:
```
km=KMeans(n_clusters=3)
km
```

Out[13]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [14]:
```
y=km.fit_predict(encwords.reshape(-1,1))
y
```

Out[14]:
```
array([2, 1, 0, 0, 0, 2, 1, 1, 0, 2, 1, 2, 2, 2, 0, 1, 0, 2, 1, 2, 0, 2,
       2, 1, 0, 1, 2, 2, 2, 0, 1, 2, 1, 0, 0, 1, 1, 2, 0, 1, 2, 1, 2, 2,
       2, 0, 0, 1, 1, 1, 1, 2, 0, 2, 2, 2, 2, 2, 0, 2, 0, 2, 1, 0, 0, 0,
       1, 0, 2, 2, 0, 1, 1, 1, 1, 1, 0, 0, 1, 2, 0, 0, 1, 1, 0, 2, 0, 0,
       2, 2, 2, 0, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2,
       1, 1, 0, 0, 1, 2, 2, 1, 1, 0, 0, 1, 0, 1, 1, 2, 1, 1, 2, 0, 0, 2,
       1, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 1, 0, 0, 1, 2, 1, 2, 1, 2, 0, 0,
       2, 1, 0, 1, 0, 1, 2, 0, 1, 2, 2], dtype=int32)
```

In [15]:
```
km.cluster_centers_
```

Out[15]:
```
array([[ 19.72727273],
       [ 61.33928571],
       [101.        ]])
```

In [ ]:
```

```