Text and Social Media Analytics

Set A

Q1. Consider any text paragraph preprocess the text to remove any special characters and digits.Generate the summary using extractive summarization process.

In [ ]:
```
!pip install nltk
 import nltk
```

In [ ]:
```
nltk.download('all')
```

In [ ]:
```
nltk.download('punkt')
 nltk.download('stopwords')
```

In [12]:
```
text = """ Artificial intelligence is the future. Artificial intelligence is science fiction. Artificial intell
```

In [30]:
```
import nltk
 from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
```

In [31]:
```
import re
 import heapq
 def clean_data(data):
  text = re.sub(r"\([0-9]*\)"," ",data)
  text = text.lower()
  text = re.sub(r'\s+', " ",text)
  text = re.sub(r",", " ",text)
  return text
 cleaned_data = clean_data(text)

 sent_tokens = sent_tokenize(cleaned_data)
 word_tokens = word_tokenize(cleaned_data)
 word_frequency = {}
 stopwords = set(stopwords.words("english"))

 for word in word_tokens:
  if word not in stopwords:
  if word not in word_frequency.keys():
  word_frequency[word]=1
  else:
  word_frequency[word] +=1
 maximum_frequency = max(word_frequency.values())
 for word in word_frequency.keys():
  word_frequency[word] = (word_frequency[word]/maximum_frequency)


 sentences_score = {}
 for sentence in sent_tokens:
 for word in word_tokenize(sentence):
  if word in word_frequency.keys():
  if len(sentence.split(" ")) <10:
  if sentence not in sentences_score.keys():
  sentences_score[sentence] = word_frequency[word]
  else:
  sentences_score[sentence] += word_frequency[word]


 def get_key(val):
  for key, value in sentences_score.items():
  if val == value:
  return key
 key = get_key(max(sentences_score.values()))
 summary = heapq.nlargest(5,sentences_score,key=sentences_score.get)
 print(" ".join(summary))

artificial intelligence is already part of our everyday lives. and all three are part of the reason why alphago
trounced lee se-dol. all those statements are true it just depends on what flavor of ai you are referring to.
artificial intelligence is science fiction. artificial intelligence is the future.
```

Set A

Q2. Consider any text paragraph remove the stopwords. Tokenize the paragraph to extract words and sentences. Calculate the word frequency distribution and plot the frequency. Plot the wordcloud of the text.

In [51]:
```
import nltk
 from nltk.corpus import stopwords
 from nltk.tokenize import sent_tokenize, word_tokenize
 from nltk.probability import FreqDist
```

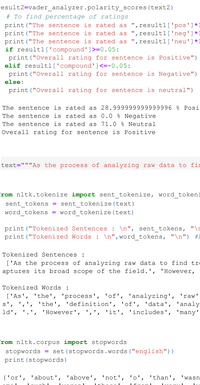
In [52]:
```
text="""An the process of analyzing raw data to find trends and answer questions, the definition of data analyt
```

In [53]:
```
sent_tokens = sent_tokenize(text)
 word_tokens = word_tokenize(text)
 stopwords = set(stopwords.words("english"))

 frequency_distribution=FreqDist(word_tokens)
 print(frequency_distribution)
 frequency_distribution
```

Out[53]:
```
<FreqDist with 30 samples and 38 outcomes>
FreqDist({'the': 3, 'of': 3, 'data': 2, ',': 2, '.': 2, 'many': 2, 'As': 1, 'process': 1, 'analyzing': 1, 'ra w': 1, ...})
```

In [54]:
```
import matplotlib.pyplot as plt
 ax = plt.axes()
 frequency_distribution.plot(20,cumulative=False) #first 20 words starting from maximum oon Xaxis
 ax.set_title('Frequency Plot of words')
 plt.show()
```



In [ ]:
```
!pip install wordcloud
 from wordcloud import WordCloud
```

In [74]:
```
wordcloud=WordCloud(background_color='black').generate(text)
```

In [75]:
```
plt.figure()
 plt.imshow(word_cloud, interpolation='bilinear')
 plt.axis("off")
 plt.show()
```

Set A

Q3. Consider the following review message .perform sentiment analysis on the message

In [ ]:
```
nltk.download('vader_lexicon')
```

In [77]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
 vader_analyzer=SentimentIntensityAnalyzer()
```

In [78]:
```
text1="I purchased headphones online. I am very happy with the product."
 print(vader_analyzer.polarity_scores(text1))

 {'neg': 0.0, 'neu': 0.667, 'pos': 0.333, 'compound': 0.6115}
```

In [79]:
```
text2="I saw the movie yesterday. The animation was really good but the script was ok."
 print(vader_analyzer.polarity_scores(text2))

 {'neg': 0.0, 'neu': 0.71, 'pos': 0.29, 'compound': 0.5989}
```

In [81]:
```
result2=vader_analyzer.polarity_scores(text2)
  # To find percentage of ratings
 print("The sentence is rated as ",result['pos']*100,"% Positive")
 print("The sentence is rated as ",result['neg']*100,"% Negative")
 print("The sentence is rated as ",result['neu']*100,"% Neutral")
  if result['compound']>=0.05:
  print("Overall rating for sentence is Positive")
 elif result['compound']<=-0.05:
  print("Overall rating for sentence is Negative")
 else:
  print("Overall rating for sentence is neutral")

 The sentence is rated as 28.999999999999996 % Positive
 The sentence is rated as 0.0 % Negative
 The sentence is rated as 71.0 % Neutral
 Overall rating for sentence is Positive
```

In [90]:
```
text="""An the process of analyzing raw data to find trends and answer questions, the definition of data analyt
```

In [92]:
```
from nltk.tokenize import sent_tokenize, word_tokenize
  sent_tokens = sent_tokenize(text)
  word_tokens = word_tokenize(text)

  print("Tokenized Sentences : \n", sent_tokens, "\n") #break paragraph into sentences (. full stop is considere
  print("Tokenized Words : \n",word_tokens, "\n") #break sentences into words ( <spaces are considered > )

 Tokenized Sentences :
  ['An the process of analyzing raw data to find trends and answer questions, the definition of data analytics c
 aptures its broad scope of the field.', 'However, it includes many techniques with many different goals.']

 Tokenized Words :
  ['An', 'the', 'process', 'of', 'analyzing', 'raw', 'data', 'to', 'find', 'trends', 'and', 'answer', 'question
 s', ',', 'the', 'definition', 'of', 'data', 'analytics', 'captures', 'its', 'broad', 'scope', 'of', 'the', 'fie
 ld', '.', 'However', ',', 'it', 'includes', 'many', 'techniques', 'with', 'many', 'different', 'goals', '.']
```

In [95]:
```
from nltk.corpus import stopwords
  stopwords = set(stopwords.words("english"))
  print(stopwords)

 {'or', 'about', 'above', 'not', 'o', 'than', 'wasn', 'yourselves', 'down', 'most', 'aren't", 'mustn', 'too', 's
 ame', 'such', 'weren', 'these', 'from', 'very', 'why', 'shan't", 'on', 'll', "hadn't", 'over', 'd', 'shan', 'bel
 ow', 'them', 'through', 've', 'herself', 'don', 'for', 'be', 'until', 'a', "haven't", "wouldn't", 'they', 'thei
 rs', 'i', 'when', 'hadn', 'ma', 'few', 'by', 'hers', 'couldn', 'it', 'under', 'no', "she's", 'because', 'thi
 s', 'should', 'just', 'after', 'he', 'what', 'will', 'needn', 'do', "it's", "won't", 'have', 'didn', 'did',
 't', 'won', 'itself', "doesn't", 'while', 'himself', 're', 'nor', 'is', 'in', "isn't", 'once', 'so', 'ours', 'y
 ourself', 'have', 'only', 'yours', 'ain', 'wasn't", 'shouldn', "you'll", 'y', 'out', 'm', 'to', "couldn't", "yo
 u'd", 'are', 'doesn', 'between', 'any', 'aren', 'been', 'haven', 'my', 'we', 'here', 'who', "don't", "that'll",
 "which", 'more', 'isn', 'it', "weren't", 'and', 'up', "shouldn't", 'again', 'of', 'against', 'his', 'other',
 'ourselves', 'me', "you're", 'those', 'were', 'during', "you've", 'but', 'an', 'where', 'was', 'then', 'does',
 'mightn't", 'a', 'this', "should've", 'your', 'before', 'both', 'had', 'the', 'further', 'having', 'can', 'is
 n', 'each', "wouldn't", 'with', "needn't", 'his', 'has', 'how', "didn't", 'being', 'as', "hasn't", "mightn", "ou
 r', "mustn't", 'am', 'themselves', 'some', 'doing', 'at', 'own', 'you', 'whom', 'there', 'all', 'that', 'its',
 'myself', 'shan', 'off', 'her', 'now'}
```

In [97]:
```
filtered_words_list=[]
  for words in word_tokens:
  if words not in stopwords:
  filtered_words_list.append(words)
 print("Tokenized Words : \n",word_tokens,"\n")
 print("Filtered Words : \n",filtered_words_list,"\n")

 Tokenized Words :
  ['An', 'the', 'process', 'of', 'analyzing', 'raw', 'data', 'to', 'find', 'trends', 'and', 'answer', 'question
 s', ',', 'the', 'definition', 'of', 'data', 'analytics', 'captures', 'its', 'broad', 'scope', 'of', 'the', 'fie
 ld', '.', 'However', ',', 'it', 'includes', 'many', 'techniques', 'with', 'many', 'different', 'goals', '.']

 Filtered Words :
  ['An', 'process', 'analyzing', 'raw', 'data', 'find', 'trends', 'answer', 'questions', ',', 'definition', 'dat
 a', 'analytics', 'captures', 'broad', 'scope', 'field', '.', 'However', ',', 'includes', 'many', 'techniques',
 'many', 'different', 'goals', '.']
```

In [98]:
```
#Stemming change writing, wrote, written can stemmed or reduced as write
  from nltk.stem import PorterStemmer
 porter_stemmer=PorterStemmer()
 stemmed_text_words=[]
  for words in filtered_words_list:
  stemmed_text_words.append(porter_stemmer.stem(words))
 print("Filtered Words : \n",word_tokens,"\n")
 print("Stemmed Words : \n",stemmed_text_words,"\n")

 Filtered Words :
  ['An', 'the', 'process', 'of', 'analyzing', 'raw', 'data', 'to', 'find', 'trends', 'and', 'answer', 'question
 s', ',', 'the', 'definition', 'of', 'data', 'analytics', 'captures', 'its', 'broad', 'scope', 'of', 'the', 'fie
 ld', '.', 'However', ',', 'it', 'includes', 'many', 'techniques', 'with', 'many', 'different', 'goals', '.']

 Stemmed Words :
  ['an', 'process', 'analyz', 'raw', 'data', 'find', 'trend', 'answer', 'question', ',', 'definit', 'data', 'ana
 lyt', 'captur', 'broad', 'scope', 'field', '.', 'howev', ',', 'includ', 'mani', 'techniqu', 'mani', 'differ',
 'goal', '.']
```

In [103]:
```
#Stemming
  from nltk.stem import PorterStemmer
  PorterStemmer().stem('eaten')
```

Out[103]:
```
'eaten'
```

In [104]:
```
PorterStemmer().stem('eating')
```

Out[104]:
```
'eat'
```

In [107]:
```
PorterStemmer().stem('eated')
```

Out[107]:
```
'eat'
```

Q. Find the lemmatized word for eaten.

In [ ]:
```
#Lemmatization
  import nltk
  nltk.download('wordnet')
```

In [106]:
```
# Lemmatization
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer=WordNetLemmatizer()
word_text="eaten"
print("Lemmatized Word :

",lemmatizer.lemmatize(word_text,"v"))Lemmatized Word : eat
```