
Practical 13 : Develop an NLP application**Program:**

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer import nltk sia =  
SentimentIntensityAnalyzer()  
  
# Tweets about AI  
tweets = [  
    "Artificial Intelligence is transforming the world in unimaginable ways!",  
  
    "AI can help solve complex problems but it must be handled responsibly.",  
  
    "I'm really excited to see how AI is being used in healthcare.",  
  
    "AI in education is going to make learning more personalized and accessible.",  
  
    "The future of AI is bright but we need to ensure it doesn't replace  
jobs.",  
    "AI technology is advancing faster than we can keep up with."  
]  
  
def analyze_sentiment(tweets):  
    for  
        tweet in tweets: print(f"Tweet:  
  
{tweet}") score =  
        sia.polarity_scores(tweet)  
  
        print(f"Sentiment Score: {score}")  
  
        print("\n")  
  
analyze_sentiment(tweets)
```

Output:

➡ Tweet: Artificial Intelligence is transforming the world in unimaginable ways!
Sentiment Score: {'neg': 0.0, 'neu': 0.702, 'pos': 0.298, 'compound': 0.5255}

Tweet: AI can help solve complex problems but it must be handled responsibly.
Sentiment Score: {'neg': 0.131, 'neu': 0.638, 'pos': 0.23, 'compound': 0.1027}

Tweet: I'm really excited to see how AI is being used in healthcare.
Sentiment Score: {'neg': 0.0, 'neu': 0.803, 'pos': 0.197, 'compound': 0.4005}

Tweet: AI in education is going to make learning more personalized and accessible.
Sentiment Score: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Tweet: The future of AI is bright but we need to ensure it doesn't replace jobs.
Sentiment Score: {'neg': 0.0, 'neu': 0.708, 'pos': 0.292, 'compound': 0.6542}

Tweet: AI technology is advancing faster than we can keep up with.
Sentiment Score: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Practical 14 : Implement Library for visual representations of text data

Program:

```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import
WordCloud import networkx as
nx from collections import
Counter import numpy as np
from sklearn.decomposition import PCA
#import pyLDAvis
#import
pyLDAvis.sklearn

def plot_wordcloud(text,max_words=100,colormap='viridis'):
wordcloud=WordCloud(max_words=max_words,colormap=colormap,background_color='white').gen
erat e(text)
plt.figure(figsize=(10,5))
plt.imshow(wordcloud,interpolation='bili
near') plt.axis('off') plt.show()

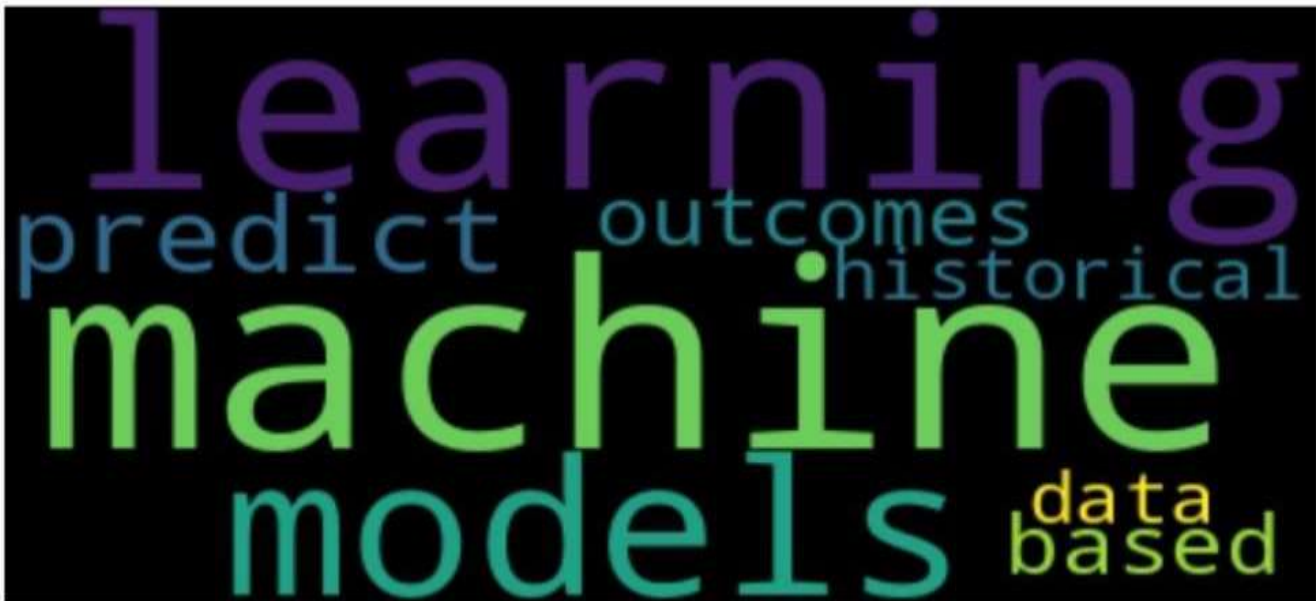
def
plot_barchart(text,top_n=20
): words=text.split()
word_counts=Counter(word
s)
most_common=word_counts.most_common(top_n)
labels,values=zip(*most_common)
plt.figure(figsize=(10,6))
sns.barplot(x=labels,y=values)
plt.title(f"Top {top_n} Words by Frequency.")
plt.show()

def plot_heatmap(data_matrix,x_labels,y_labels):
plt.figure(figsize=(10,6))
sns.heatmap(data_matrix,annot=True,cmap='coolwarm',xticklabels=x_labels,yticklabels=y_la
bels) plt.title("Heatmap of Text Features") plt.show()

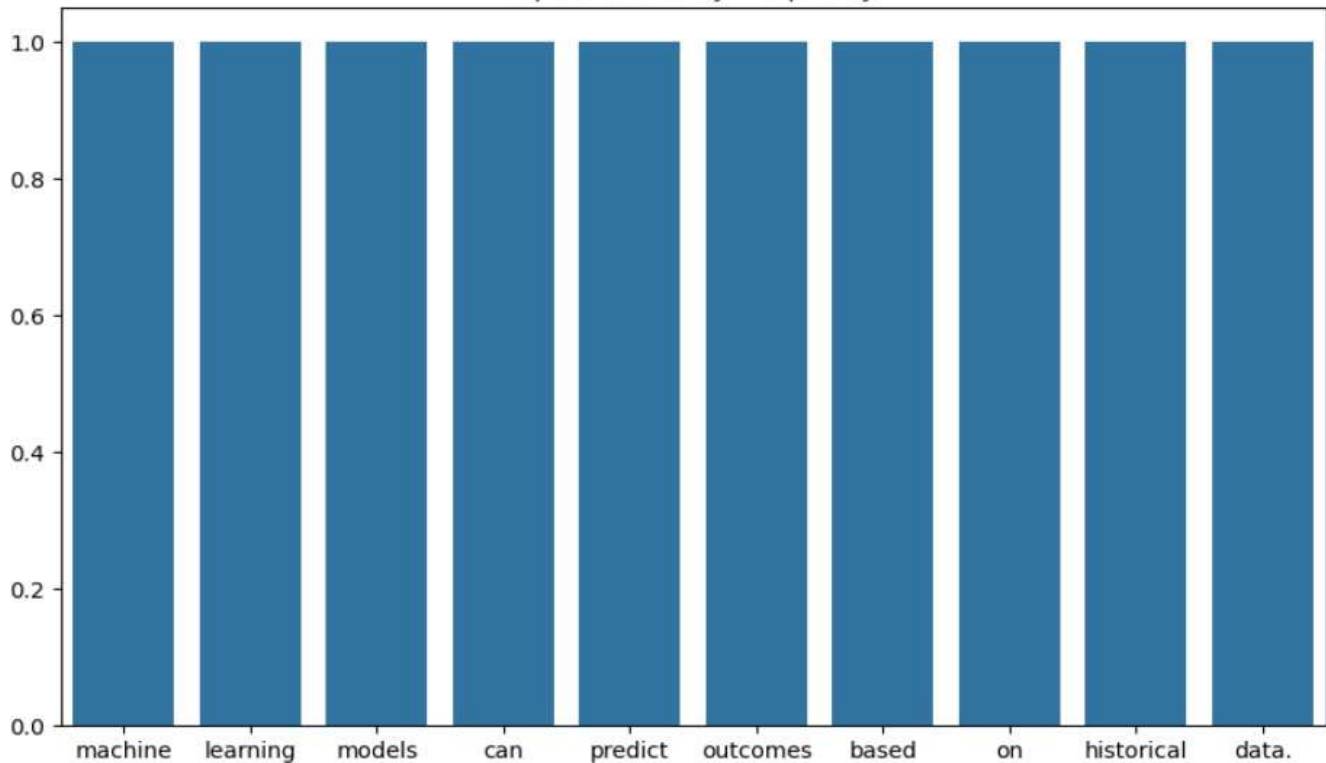
def
plot_graph(edges)
:
G=nx.Graph()
G.add_edges_from(edges)
plt.figure(figsize=(10,6))
pos=nx.spring_layout(G)
nx.draw(G,pos,with_labels=True,node_color='
lightblue',edge_color='gray',node_size=1500,f
ont_size=10
```

```
)  
plt.show()  
  
def plot_scatterplot(embeddings, labels=None):  
    pca=PCA(n_components=2)  
    reduced_embeddings=pca.fit_transform(embeddings)  
    plt.figure(figsize=(10,6))  
    plt.scatter(reduced_embeddings[:,0],reduced_embeddings[:,1],  
s=50) plt.xlabel("Principal Component 1") plt.ylabel("Principal  
Component 2") plt.title("PCA projection of text embeddings.")  
    plt.show()  
  
if __name__ == "__main__": text = "machine learning models can predict  
outcomes based on historical data." plot_wordcloud(text)  
plot_barchart(text) data=np.random.rand(10,10)  
plot_heatmap(data,x_labels=[f'word{i}' for i in range(10)],y_labels=[f'word{i}' for i in  
range(10)]) edges=[('A','B'),('B','C'),('C','D'),('D','E'),('E','A')] plot_graph(edges)  
embeddings=np.random.rand(100,100)  
plot_scatterplot(embeddings)
```

Output:



Top 20 Words by Frequency.



Heatmap of Text Features



