Practical 13: Develop an NLP application for Sentiment Analysis of Tweets

Source Code:

```
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.

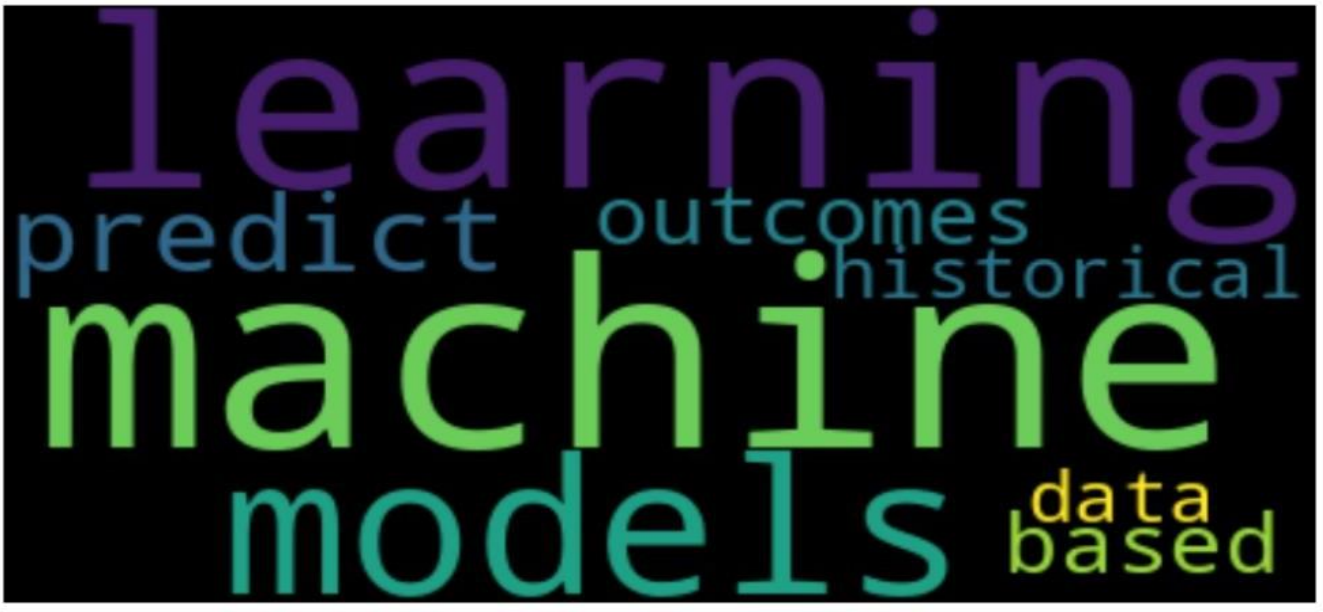
Source Code:

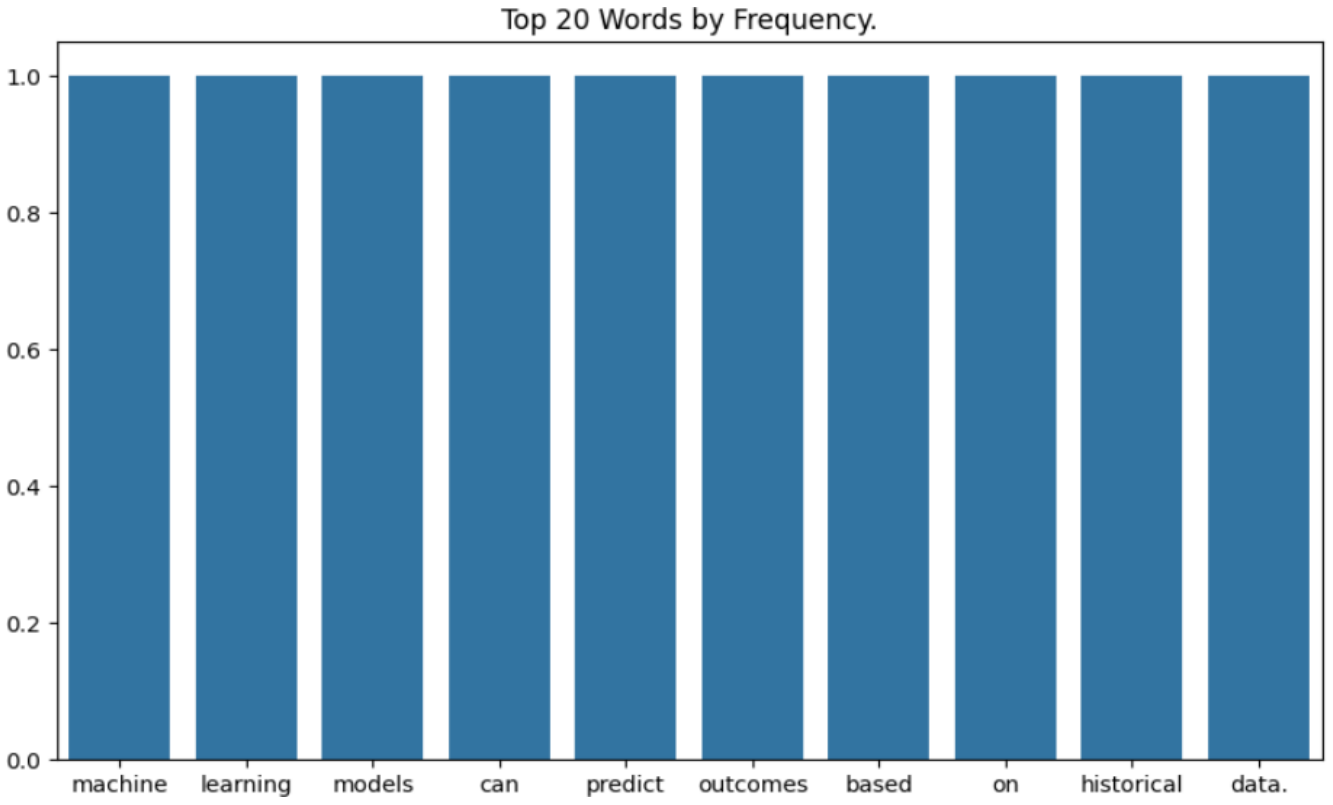
```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import networkx as nx
from collections import Counter
import numpy as np
from sklearn.decomposition import PCA
#import pyLDAvis
#import pyLDAvis.sklearn

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

def plot_barchart(text,top_n=20):
  words=text.split()
  word_counts=Counter(words)
  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_labels)
  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)
```

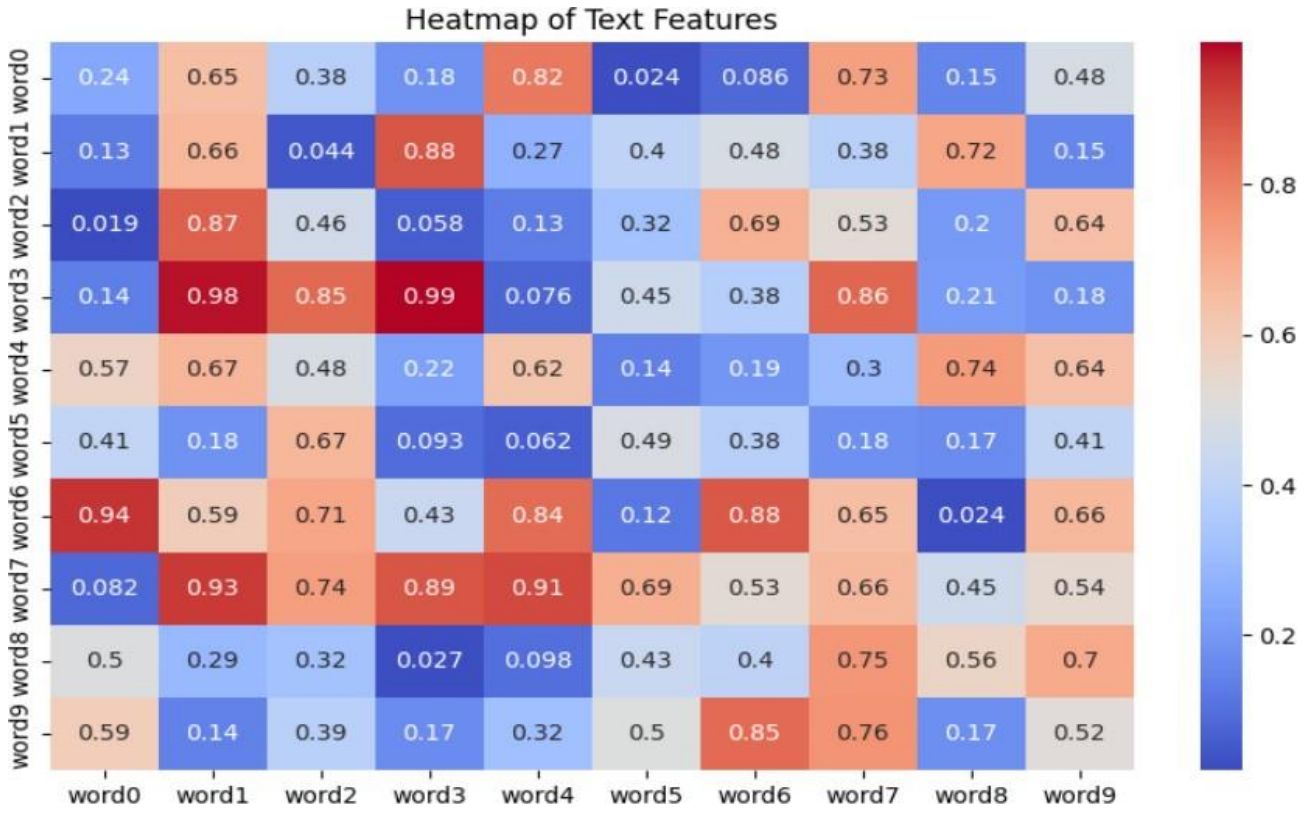
```
  nx.draw(G,pos,with_labels=True,node_color='lightblue',edge_color='gray',node_size=1500,font_size=10
)
  plt.show()

 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 :

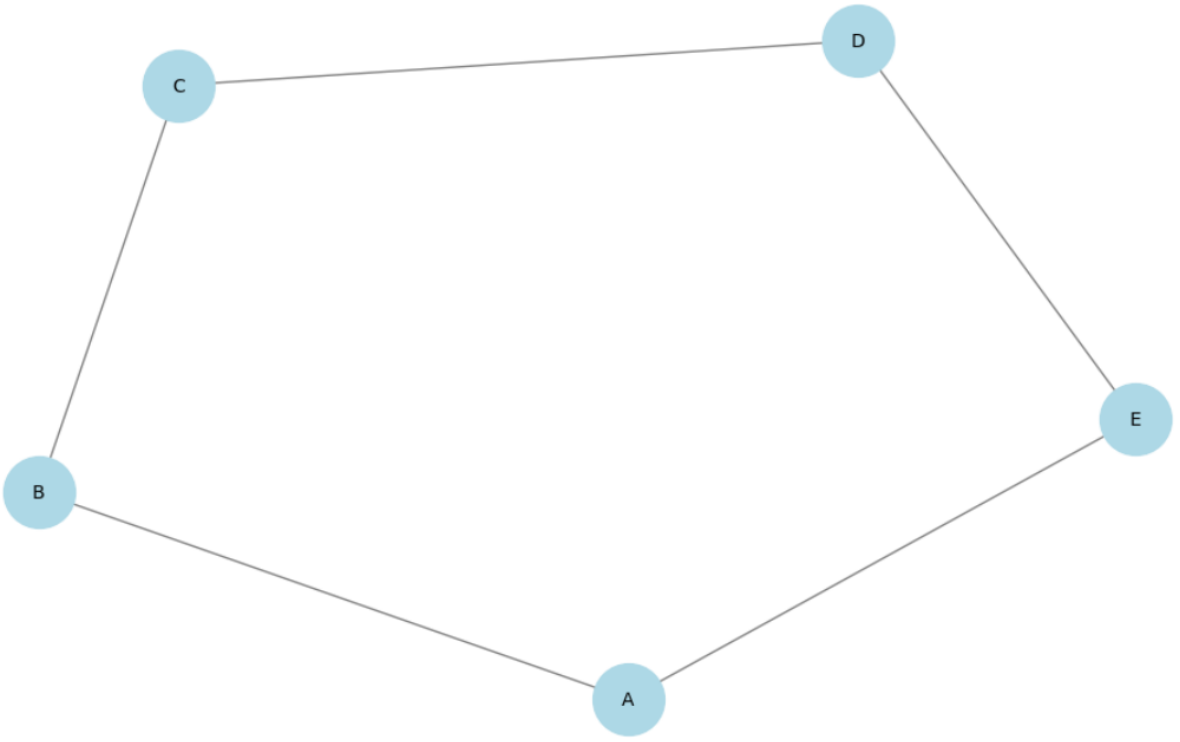
WordCloud :

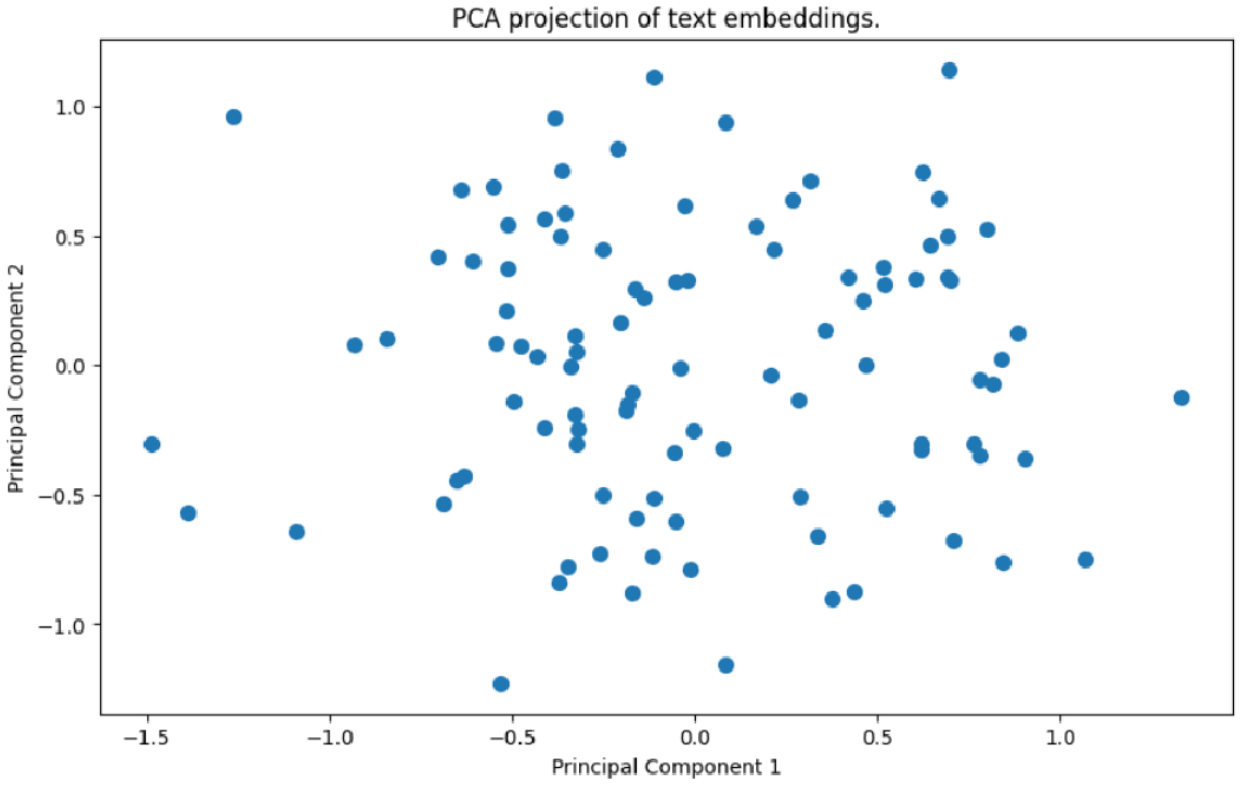



Top 20 Words by Frequency.
1.0
0.8
0.6
0.4
0.2
0.0
machine
learning
models
can
predict
outcomes
based
on
historical
data.


Heatmap of Text Features
word0 0.24 0.65 0.38 0.18 0.82 0.024 0.086 0.73 0.15 0.48
word1 0.13 0.66 0.044 0.88 0.27 0.4 0.48 0.38 0.72 0.15
word2 0.019 0.87 0.46 0.058 0.13 0.32 0.69 0.53 0.2 0.64
word3 0.14 0.98 0.85 0.99 0.076 0.45 0.38 0.86 0.21 0.18
word4 0.57 0.67 0.48 0.22 0.62 0.14 0.19 0.3 0.74 0.64
word5 0.41 0.18 0.67 0.093 0.062 0.49 0.38 0.18 0.17 0.41
word6 0.94 0.59 0.71 0.43 0.84 0.12 0.88 0.65 0.024 0.66
word7 0.082 0.93 0.74 0.89 0.91 0.69 0.53 0.66 0.45 0.54
word8 0.5 0.29 0.32 0.027 0.098 0.43 0.4 0.75 0.56 0.7
word9 0.59 0.14 0.39 0.17 0.32 0.5 0.85 0.76 0.17 0.52
word0 word1 word2 word3 word4 word5 word6 word7 word8 word9
0.8
0.6
0.4
0.2

Graph Plot


D
C
E
B
A


PCA projection of text embeddings.
Principal Component 2
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
Principal Component 1