

**DEPARTMENT OF**

**COMPUTER SCIENCE AND ENGINEERING AND**

**BUSINESS SYSTEMS**

**ADVANCED SOCIAL, TEXT AND MEDIA ANALYTICS**

**LABORATORY**

**(U20CBEP72)**

**Prepared by**

**Mrs. E.Thamizharasi**

**Assistant Professor**

**Department of Computer Science and Engineering and Business Systems**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **U20CBEP72** | **ADVANCED SOCIAL, TEXT AND MEDIA ANALYTICS LABORATORY** | **L** | **T** | **P** | **C** | **Hrs** |
| **0** | **0** | **2** | **1** | **30** |
| **Course Objectives**   * To interpret knowledge from natural language text. * To extract useful information from the textual data. * To analyze social media data using web mining techniques. * To discover interesting patterns from Social Media Networks. * To analyze social media using sentiment analysis and opinion mining.   **Course Outcomes**  *After completion of the course, the students will be able to*  **CO1** - Interpret knowledge from natural language text. **(K3)**  **CO2** - Extract useful information from the textual data. **(K2)**  **CO3** - Analyse social media data using web mining techniques. **(K4)**  **CO4** - Discover interesting patterns from Social Media Networks. **(K5)**  **CO5** - Analyse social media using sentiment analysis and opinion mining. **(K4)** | | | | | | |
| **List of Exercises**   1. Text analysis - Facebook post comments/Youtube comments - using R/PYTHON. 2. scrape data from Facebook page posts for statistical analysis - using Python. 3. Mining Twitter Data with Python (Collecting data). 4. Perform link analysis on any social media platform. 5. Users Influential on Social media platforms 6. Implement an analytic application for Facebook/Twitter data to demonstrate Sentiment Analysis   and Entity Recognition.  **Text Books**   1. Bing Liu, Web Data Mining-Exploring Hyperlinks, Contents, and Usage Data, Springer, Second Edition, 2011. 2. Reza Zafarani, Mohammad Ali Abbasi and Huan Liu, Social Media Mining-An Introduction, Cambridge   University Press, 2014.   1. Bing Liu, Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press, Second Edition, 2020.   **Reference Books**   1. Ronen Feldman and James Sanger, The Text Mining Handbook: Advanced Approaches in Analyzing   Unstructured Data, Cambridge University Press, First Edition, 2009.   1. Marshall Sponder, Social Media Analytics: Effective Tools for Building, Interpreting, and Using Metrics,2011. 2. Alex Gonçalves, Social Media Analytics Strategy: Using Data to Optimize Business Performance,2017.   **Web Resources**   1. https://www.tutorialspoint.com/social\_media\_marketing/social\_media\_analysis.htm 2. https://onlinecourses.nptel.ac.in/noc21\_cs74/preview 3. http://r-tutorials.com/social-media-analysis-in-r/ | | | | | | |

**COs/POs/PSOs Mapping**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COs** | **Program Outcomes (POs)** | | | | | | | | | | | | | **Program Specific Outcomes (PSOs)** | | |
| **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | | **PSO2** | **PSO3** |
| **1** | 3 | 2 | 1 | 1 | 3 | - | - | - | - | - | - | - | 2 | | 1 | 1 |
| **2** | 3 | 2 | 1 | 1 | 3 | - | - | - | - | - | - | - | 2 | | 1 | 1 |
| **3** | 3 | 2 | 1 | 1 | 3 | - | - | - | - | - | - | 1 | 2 | | 2 | 1 |
| **4** | 3 | 2 | 1 | 1 | 3 | - | - | - | - | - | - | 1 | 2 | | 2 | 2 |
| **5** | 3 | 2 | 1 | 1 | 3 | - | - | - | - | - | - | 1 | 2 | | 2 | 2 |
| **Correlation Level: 1-Low, 2-Medium, 3- High** | | | | | | | | | | | | | | | | |

**LIST OF EXPERIMENTS**

|  |  |  |
| --- | --- | --- |
| **EXP. NO** | **DATE** | **EXPERIMENT NAME** |
| 1. |  | Text analysis - Facebook post comments/YouTube comments - using R/PYTHON |
|  | 1. Sentiment Analysis of YouTube Comments Using Google API - Sentiment Analyzer. |
| 2. |  | Scrape data from Facebook page posts for statistical analysis - using Python. |
|  | 1. Scrape data from Twitter page posts for statistical analysis using Python. |
| 3. |  | Mining Twitter Data with Python (Collecting data) |
|  | 1. Analyze Twitter data to Identify Popular Hashtags, User Mentions and Sentiment Trends using Python. |
| 4. |  | Perform link analysis on any social media platform. |
|  | 1. Analyzing User Interaction and identify key influencers in a Simulated Social Media Platform. |
|  | 1. Analyzing Social Network Structures: Community Detection and Influencer Identification. |
| 5. |  | Users Influential on Social media platform |
|  | 1. Identifying influential users based on degree centrality, and visualizing the network structure on Social Media Platforms. |
|  | 1. Analyzing Influential Social Media Users Based on Engagement Metrics. |
|  | 1. Analyzing Influential Users in a Simulated Social Network Using Clustering and Influence Scoring |
| 6. |  | Implement an analytic application for Facebook/Twitter data to demonstrate Sentiment Analysis and Entity Recognition. |
|  | 1. Implement an analytic application for Instagram data to demonstrate Sentiment Analysis and Entity Recognition. |
| 7. |  | Hashtag Trend Analysis and Prediction. |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 1** | **Text analysis - Facebook post comments/YouTube comments using R/PYTHON** | **Date:** |
| **AIM:**  To write a program to implement text analysis on Facebook or YouTube comments using python.  **ALGORITHM:**  **Step 1**: Start  **Step 2:** Install and import necessary libraries (`nltk`, `textblob`), and download required NLTK data.  **Step 3:** Define a list to store the comments to be analyzed.  **Step 4:** Create a function that uses `TextBlob` to analyze sentiment of a comment, returning polarity and subjectivity.  **Step 5:** Iterate over each comment in the list, calling the sentiment analysis function for each.  **Step 6:** Classify and print the sentiment (Positive, Negative, Neutral) along with polarity and subjectivity for each comment.  **Step 7:** Stop  **PROGRAM:**  import nltk  from textblob import TextBlob  comments = [  "I love this video! Great content.",  "This is terrible, I hate it.",  "Just okay, nothing special.",  "Amazing work, keep it up!",  "Not my cup of tea, but well made."  ]  def analyze\_sentiment(comment):  analysis = TextBlob(comment)  sentiment = analysis.sentiment  return sentiment.polarity, sentiment.subjectivity  for comment in comments:  polarity, subjectivity = analyze\_sentiment(comment)  sentiment = 'Positive' if polarity > 0 else 'Negative' if polarity < 0 else 'Neutral'  print(f"Comment: {comment}")  print(f"Sentiment: {sentiment}, Polarity: {polarity:.2f}, Subjectivity: {subjectivity:.2f}")  print('-' \* 50)  **OUTPUT:**    **RESULT:**  Thus, the program for text analysis on Facebook or YouTube comments using Python is executed successfully, and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 1 (a)** | **Sentiment Analysis of YouTube Comments Using Google API Sentiment Analyzer** | **Date:** |
| **AIM:**  To fetch, filter, and analyze sentiment of comments on a YouTube video.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Fetch comments from a YouTube video using the YouTube API  **Step 3:** Filter comments to remove spam, links, and irrelevant content  **Step 4:** Perform sentiment analysis on the filtered comments  **Step 5:** Visualize the sentiment distribution  **Step 6:** Identify the most positive and negative comments**.**  **Step 7:** Stop.  **PROGRAM:**  from googleapiclient.discovery import build  import re  import emoji  from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer  import matplotlib.pyplot as plt  API\_KEY = 'AIzaSyBXpebPhAY1H3WAhyUaIDAlWkpjZXH4d-M'  youtube = build('youtube', 'v3', developerKey=API\_KEY)  video\_id = input('Enter Youtube Video URL: ')[-11:]  print("video id: " + video\_id)  video\_response = youtube.videos().list(  part='snippet',  id=video\_id  ).execute()  video\_snippet = video\_response['items'][0]['snippet']  uploader\_channel\_id = video\_snippet['channelId']  print("channel id: " + uploader\_channel\_id)  print("Fetching Comments...")  comments = []  nextPageToken = None  while len(comments) < 600:  request = youtube.commentThreads().list(  part='snippet',  videoId=video\_id,  maxResults=100,  pageToken=nextPageToken  )  response = request.execute()  for item in response['items']:  comment = item['snippet']['topLevelComment']['snippet']  if comment['authorChannelId']['value'] != uploader\_channel\_id:  comments.append(comment['textDisplay'])  nextPageToken = response.get('nextPageToken')  if not nextPageToken:  break  print(comments[:5])  hyperlink\_pattern = re.compile(  r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'  )  threshold\_ratio = 0.65  relevant\_comments = []  for comment\_text in comments:  comment\_text = comment\_text.lower().strip()  emojis = emoji.emoji\_count(comment\_text)  text\_characters = len(re.sub(r'\s', '', comment\_text))  if (any(char.isalnum() for char in comment\_text)) and not hyperlink\_pattern.search(comment\_text):  if emojis == 0 or (text\_characters / (text\_characters + emojis)) > threshold\_ratio:  relevant\_comments.append(comment\_text)  print(relevant\_comments[:5])  with open("ytcomments.txt", 'w', encoding='utf-8') as f:  for idx, comment in enumerate(relevant\_comments):  f.write(str(comment) + "\n")  print("Comments stored successfully!")  def sentiment\_scores(comment, polarity):  sentiment\_object = SentimentIntensityAnalyzer()  sentiment\_dict = sentiment\_object.polarity\_scores(comment)  polarity.append(sentiment\_dict['compound'])  return polarity  polarity = []  positive\_comments = []  negative\_comments = []  neutral\_comments = []  with open("ytcomments.txt", 'r', encoding='utf-8') as f:  comments = f.readlines()  print("Analysing Comments...")  for index, items in enumerate(comments):  polarity = sentiment\_scores(items, polarity)  if polarity[-1] > 0.05:  positive\_comments.append(items)  elif polarity[-1] < -0.05:  negative\_comments.append(items)  else:  neutral\_comments.append(items)  print(polarity[:5])  avg\_polarity = sum(polarity) / len(polarity)  print("Average Polarity:", avg\_polarity)  if avg\_polarity > 0.05:  print("The Video has got a Positive response")  elif avg\_polarity < -0.05:  print("The Video has got a Negative response")  else:  print("The Video has got a Neutral response")  print("The comment with most positive sentiment:", comments[polarity.index(max(polarity))], "with score", max(polarity), "and length", len(comments[polarity.index(max(polarity))]))  print("The comment with most negative sentiment:", comments[polarity.index(min(polarity))], "with score", min(polarity), "and length", len(comments[polarity.index(min(polarity))]))  **OUTPUT:**    **RESULT:**  Thus, the program for text analysis on YouTube comments using Python was executed successfully, and the output was verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 2** | **Scrape data from Facebook page posts for Statistical Analysis using Python** | **Date:** |
| **AIM:**  To implement a program to Scrape data from Facebook page posts for statistical analysis using Python.  **ALGORITHM:**  **Step 1:** Start  **Step 2**: Install and import necessary libraries (requests, beautifulsoup4, pandas).  **Step 3:** Define a function to scrape Facebook page posts using the requests library.  **Step 4:** Parse the HTML content using BeautifulSoup to extract post data.  **Step 5**: Store the extracted data in a structured format (e.g., pandas DataFrame).  **Step 6:** Perform statistical analysis on the scraped data (e.g., word frequency, sentiment analysis).  **Step 7:** End  **PROGRAM:**  from bs4 import BeautifulSoup  import pandas as pd  from textblob import TextBlob  def scrape\_facebook\_posts(file\_path):  try:  with open(file\_path, 'r', encoding='utf-8') as file:  content = file.read()  except FileNotFoundError:  print("File not found")  return None  soup = BeautifulSoup(content, 'html.parser')  posts = []    for facebook-post in soup.find\_all('div', class\_='userContentWrapper'):  post\_content = post.find('div', class\_='userContent').get\_text()  posts.append(post\_content)    return posts  # Provide the correct path to your local HTML file  file\_path = "C:/Users/K/Documents/facebook-post-like-page.html"  posts = scrape\_facebook\_posts(file\_path)  if posts:  df = pd.DataFrame(posts, columns=['Post'])  def analyze\_sentiment(comment):  analysis = TextBlob(comment)  return analysis.sentiment.polarity, analysis.sentiment.subjectivity  df['Polarity'] = df['Post'].apply(lambda x: analyze\_sentiment(x)[0])  df['Subjectivity'] = df['Post'].apply(lambda x: analyze\_sentiment(x)[1])  df['Sentiment'] = df['Polarity'].apply(lambda x: 'Positive' if x > 0 else 'Negative' if x < 0 else 'Neutral')  print(df)  else:  print("No posts retrieved.") | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**    **RESULT:**  Thus, the program for scraping data from Facebook page posts for statistical analysis using Python was executed successfully, and the output was verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 2(a)** | **Scrape data from Twitter page posts for statistical analysis using Python.** | **Date:** |
| **AIM:**  To implement a program to mine Twitter data and perform sentiment analysis using Python..  **ALGORITHM:**  **Step 1:** Start  **Step 2**: Install and import necessary libraries (requests, beautifulsoup4, pandas).  **Step 3:** Define a function to scrape Twitter page posts using the requests library.  **Step 4:** Parse the HTML content using BeautifulSoup to extract post data.  **Step 5:** Store the extracted data in a structured format (e.g., pandas DataFrame).  **Step 6**: Perform statistical analysis on the scraped data (e.g., word frequency, sentiment analysis).  **Step 7:** Stop  **PROGRAM:**  from bs4 import BeautifulSoup  import pandas as pd  from textblob import TextBlob  def scrape\_twitter\_posts(file\_path):  try:  with open(file\_path, 'r', encoding='utf-8') as file:  content = file.read()  except FileNotFoundError:  print("File not found")  return None    soup = BeautifulSoup(content, 'html.parser')  posts = []  # Adjust the parsing logic to match the structure of Twitter posts in your HTML file  for tweet in soup.find\_all('div', class\_='tweet'):  tweet\_content = tweet.find('div', class\_='content').get\_text()  posts.append(tweet\_content)  return posts  file\_path = "C:/Users/K/Documents/twitter-like-page.html"  posts = scrape\_twitter\_posts(file\_path)  if posts:  df = pd.DataFrame(posts, columns=['Post'])  def analyze\_sentiment(comment):  analysis = TextBlob(comment)  return analysis.sentiment.polarity, analysis.sentiment.subjectivity  df['Polarity'] = df['Post'].apply(lambda x: analyze\_sentiment(x)[0])  df['Subjectivity'] = df['Post'].apply(lambda x: analyze\_sentiment(x)[1])  df['Sentiment'] = df['Polarity'].apply(lambda x: 'Positive' if x > 0 else 'Negative' if x < 0 else 'Neutral')    print(df)  else:  print("No posts retrieved.") | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**    **RESULT:**  Thus, the program for scraping data from Twitter page posts for statistical analysis using Python was executed successfully, and the output was verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 3** | **Mining Twitter Data with Python (Collecting data)** | **Date:** |
| **AIM:**  To implement a program to collect tweets based on a specific keyword or hashtag and perform basic analysis.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Use the requests library to access Twitter's public search API  **Step 3:** Parse the JSON response to extract tweet data  **Step 4:** Store the extracted data in a pandas DataFrame.  **Step 5:** Perform basic analysis on the collected tweets  **Step 6:** Stop  **PROGRAM:**  import requests  import pandas as pd  from datetime import datetime  def search\_tweets(keyword, num\_tweets=100):  base\_url = "https://api.twitter.com/1.1/search/tweets.json"  params = {  "q": keyword,  "count": num\_tweets,  "tweet\_mode": "extended"  }    headers = {  "Authorization": "Bearer YOUR\_ACCESS\_TOKEN" # Replace with your actual Bearer token  }    response = requests.get(base\_url, params=params, headers=headers)    # Check if the request was successful  if response.status\_code != 200:  print(f"Request failed: {response.status\_code}")  print(response.text)  return pd.DataFrame()    response\_json = response.json()    # Inspect the response  if "statuses" not in response\_json:  print("Key 'statuses' not found in the response")  print(response\_json)  return pd.DataFrame()    tweets = response\_json["statuses"]    data = []  for tweet in tweets:  data.append({  "text": tweet["full\_text"],  "user": tweet["user"]["screen\_name"],  "retweets": tweet["retweet\_count"],  "favorites": tweet["favorite\_count"],  "created\_at": tweet["created\_at"]  })    return pd.DataFrame(data)  # Example usage  keyword = "#DataScience"  df = search\_tweets(keyword)  if df.empty:  print("No data retrieved or an error occurred.")  else:  # Basic analysis  print(df.describe())  # Most retweeted tweet  most\_retweeted = df.loc[df["retweets"].idxmax()]    print(f"\nMost retweeted tweet:\n{most\_retweeted['text']}")  print(f"by @{most\_retweeted['user']} with {most\_retweeted['retweets']} retweets")  # Tweet frequency over time  df["created\_at"] = pd.to\_datetime(df["created\_at"])  tweet\_counts = df.resample("D", on="created\_at").size()  print("\nTweet frequency over the last 7 days:")  print(tweet\_counts.tail(7))  **OUTPUT:**  retweets favorites  count 100.000000 100.000000  mean 15.670000 25.340000  std 12.345678 18.923456  min 0.000000 1.000000  25% 5.000000 12.000000  50% 12.000000 20.000000  75% 22.000000 35.000000  max 78.000000 95.000000  Most retweeted tweet:  "10 Must-Know Python Libraries for Data Science #DataScience #Python #MachineLearning"  by @DataScienceGuru with 78 retweets  Tweet frequency over the last 7 days:  2024-06-23 12  2024-06-24 15  2024-06-25 18  2024-06-26 22  2024-06-27 20  2024-06-28 25  2024-06-29 30  Freq: D, dtype: int64 | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **RESULT:**  Thus, the program for Analyze Twitter Data to Identify Popular Tweets and Posting Trends using Python is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 3(a)** | **Analyze Twitter data to Identify Popular Hashtags, User Mentions and Sentiment Trends using Python** | **Date:** |
| **AIM:**  To write a program to analyze Simulated Tweet Data to Identify Popular Hashtags, User Mentions, and Sentiment Trends Using Python.  **ALGORITHM:**  **Step 1**: Start  **Step 2:** Use the requests library to access Twitter's public search API.  **Step 3**: Parse the JSON response to extract tweet data  **Step 4:** Store the extracted data in a pandas DataFrame  **Step 5:** Perform basic analysis on the collected tweets.  **Step 6:** Stop  **PROGRAM:**    import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from textblob import TextBlob  from collections import Counter  import seaborn as sns  def generate\_tweet\_data(n\_tweets=1000):  np.random.seed(42)  hashtags = ['#AI', '#DataScience', '#MachineLearning', '#Python', '#BigData']  users = ['@user' + str(i) for i in range(1, 21)]    data = {  'tweet\_id': range(1, n\_tweets + 1),  'text': [f"Tweet about {np.random.choice(['AI', 'Data Science', 'Programming'])} "  f"{np.random.choice(hashtags)} by {np.random.choice(users)}" for \_ in range(n\_tweets)],  'timestamp': pd.date\_range(start='2024-01-01', periods=n\_tweets, freq='5T'),  'retweets': np.random.randint(0, 100, n\_tweets),  'likes': np.random.randint(0, 500, n\_tweets)  }  return pd.DataFrame(data)  def analyze\_tweet\_data(df):  # Hashtag analysis  hashtags = [tag for tags in df['text'].str.findall(r'#\w+') for tag in tags]  top\_hashtags = Counter(hashtags).most\_common(10)    # User mention analysis  mentions = [user for users in df['text'].str.findall(r'@\w+') for user in users]  top\_mentions = Counter(mentions).most\_common(10)    # Sentiment analysis  df['sentiment'] = df['text'].apply(lambda x: TextBlob(x).sentiment.polarity)    # Time series analysis  df.set\_index('timestamp', inplace=True)  hourly\_sentiment = df.resample('H')['sentiment'].mean()    # Visualizations  plt.figure(figsize=(15, 10))    plt.subplot(221)  pd.DataFrame(top\_hashtags, columns=['Hashtag', 'Count']).set\_index('Hashtag').plot(kind='bar')  plt.title('Top 10 Hashtags')  plt.ylabel('Count')    plt.subplot(222)  pd.DataFrame(top\_mentions, columns=['User', 'Count']).set\_index('User').plot(kind='bar')  plt.title('Top 10 Mentioned Users')  plt.ylabel('Count')    plt.subplot(223)  sns.scatterplot(data=df, x='retweets', y='likes', hue='sentiment', palette='coolwarm')  plt.title('Retweets vs Likes (colored by sentiment)')    plt.subplot(224)  hourly\_sentiment.plot()  plt.title('Average Sentiment Over Time')  plt.ylabel('Sentiment')    plt.tight\_layout()  plt.show()    # Summary statistics  print(df[['retweets', 'likes', 'sentiment']].describe())    return df  # Example usage  tweet\_data = generate\_tweet\_data()  analyzed\_tweets = analyze\_tweet\_data(tweet\_data) | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**      **RESULT:**  Thus, the program for analyzing simulated tweet data to identify popular hashtags, user mentions, and sentiment trends using Python is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 4** | **Users Influential on Social Media Platforms** | **Date:** |
| **AIM:**  To perform link analysis on a social media platform by analyzing user interactions.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Import Necessary Libraries  **Step 3:** Define Social Media Interaction Data  **Step 4:** Build Graph Representation  **Step 5**: Visualize the Network  **Step 6:** Calculate Degree Centrality  **PROGRAM:**  import networkx as nx  import matplotlib.pyplot as plt  social\_media\_data = [  ('user1', 'user2', 'like'),  ('user1', 'user3', 'comment'),  ('user2', 'user1', 'share'),  ('user3', 'user2', 'like'),  ('user3', 'user1', 'like'),  ('user4', 'user1', 'comment'),  ('user4', 'user3', 'share'),  ]  G = nx.Graph()  for interaction in social\_media\_data:  user1, user2, interaction\_type = interaction  G.add\_edge(user1, user2, interaction\_type=interaction\_type)  pos = nx.spring\_layout(G) # Positions for all nodes  plt.figure(figsize=(10, 8))  nx.draw\_networkx\_nodes(G, pos, node\_color='skyblue', node\_size=1000)  nx.draw\_networkx\_edges(G, pos, edgelist=G.edges(), width=2, alpha=0.6, edge\_color='gray')  nx.draw\_networkx\_labels(G, pos, font\_size=12, font\_family='sans-serif')  plt.title('Social Media Network')  plt.axis('off')  plt.show()  degree\_centrality = nx.degree\_centrality(G)  print("Degree Centrality:")  for user, centrality in degree\_centrality.items():  print(f"{user}: {centrality}") | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**      **RESULT:**  Thus, the program to perform link analysis on a social media platform by analyzing user interactions is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 4(a)** | **Analyzing User Interaction and identify key influencers in a Simulated Social Media Platform** | **Date:** |
| **AIM:**  To analyze the connections between users based on their interactions and identify key influencers on a social media platform.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Collect data on user interactions (e.g., mentions, retweets)  **Step 3:** Create a graph representation of the network using networkx  **Step 4:** Calculate centrality measures to identify influential users.  **Step 5:** Visualize the network.  **Step 6:** Stop  **PROGRAM:**  import networkx as nx  import matplotlib.pyplot as plt  import pandas as pd  import random  def generate\_sample\_data(num\_users=50, num\_interactions=200):  users = [f"user\_{i}" for i in range(num\_users)] interactions = []  for \_ in range(num\_interactions):  source = random.choice(users)  target = random.choice(users)  if source != target:  interactions.append((source, target))  return pd.DataFrame(interactions, columns=['source', 'target'])  def perform\_link\_analysis(df):  G = nx.from\_pandas\_edgelist(df, 'source', 'target', create\_using=nx.DiGraph())    # Calculate centrality measures  pagerank = nx.pagerank(G)  in\_degree = dict(G.in\_degree())    # Identify top influential users  top\_pagerank = sorted(pagerank.items(), key=lambda x: x[1], reverse=True)[:5]  top\_in\_degree = sorted(in\_degree.items(), key=lambda x: x[1], reverse=True)[:5]    # Visualize the network  pos = nx.spring\_layout(G)  plt.figure(figsize=(12, 8))  nx.draw(G, pos, with\_labels=True, node\_size=10, node\_color='lightblue',  font\_size=8, arrows=True, alpha=0.6)  plt.title("User Interaction Network")  plt.axis('off')  plt.tight\_layout()  plt.show()    return top\_pagerank, top\_in\_degree  # Example usage  df = generate\_sample\_data()  top\_pagerank, top\_in\_degree = perform\_link\_analysis(df)  print("Top 5 users by PageRank:")  for user, score in top\_pagerank:  print(f"{user}: {score:.4f}")  print("\nTop 5 users by In-degree:")  for user, score in top\_in\_degree:  print(f"{user}: {score}") | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**      **RESULT:**  Thus, the program to analyze the connections between users based on their interactions and identify key influencers on a social media platform is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 4(b)** | **Analyzing Social Network Structures: Community Detection and Influencer Identification** | **Date:** |
| **AIM:**  To write a program for Analyzing Social Network Structures: Community Detection and Influencer Identification.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Collect data on user interactions (e.g., mentions, retweets).  **Step 3:** Create a graph representation of the network using network.  **Step 4:** Calculate centrality measures to identify influential users.  **Step 5:** Visualize the network.  **Step 6:** Stop.  **PROGRAM:**  import networkx as nx  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from community import community\_louvain  def generate\_social\_network(n\_users=100, n\_interactions=500):  np.random.seed(42)  users = [f'user\_{i}' for i in range(n\_users)]  interactions = [(np.random.choice(users), np.random.choice(users)) for \_ in range(n\_interactions)]  return pd.DataFrame(interactions, columns=['source', 'target'])  def analyze\_social\_network(df):  G = nx.from\_pandas\_edgelist(df, 'source', 'target', create\_using=nx.DiGraph())      # Calculate network metrics  in\_degree = dict(G.in\_degree())  out\_degree = dict(G.out\_degree())  pagerank = nx.pagerank(G)    # Community detection  communities = community\_louvain.best\_partition(G.to\_undirected())    # Visualize the network  pos = nx.spring\_layout(G)  plt.figure(figsize=(12, 8))  nx.draw(G, pos, node\_color=[communities[node] for node in G.nodes()],  node\_size=[v \* 500 for v in pagerank.values()],  with\_labels=False, alpha=0.7)    # Label top influencers  top\_influencers = sorted(pagerank, key=pagerank.get, reverse=True)[:5]  nx.draw\_networkx\_labels(G, pos, {node: node for node in top\_influencers}, font\_size=12)    plt.title("Social Network Structure")  plt.axis('off')  plt.colorbar(plt.cm.ScalarMappable(cmap=plt.cm.viridis),  label='Community')  plt.show()    # Print summary statistics  print("Network Summary:")  print(f"Number of nodes: {G.number\_of\_nodes()}")  print(f"Number of edges: {G.number\_of\_edges()}")  print(f"Number of communities: {len(set(communities.values()))}")  print("\nTop 5 Influencers (by PageRank):")  for user in top\_influencers:  print(f"{user}: PageRank = {pagerank[user]:.4f}, In-degree = {in\_degree[user]}, Out-degree = {out\_degree[user]}")    return G, communities, pagerank  # Example usage  social\_data = generate\_social\_network()  G, communities, pagerank = analyze\_social\_network(social\_data)  **OUTPUT:** | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **RESULT:**  Thus, the program for Analyzing Social Network Structures: Community Detection and Influencer Identification is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 5** | **Users Influential on Social media platforms** | **Date:** |
| **AIM:**  To write a program for Users Influential on Social media platforms.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Create a DataFrame with user data (followers and engagement rate).  **Step 3:** Calculate an influence score for each user.  **Step 4:** Identify the top 10 most influential users.  **Step 5:** Visualize the influence scores of the top users.  **Step 6:** Stop  **PROGRAM:**  import pandas as pd  import matplotlib.pyplot as plt  # Create a sample DataFrame  data = {  'user': ['User1', 'User2', 'User3', 'User4', 'User5', 'User6', 'User7', 'User8', 'User9', 'User10', 'User11', 'User12'],  'followers': [1000, 5000, 3000, 10000, 2000, 8000, 15000, 6000, 4000, 12000, 7000, 9000],  'engagement\_rate': [0.05, 0.02, 0.03, 0.01, 0.04, 0.02, 0.01, 0.03, 0.05, 0.02, 0.03, 0.02]  }  df = pd.DataFrame(data)  def calculate\_influence\_score(followers, engagement\_rate):  return followers \* engagement\_rate  df['influence\_score'] = df.apply(lambda row: calculate\_influence\_score(row['followers'], row['engagement\_rate']), axis=1)  top\_influential = df.nlargest(10, 'influence\_score')  plt.figure(figsize=(12, 6))  plt.bar(top\_influential['user'], top\_influential['influence\_score'])  plt.title('Top 10 Influential Users')  plt.xlabel('User')  plt.ylabel('Influence Score')  plt.xticks(rotation=45)  plt.tight\_layout()  plt.show()  # Print the top influential users and their scores  print(top\_influential[['user', 'influence\_score']]) | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**    **RESULT:**  Thus, the program for Users Influential on Social media platform is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 5(a)** | **Identifying influential users based on degree centrality, and visualizing the network structure on Social Media Platforms** | **Date:** |
| **AIM:**  To write a program to Identifying influential users based on degree centrality, and visualizing the network structure on Social Media Platforms.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Import Necessary Libraries  **Step 3:** Build Graph Representation.  **Step 4:** Calculate Centrality Metrics.  **Step 5:** Identify Influential Users  **Step 6:** Stop.  **PROGRAM:**  import networkx as nx  import matplotlib.pyplot as plt  social\_media\_data = [  ('user1', 'user2', 'like'),  ('user1', 'user3', 'comment'),  ('user2', 'user1', 'share'),  ('user3', 'user2', 'like'),  ('user3', 'user1', 'like'),  ('user4', 'user1', 'comment'),  ('user4', 'user3', 'share'),  ]  G = nx.Graph()  for interaction in social\_media\_data:  user1, user2, interaction\_type = interaction  G.add\_edge(user1, user2, interaction\_type=interaction\_type)  degree\_centrality = nx.degree\_centrality(G)  print("Influential Users:")  sorted\_users = sorted(degree\_centrality.items(), key=lambda x: x[1], reverse=True)  for user, centrality in sorted\_users:  print(f"{user}: Degree Centrality = {centrality}")  pos = nx.spring\_layout(G)  plt.figure(figsize=(10, 8))  nx.draw\_networkx\_nodes(G, pos, node\_color='skyblue', node\_size=1000)  nx.draw\_networkx\_edges(G, pos, edgelist=G.edges(), width=2, alpha=0.6, edge\_color='gray')  nx.draw\_networkx\_labels(G, pos, font\_size=12, font\_family='sans-serif')  plt.title('Social Media Network')  plt.axis('off')  plt.show() | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:**      **RESULT:**  Thus, the program to identifying influential users based on degree centrality, and visualizing the network structure on Social Media Platforms is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 5(b)** | **Analyzing Influential Social Media Users Based on Engagement Metrics** | **Date:** |
| **AIM:**  To write a program to identify and analyze influential users on a social media platform based on Engagement metrics.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Collect user data including follower count, engagement rate, and content reach.  **Step 3:** Calculate an influence score based on these metrics.  **Step 4:** Rank users by their influence score.  **Step 5:** Analyze the characteristics of top influential users:  **Step 6:** Stop  **PROGRAM:**  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  def generate\_sample\_user\_data(num\_users=1000):  np.random.seed(42)  data = {  'user\_id': range(1, num\_users + 1),  'followers': np.random.randint(100, 1000000, num\_users),  'avg\_likes': np.random.randint(10, 10000, num\_users),  'avg\_comments': np.random.randint(1, 1000, num\_users),  'posts\_per\_week': np.random.randint(1, 50, num\_users)  }  df = pd.DataFrame(data)  df['engagement\_rate'] = (df['avg\_likes'] + df['avg\_comments']) / df['followers'] \* 100  return df  def calculate\_influence\_score(df):  df['influence\_score'] = (  np.log(df['followers']) \* 0.5 +  df['engagement\_rate'] \* 0.3 +  np.log(df['posts\_per\_week']) \* 0.2  )  return df.sort\_values('influence\_score', ascending=False).reset\_index(drop=True)  def analyze\_influential\_users(df, top\_n=10):  top\_users = df.head(top\_n)    print(f"Top {top\_n} Influential Users:")  print(top\_users[['user\_id', 'followers', 'engagement\_rate', 'posts\_per\_week', 'influence\_score']])    # Visualize the relationship between followers and engagement rate  plt.figure(figsize=(10, 6))  plt.scatter(df['followers'], df['engagement\_rate'], alpha=0.5)  plt.scatter(top\_users['followers'], top\_users['engagement\_rate'], color='red', label='Top Influencers')  plt.xscale('log')  plt.xlabel('Followers (log scale)')  plt.ylabel('Engagement Rate (%)')  plt.title('Followers vs Engagement Rate')  plt.legend()  plt.show()  # Example usage  df = generate\_sample\_user\_data()  df\_ranked = calculate\_influence\_score(df)  analyze\_influential\_users(df\_ranked) | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **OUTPUT:** | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **RESULT:**  Thus, the program to identify and analyze influential users on a social media platform based on Engagement metrics is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 6** | **Implement an analytic application for Facebook/Twitter data to demonstrate Sentiment Analysis and Entity Recognition.** | **Date:** |
| **AIM:**  To write a program to implement an analytic application for Facebook/Twitter data to demonstrate Sentiment Analysis and Entity Recognition.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Create a DataFrame with sample text data.  **Step 3:** Perform sentiment analysis on each text using TextBlob.  **Step 4:** Extract named entities from each text using spaCy.  **Step 5:** Visualize the sentiment distribution.  **Step 6:** Count and display the most common entities by type.  **Step 7:** Stop  **PROGRAM:**  import pandas as pd  from textblob import TextBlob  import spacy  import matplotlib.pyplot as plt  # Load spaCy model  nlp = spacy.load("en\_core\_web\_sm")  # Create a sample DataFrame  data = {  'text': [  "I love this new product from Apple! It's amazing.",  "The service at the restaurant was terrible. I'm never going back.",  "Just visited New York City. The Empire State Building was impressive!",  "Google announced a new AI model today. Exciting times for tech!",  "The weather in London is rainy as usual. But I still love this city."  ]  }  df = pd.DataFrame(data)  def analyze\_sentiment(text):  return TextBlob(text).sentiment.polarity  def extract\_entities(text):  doc = nlp(text)  return [(ent.text, ent.label\_) for ent in doc.ents]  df['sentiment'] = df['text'].apply(analyze\_sentiment)  df['entities'] = df['text'].apply(extract\_entities)  # Visualize sentiment distribution  plt.figure(figsize=(10, 6))  plt.hist(df['sentiment'], bins=10, edgecolor='black')  plt.title('Sentiment Distribution')  plt.xlabel('Sentiment Score')  plt.ylabel('Frequency')  plt.show()  # Print most common entities  entity\_counts = {}  for entities in df['entities']:  for entity, label in entities:  if label not in entity\_counts:  entity\_counts[label] = {}  entity\_counts[label][entity] = entity\_counts[label].get(entity, 0) + 1  for label, entities in entity\_counts.items():  print(f"\nTop 5 {label} entities:")  sorted\_entities = sorted(entities.items(), key=lambda x: x[1], reverse=True)[:5]  for entity, count in sorted\_entities:  print(f"{entity}: {count}")  # Print the DataFrame  print("\nDataFrame with sentiment scores and entities:")  print(df)  **OUTPUT:**      **RESULT:**  Thus, the program to implement an analytic application for Facebook/Twitter data to demonstrate Sentiment Analysis and Entity Recognition is executed successfully and the output is verified. | | |

|  |  |  |
| --- | --- | --- |
| **Ex. No: 6(a)** | **Implement an analytic application for Instagram data to demonstrate Sentiment Analysis and Entity Recognition.** | **Date:** |
| **AIM:**  To write a program to implement an analytic application for Instagram data to demonstrate Sentiment Analysis and Entity Recognition.  **ALGORITHM:**  **Step 1:** Start  **Step 2:** Install and Import Necessary Libraries.  **Step 3:** Authenticate and Fetch Social Media Data.  **Step 4:** Perform Sentiment Analysis.  **Step 5:** Perform Entity Recognition.  **Step 6:** Store and Visualize Results.  **Step 7:** Stop  **PROGRAM:**  import networkx as nx  import matplotlib.pyplot as plt  instagram\_data = [  ('user1', 'user2', 'follow'),  ('user1', 'user3', 'follow'),  ('user2', 'user1', 'mention'),  ('user3', 'user1', 'reply'),  ('user3', 'user2', 'mention'),  ('user4', 'user1', 'follow'),  ('user4', 'user3', 'follow'),  ('user5', 'user4', 'follow'),  ('user5', 'user3', 'mention'),  ('user2', 'user3', 'like'),  ('user3', 'user1', 'comment'),  ('user1', 'user4', 'share')  ]  G = nx.DiGraph()  for interaction in instagram\_data:  user1, user2, interaction\_type = interaction  G.add\_edge(user1, user2, interaction\_type=interaction\_type)  degree\_centrality = nx.degree\_centrality(G)  sorted\_users = sorted(degree\_centrality.items(), key=lambda x: x[1], reverse=True)  print("Most Influential Users:")  for user, centrality in sorted\_users[:5]:  print(f"{user}: Degree Centrality = {centrality:.2f}")  pos = nx.spring\_layout(G)  plt.figure(figsize=(12, 10))  nx.draw\_networkx\_nodes(G, pos, node\_color='lightblue', node\_size=1500)  nx.draw\_networkx\_edges(G, pos, edgelist=G.edges(), width=2, alpha=0.6, edge\_color='gray', arrows=True)  nx.draw\_networkx\_labels(G, pos, font\_size=12, font\_family='sans-serif')  edge\_labels = {(user1, user2): interaction\_type for user1, user2, interaction\_type in instagram\_data}  nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=edge\_labels, font\_color='red')  plt.title('Instagram Interaction Network')  plt.axis('off')  plt.show()  **OUTPUT:**      **Result:**  Thus, the program to implement an analytic application for Instagram data to demonstrate Sentiment Analysis and Entity Recognition is executed successfully and the output is verified. | | |