**Practical 2**

**Aim:** Data Collection-Select the social media platforms of your choice (Twitter, Facebook, LinkedIn, YouTube, Web blogs etc) ,connect to and capture social media data for business ( scraping, crawling, parsing).

Code:-

import pandas as pd

import requests

from textblob import TextBlob

# Set YouTube video ID, maximum number of comments to retrieve, and API key

video\_id = "Q33TkQKlIMg"

max\_result = 50

api\_key = "AIzaSyC\_4xZTiNuz1O-Qu5kYnlg82riP30KRIxY"

# Retrieve video information

video\_info\_url = f"https://www.googleapis.com/youtube/v3/videos?part=id%2Csnippet&id={video\_id}&key={api\_key}"

video\_info\_response = requests.get(video\_info\_url)

video\_info\_data = video\_info\_response.json()

# Retrieve video comments

comments\_url = f"https://www.googleapis.com/youtube/v3/commentThreads?key={api\_key}&videoId={video\_id}&part=snippet&maxResults={max\_result}"

comments\_response = requests.get(comments\_url)

comments\_data = comments\_response.json()

# Extract comments from the comments\_data JSON

comments = []

for item in comments\_data['items']:

comment = item['snippet']['topLevelComment']['snippet']['textOriginal']

comments.append(comment)

# Define function to perform sentiment analysis on a given comment

def get\_comment\_sentiment(comment):

analysis = TextBlob(comment)

if analysis.sentiment.polarity > 0:

return "Positive"

elif analysis.sentiment.polarity == 0:

return "Neutral"

else:

return "Negative"

# Perform sentiment analysis on all comments

comment\_list = []

sentiment\_list = []

for comment in comments:

sentiment = get\_comment\_sentiment(comment)

comment\_list.append(comment)

sentiment\_list.append(sentiment)

print(f"{comment} : {sentiment}")

# Create DataFrame from comments and sentiments

sentiment\_df = pd.DataFrame({"Comments": comment\_list, "Sentiment": sentiment\_list})

# Save DataFrame to a CSV file

sentiment\_df.to\_csv("YouTube\_Comments\_Sentiment.csv", index=False)

**Practical 3**

**AIM:-** To create sociograms and analyze networks for the given datasets ("Les Misérables," "Airlines," and "Internet Core Routers"), we'll use the NetworkX library in Python. We'll visualize these networks using Matplotlib.

Code:-

import networkx as nx

import matplotlib.pyplot as plt

from networkx.algorithms import bipartite

# Step 1: Load the Les Misérables dataset

G = nx.les\_miserables\_graph()

# Step 2: Create one-mode and two-mode networks

# One-mode network (characters only)

one\_mode\_graph = G

# Create a bipartite two-mode network (characters and their communities)

# In this example, we'll assume "community" as groups of characters based on their centrality (for simplicity)

centrality = nx.degree\_centrality(G)

communities = {}

for node, cent in centrality.items():

group = int(cent \* 10) # Group based on centrality value

if group not in communities:

communities[group] = []

communities[group].append(node)

# Creating a bipartite graph

B = nx.Graph()

character\_nodes = list(G.nodes())

community\_nodes = list(communities.keys())

B.add\_nodes\_from(character\_nodes, bipartite=0)

B.add\_nodes\_from(community\_nodes, bipartite=1)

for group, chars in communities.items():

for char in chars:

B.add\_edge(char, group)

# Step 3: Generate sociograms

# One-mode network sociogram

plt.figure(figsize=(12, 12))

pos = nx.spring\_layout(one\_mode\_graph)

nx.draw(one\_mode\_graph, pos, with\_labels=True, node\_color="skyblue", edge\_color="gray", node\_size=500, font\_size=10)

plt.title("One-mode Network: Les Misérables Characters")

plt.show()

# Two-mode network sociogram

plt.figure(figsize=(12, 12))

pos = nx.spring\_layout(B)

node\_colors = ["skyblue" if node in character\_nodes else "lightgreen" for node in B.nodes()]

nx.draw(B, pos, with\_labels=True, node\_color=node\_colors, edge\_color="gray", node\_size=500, font\_size=10)

plt.title("Two-mode Network: Les Misérables Characters and Communities")

plt.show()

**Practical 4**

**Aim:-D**evelop Content (text, emoticons, image, audio, video) based social media analytics model for business. (e.g., Content Based Analysis: Topic, Issue, Trend, sentiment/opinion analysis, audio, video, image analytics)

**Code:-**

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.decomposition import LatentDirichletAllocation

import matplotlib.pyplot as plt

# Sample Data

texts = ["This is the first document.", "The Document is pathetic.", "This document is really nice", "Is this the first document?"]

# Vectorize the text

vectorizer = CountVectorizer(stop\_words='english')

X = vectorizer.fit\_transform(texts)

# Topic Modeling with LDA

lda = LatentDirichletAllocation(n\_components=2, random\_state=42)

lda.fit(X)

# Display topics

def display\_topics(model, feature\_names, no\_top\_words):

for topic\_idx, topic in enumerate(model.components\_):

print(f"Topic {topic\_idx}:")

print(" ".join([feature\_names[i] for i in topic.argsort()[:-no\_top\_words - 1:-1]]))

display\_topics(lda, vectorizer.get\_feature\_names\_out(), 3)

# Trend Analysis with TF-IDF

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english')

tfidf = tfidf\_vectorizer.fit\_transform(texts)

df = pd.DataFrame(tfidf.toarray(), columns=tfidf\_vectorizer.get\_feature\_names\_out())

print(df)

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

# Analyze sentiments

sentiments = [analyzer.polarity\_scores(text) for text in texts]

sentiment\_df = pd.DataFrame(sentiments)

print(sentiment\_df)

**Practical 5**

**AIM:-** To develop a structure-based social media analytics model for a business, we can focus on two key areas: community detection and influence analysis. These methods are crucial for understanding how information spreads within a social network and identifying key influencers who can drive engagement and growth.

**CODE:-**

import pandas as pd

import networkx as nx

import community as community\_louvain

import matplotlib.pyplot as plt

#install python\_louvain

# Step 1: Data Collection (Load the data)

# For demonstration, let's create a synthetic dataset

data = {

'source': ['A', 'A', 'B', 'C', 'D', 'E', 'E', 'F', 'F', 'G'],

'target': ['B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K']

}

df = pd.DataFrame(data)

# Step 2: Network Construction

G = nx.from\_pandas\_edgelist(df, 'source', 'target')

# Step 3: Community Detection

partition = community\_louvain.best\_partition(G)

# Add community information to nodes

for node, community in partition.items():

G.nodes[node]['community'] = community

# Step 4: Influence Analysis

pagerank = nx.pagerank(G)

nx.set\_node\_attributes(G, pagerank, 'pagerank')

# Step 5: Visualization

pos = nx.spring\_layout(G) # Layout for visualization

plt.figure(figsize=(12, 8))

# Draw nodes with community colors

colors = [partition[node] for node in G.nodes()]

nx.draw\_networkx\_nodes(G, pos, node\_size=500, node\_color=colors, cmap=plt.cm.jet)

nx.draw\_networkx\_edges(G, pos, alpha=0.5)

nx.draw\_networkx\_labels(G, pos, font\_size=12)

# Draw node labels for pagerank

pagerank\_labels = {node: f'{round(rank, 2)}' for node, rank in pagerank.items()}

nx.draw\_networkx\_labels(G, pos, labels=pagerank\_labels, font\_color='red')

plt.title('Social Network with Community Detection and Influence Analysis')

plt.show()

**Practical 6**

**Aim:-**Develop a dashboard and reporting tool based on real time social media data Using Power BI

**CODE:-**

import dash

from dash import html, dcc

import pandas as pd

from ntscraper import Nitter

# Initialize Nitter scraper

scraper = Nitter(0)

# Function to fetch tweets

def get\_tweets(name, modes, no):

tweets = scraper.get\_tweets(name, mode=modes, number=no)

final\_tweets = []

for x in tweets['tweets']:

data = [x['link'], x['text'], x['date'], x['stats']['likes'], x['stats']['comments']]

final\_tweets.append(data)

dat = pd.DataFrame(final\_tweets, columns=['twitter\_link', 'text', 'date', 'likes', 'comments'])

return dat

# Fetch data

data = get\_tweets('World cup 2023', 'term', 10)

# Initialize Dash app

app = dash.Dash(\_\_name\_\_)

# Define layout

app.layout = html.Div([

html.H1("Twitter Dashboard"),

dcc.Graph(

id='tweets-graph',

figure={

'data': [

{'x': data['date'], 'y': data['likes'], 'type': 'bar', 'name': 'Likes'},

{'x': data['date'], 'y': data['comments'], 'type': 'bar', 'name': 'Comments'}

],

'layout': {

'title': 'Likes and Comments Over Time'

}

}

),

html.Table([

html.Thead(

html.Tr([html.Th(col) for col in data.columns])

),

html.Tbody([

html.Tr([

html.Td(data.iloc[i][col]) for col in data.columns

]) for i in range(len(data))

])

])

])

# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server(debug=True)

**Practical 7**

**Aim:-**Use Google Visualization Charts to analyze social media data

**Code:-**

import tweepy

import pandas as pd

from jinja2 import Template

# Twitter API credentials

API\_KEY = '78hRQZhIIM9BXPSguKfLjcgd9'

API\_SECRET\_KEY = 'VcBeUPgTLya9wm2VnOee8F0zKSQXlH7NAJKaICddp3uaYf2Udv'

ACCESS\_TOKEN = '840343727746904064-HBrbIYuBoxmpHrSXxOq9hqmL5hJzchl'

ACCESS\_TOKEN\_SECRET = 'jgE5tYbkCunlibk3vUpZiXRNAgLj6RudPXwDJnvUEABUW'

# Authenticate with the Twitter API

auth = tweepy.OAuthHandler(API\_KEY, API\_SECRET\_KEY)

auth.set\_access\_token(ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET)

api = tweepy.API(auth, wait\_on\_rate\_limit=True)

# Function to fetch tweets using Tweepy

def fetch\_tweets(query, num\_tweets):

tweets\_list = []

for tweet in tweepy.Cursor(api.search\_tweets, q=query, lang="en", tweet\_mode='extended').items(num\_tweets):

tweets\_list.append([

f"https://twitter.com/twitter/statuses/{tweet.id}",

tweet.full\_text,

tweet.created\_at.strftime('%Y-%m-%d'),

tweet.favorite\_count,

tweet.retweet\_count

])

df = pd.DataFrame(tweets\_list, columns=['twitter\_link', 'text', 'date', 'likes', 'comments'])

return df

# Fetch tweets with the term 'World Cup 2023'

data = fetch\_tweets('World Cup 2023', 10)

# HTML template for Google Visualization Charts

html\_template = """

<!DOCTYPE html>

<html>

<head>

<title>Social Media Data Visualization</title>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript">

google.charts.load('current', {'packages':['corechart', 'table']});

google.charts.setOnLoadCallback(drawCharts);

function drawCharts() {

var lineData = new google.visualization.DataTable();

lineData.addColumn('string', 'Date');

lineData.addColumn('number', 'Likes');

lineData.addRows([

{% for row in data.itertuples() %}

['{{ row.date }}', {{ row.likes }}],

{% endfor %}

]);

var lineOptions = {

title: 'Likes Over Time',

curveType: 'function',

legend: { position: 'bottom' }

};

var lineChart = new google.visualization.LineChart(document.getElementById('line\_chart'));

lineChart.draw(lineData, lineOptions);

var pieData = new google.visualization.DataTable();

pieData.addColumn('string', 'Tweet');

pieData.addColumn('number', 'Comments');

pieData.addRows([

{% for row in data.itertuples() %}

['{{ row.text|truncate(50) }}', {{ row.comments }}],

{% endfor %}

]);

var pieOptions = {

title: 'Comments Distribution',

is3D: true,

};

var pieChart = new google.visualization.PieChart(document.getElementById('pie\_chart'));

pieChart.draw(pieData, pieOptions);

var barData = new google.visualization.DataTable();

barData.addColumn('string', 'Tweet');

barData.addColumn('number', 'Likes');

barData.addRows([

{% for row in data.itertuples() %}

['{{ row.text|truncate(50) }}', {{ row.likes }}],

{% endfor %}

]);

var barOptions = {

title: 'Likes Distribution',

chartArea: {width: '50%'},

hAxis: {

title: 'Likes',

minValue: 0

},

vAxis: {

title: 'Tweet'

}

};

var barChart = new google.visualization.BarChart(document.getElementById('bar\_chart'));

barChart.draw(barData, barOptions);

var tableData = new google.visualization.DataTable();

tableData.addColumn('string', 'Tweet');

tableData.addColumn('string', 'Date');

tableData.addColumn('number', 'Likes');

tableData.addColumn('number', 'Comments');

tableData.addRows([

{% for row in data.itertuples() %}

['<a href="{{ row.twitter\_link }}" target="\_blank">{{ row.text|truncate(50) }}</a>', '{{ row.date }}', {{ row.likes }}, {{ row.comments }}],

{% endfor %}

]);

var table = new google.visualization.Table(document.getElementById('table\_div'));

table.draw(tableData, {showRowNumber: true, width: '100%', height: '100%'});

}

</script>

</head>

<body>

<h1>Social Media Data Visualization</h1>

<div id="line\_chart" style="width: 900px; height: 500px;"></div>

<div id="pie\_chart" style="width: 900px; height: 500px;"></div>

<div id="bar\_chart" style="width: 900px; height: 500px;"></div>

<div id="table\_div" style="width: 900px; height: 500px;"></div>

</body>

</html>

"""

# Render the template with the data

template = Template(html\_template)

html\_content = template.render(data=data)

# Save the rendered HTML to a file with UTF-8 encoding

file\_path = 'visualization.html'

with open(file\_path, 'w', encoding='utf-8') as file:

file.write(html\_content)

print("HTML file created successfully.")

**Practical 8 - GNN**

Code:-

import torch

import torch.nn.functional as F

from torch\_geometric.datasets import Planetoid

import torch\_geometric.transforms as T

from torch\_geometric.nn import GCNConv

# Load the Cora dataset

dataset = Planetoid(root='/tmp/Cora', name='Cora', transform=T.NormalizeFeatures())

data = dataset[0]

print(f'Dataset: {dataset}:')

print('======================')

print(f'Number of graphs: {len(dataset)}')

print(f'Number of features: {dataset.num\_features}')

print(f'Number of classes: {dataset.num\_classes}')

class GCN(torch.nn.Module):

def \_\_init\_\_(self):

super(GCN, self).\_\_init\_\_()

self.conv1 = GCNConv(dataset.num\_features, 16)

self.conv2 = GCNConv(16, dataset.num\_classes)

def forward(self, data):

x, edge\_index = data.x, data.edge\_index

x = self.conv1(x, edge\_index)

x = F.relu(x)

x = F.dropout(x, training=self.training)

x = self.conv2(x, edge\_index)

return F.log\_softmax(x, dim=1)

model = GCN()

print(model)

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model = model.to(device)

data = data.to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight\_decay=5e-4)

def train():

model.train()

optimizer.zero\_grad()

out = model(data)

loss = F.nll\_loss(out[data.train\_mask], data.y[data.train\_mask])

loss.backward()

optimizer.step()

return loss.item()

def test():

model.eval()

logits, accs = model(data), []

for mask in [data.train\_mask, data.val\_mask, data.test\_mask]:

pred = logits[mask].max(1)[1]

acc = pred.eq(data.y[mask]).sum().item() / mask.sum().item()

accs.append(acc)

return accs

for epoch in range(200):

loss = train()

train\_acc, val\_acc, test\_acc = test()

print(f'Epoch: {epoch:03d}, Loss: {loss:.4f}, Train Acc: {train\_acc:.4f}, Val Acc: {val\_acc:.4f}, Test Acc: {test\_acc:.4f}')

model.eval()

logits = model(data)

pred = logits[data.test\_mask].max(1)[1]

test\_acc = pred.eq(data.y[data.test\_mask]).sum().item() / data.test\_mask.sum().item()

print(f'Test Accuracy: {test\_acc:.4f}')

**Practical 9**

**Aim:-** Analyze Twitter conversations to identify the most active and influential users using Machine Learning Algorithms with Gephi Tool.

Code:-

import networkx as nx

import matplotlib.pyplot as plt

# Sample data (replace with your own data)

edges = [("user1", "user2"), ("user1", "user3"), ("user2", "user3"),

("user2", "user4")]

# Create a directed graph

G = nx.DiGraph()

# Add edges to the graph

G.add\_edges\_from(edges)

# Visualize the network

nx.draw(G, with\_labels=True)

plt.show()

# Calculate degree centrality

degree\_centrality = nx.degree\_centrality(G)

print("Degree Centrality:", degree\_centrality)

# Calculate betweenness centrality

betweenness\_centrality = nx.betweenness\_centrality(G)

print("Betweenness Centrality:", betweenness\_centrality)