## **EXPERIMENT NO. 3**

**Aim:** Data Cleaning and Storage – Preprocess, filter and store social media data for business. (Using Python, MongoDB, R, etc.)

**Theory:**

Scrape the data from twitter and then perform different techniques of cleaning, preprocessing and then store it for further analysis.

**Snscrape:**

snscrape is a scraper for social networking services (SNS). It scrapes things like user profiles, hashtags, or searches and returns the discovered items, e.g. the relevant posts.

The following services are currently supported:

* Facebook: user profiles, groups, and communities (aka visitor posts)
* Instagram: user profiles, hashtags, and locations
* Mastodon: user profiles and toots (single or thread)
* Reddit: users, subreddits, and searches (via Pushshift)
* Telegram: channels
* Twitter: users, user profiles, hashtags, searches (live tweets, top tweets, and users), tweets (single or surrounding thread), list posts, communities, and trends
* VKontakte: user profiles
* Weibo (Sina Weibo): user profiles

**Requirements**

snscrape requires Python 3.8 or higher. The Python package dependencies are installed automatically when you install snscrape.

!pip install snscrape

import pandas as pd

import snscrape.modules.twitter as sntwitter

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import nltk nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import WordNetLemmatizer

from nltk.stem.porter import PorterStemmer

import string

import re

import textblob

from textblob import TextBlob

import os

from wordcloud import WordCloud, STOPWORDS

from wordcloud import ImageColorGenerator

import warnings

%matplotlib inline

1. **Scrape Twitter Data for Union Budget 2023:**

#using OS library to call CLI command in python

os.system("snscrape --jsonl --max-results 5000 --since 2023-01-31 twitter-search 'Chatgpt until:2023-02-07'>text-query-tweets.json")

#Read the JSON generated from the CLI command above and create a pandas

dataframe tweets\_df = pd.read\_json("text-query-tweets.json" ,lines=True)

tweets\_df.head(5)

tweets\_df.to\_csv()

1. **Data Loading:**

df1 = tweets\_df[[ 'date', 'rawContent' , 'renderedContent' , 'user' , 'replyCount' ,'retweetCount' , 'likeCount' , 'lang' , 'place' , 'hashtags' , 'viewCount']].copy()

df1.head()

df1.shape

1. **Twitter Data Cleaning and Preprocessing:**

from nltk. corpus import stopwords

stop = stopwords.words('english')

df1['renderedContent'].apply(lambda x: [item for item in x if item not in stop])

df1.shape

!pip install tweet-preprocessor

#Remove unnecessary characters punct = ['%','/',':','\\','&amp','&',';','?']

def remove\_punctuations(text):

for punctuation in punct:

text = text.replace(punctuation,'')

return text

df1['renderedContent'] = df1['renderedContent'].apply(lambda x: remove\_punctuations(x))

df1['renderedContent'].replace( '', np.nan, inplace=True)

df1.dropna(subset=["renderedContent"],inplace=True)

len(df1)

df1 = df1.reset\_index(drop=True)

df1.head()

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

sns.set\_style('whitegrid')

%matplotlib inline

stop=stop+['budget2023' , 'budget' , 'httpst' , '2023', 'modi' ,'nsitaraman' , 'union', 'pmindia' , 'tax' , 'india']

def plot\_20\_most\_common\_words(count\_data, count\_vectorizer) : import matplotlib. pyplot as plt

words = count\_vectorizer.get\_feature\_names() total\_counts = np. zeros(len(words))

for t in count\_data:

total\_counts = t.toarray()[0]

count\_dict = (zip(words, total\_counts))

count\_dict = sorted(count\_dict, key=lambda x:x[1],reverse=True)[0:20]

words = [w[0] for w in count\_dict]

counts = [w[1] for w in count\_dict]

x\_pos = np.arange(len(words))

plt.figure(2, (40,40))

plt.subplot(title = '20 most common words')

sns. set\_context('notebook',font\_scale=4,rc={ 'lines.linewidth' :2.5})

sns.barplot(x\_pos, counts, palette='husl')

plt.xticks(x\_pos, words, rotation=90)

plt.xlabel('words')

plt.ylabel('counts')

plt.show()

count\_vectorizer = CountVectorizer(stop\_words=stop) # Fit and transform the processed titles

count\_data = count\_vectorizer.fit\_transform(df1['renderedContent'])

# print(count\_vectorizer)

# print(count\_data)

# Visualise the 20 most common words plot\_20\_most\_common\_words(count\_data,count\_vectorizer) plt.savefig( 'saved\_figure.png')

import cufflinks as cf cf.go\_offline()

cf.set\_config\_file(offline=False, world\_readable=True)

def get\_top\_n\_bigram(corpus, n=None) :

vec = CountVectorizer(ngram\_range=(2, 4), stop\_words="english").fit(corpus) bag\_of\_words = vec.transform(corpus)

sum\_words = bag\_of\_words.sum(axis=0)

words\_freq =[(word, sum\_words[0, idx]) for word, idx in vec.vocabulary\_.items()] words\_freq =sorted(words\_freq, key = lambda x: x[1], reverse=True)

return words\_freq[:n]

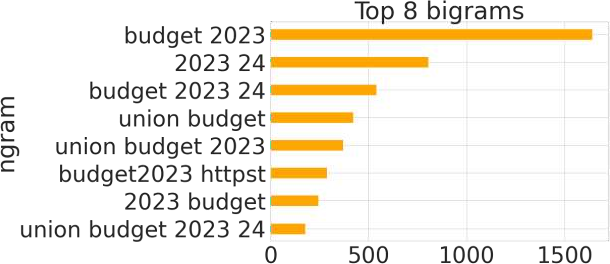
common\_words = get\_top\_n\_bigram(df1['renderedContent'] , 8) mydict={}

for word, freq in common\_words:

bigram\_df = pd.DataFrame(common\_words,columns = ['ngram', 'count'])

bigram\_df.groupby('ngram').sum()['count'].sort\_values(ascending=False).sort\_values().plot.barh(title = 'Top 8 bigrams',color='orange' , width=.4, figsize=(12,8),stacked = True)

**Output:**



**Conclusion:**

By scraping data from twitter using snscrape, further it was cleaned, preprocessed and stored for further analysis.