Understanding the Impact of Ideological Bias on NLP Tasks

CSE 472 Project 2

Submitted by

Ayushi Nirmal 1225466320
Muhammed Shanfer Majeed 1225290521

Under the Guidance of

Dr. Huan Liu
Professor
&
Zhen Tan
Teaching Assistant
ztan36@asu.edu

DECLARATION

I hereby declare that the project report for **Understanding the impact of ideological bias on NLP Tasks**, is my original work and to my knowledge, has not been published before. The references and assumptions for the abstract have been defined in the report for each distribution.

Place : Tempe, Arizona.
Date : 26 September 2022

Name: Ayushi Nirmal 1225466320
Muhammed Shanfer Majeed 1225290521

ABSTRACT

For the purpose of advancing a political agenda, journalists who are ideologically inclined purposefully publish biased news stories. For political reasons, they could, for instance, report just one side of a story or incident. According to a study, consuming politically slanted news can influence how one interprets specific facts or information. Recently with the great success of word embedding techniques and language models such as Word2Vec, GLoVe, ELMo and Bert, Natural Language Processing (NLP) has seen significant growth and advancement. Many downstream tasks, including topic classification, fake news detection, and sentiment analysis, have embraced them. The research community does not, however, fully comprehend how the ideological bias in the training data influences the model. The extent to which ideological bias affects certain NLP tasks is the aim of this project.In this project, information was gathered from major media's official Twitter accounts across the political spectrum, including CNN (left), Fox News (right), and Reuters(neutral), in order to obtain distinct perspectives from various writers. And then Topic classification experiment was designed to run state of the art text classification models on the collected datasets. Later, the models are evaluated by training them on one type of dataset and then testing them on another. To comprehend the effect of ideological bias on the model, we have chosen a variety of assessment measures, focusing primarily on the model's accuracy.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Huan Liu and Zhen Tan for providing us this project to work upon. The project has helped me understand social media networks between people and the spread of biases among people. The project gave a real insight into the working of different kinds of state of the art NLP models that are existing in the deep learning world. Through this project, I have also had the experience to understand to what extent ideological bias has an impact on the NLP tasks.

TABLE OF CONTENTS

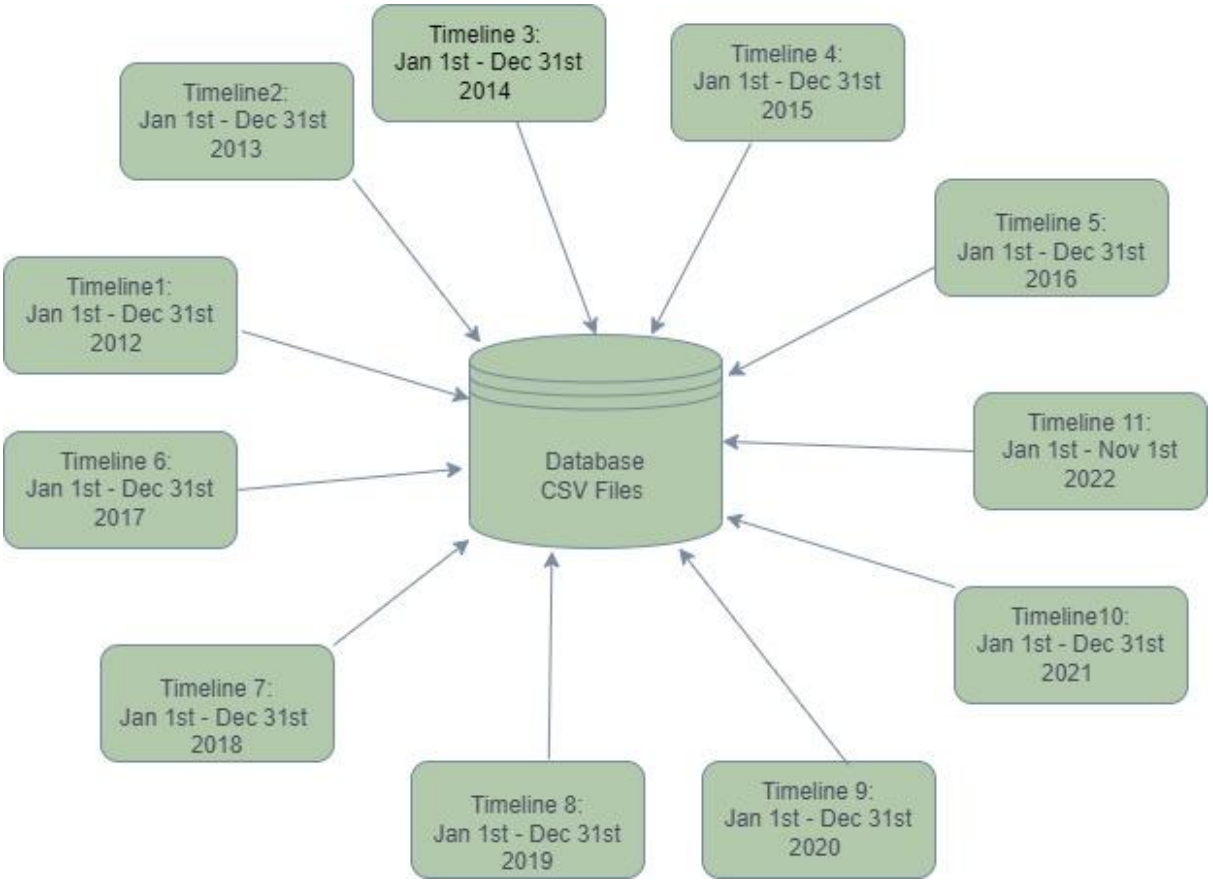
Chapter 1. Data Collection:

1.1 Introduction:

The data was gathered from several timelines on each of the Twitter accounts for CNN, Fox News, and Reuters. Different csv files with the data from each twitter handle were created. The whole project's database will be based on csv files.

Using a third-party library called tweepy, the information was gathered from Twitter. The api utilized to obtain user timeline information is **tweepy.cursor(api.user_timeline).**

For two reasons, data was collected from several timeframes. The first argument has to do with obtaining access to various mainstream media authors' writing styles throughout various time periods. The 11 years up to the present were used to create the timeline. The second reason is due to a restriction we encountered, which was that we could only get 3200 tweets per request from a user handle at once. According to the chronology shown below, the facts are gathered in chronological order from the past to the present.



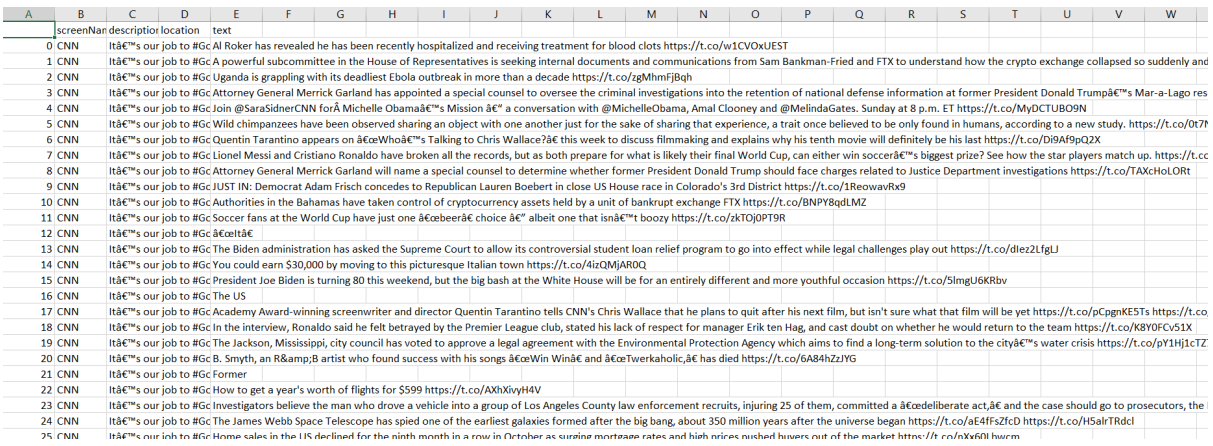
1.2 Fields collected per tweets are in format:

Index	ScreenName	Description	Location	Text

1.3. Number of tweets scraped from each twitter handle:

Screen Name	Number of Tweets
CNN (left)	26160 Tweets
Fox News (Right)	35746 Tweets
Reuters News (Neutral)	27134 Tweets

1.4 CSV Demo from CNN:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
		screenNan	descriptio	location	text																		
0	CNN	Itâ€™s our job to #Gc Al Roker has revealed he has been recently hospitalized and receiving treatment for blood clots https://t.co/w1CVOxUEST																					
1	CNN	Itâ€™s our job to #Gc A powerful subcommittee in the House of Representatives is seeking internal documents and communications from Sam Bankman-Fried and FTX to understand how the crypto exchange collapsed so suddenly and																					
2	CNN	Itâ€™s our job to #Gc Uganda is grappling with its deadliest Ebola outbreak in more than a decade https://t.co/zgMhmFjBqh																					
3	CNN	Itâ€™s our job to #Gc Attorney General Merrick Garland has appointed a special counsel to oversee the criminal investigations into the retention of national defense information at former President Donald Trumpâ€™s Mar-a-Lago res																					
4	CNN	Itâ€™s our job to #Gc Join @SaraSidnerCNN forÂ Michelle Obamaâ€™s Mission â€“ a conversation with @MichelleObama, Amal Clooney and @MelindaGates. Sunday at 8 p.m. ET https://t.co/MyDCTUBO9N																					
5	CNN	Itâ€™s our job to #Gc Wild chimpanzees have been observed sharing an object with one another just for the sake of sharing that experience, a trait once believed to be only found in humans, according to a new study. https://t.co/0t7																					
6	CNN	Itâ€™s our job to #Gc Quentin Tarantino appears on â€œWhoâ€™s Talking to Chris Wallace?â€ this week to discuss filmmaking and explains why his tenth movie will definitely be his last https://t.co/Di9Af9pQ2X																					
7	CNN	Itâ€™s our job to #Gc Lionel Messi and Cristiano Ronaldo have broken all the records, but as both prepare for what is likely their final World Cup, can either win soccerâ€™s biggest prize? See how the star players match up. https://t.co																					
8	CNN	Itâ€™s our job to #Gc Attorney General Merrick Garland will name a special counsel to determine whether former President Donald Trump should face charges related to Justice Department investigations https://t.co/TAXcHoLORt																					
9	CNN	Itâ€™s our job to #Gc JUST IN: Democrat Adam Frisch concedes to Republican Lauren Boebert in close US House race in Colorado's 3rd District https://t.co/1ReowavRx9																					
10	CNN	Itâ€™s our job to #Gc Authorities in the Bahamas have taken control of cryptocurrency assets held by a unit of bankrupt exchange FTX https://t.co/BNPY8qdLMZ																					
11	CNN	Itâ€™s our job to #Gc Soccer fans at the World Cup have just one â€œbeerâ€ choice â€“ albeit one that isnâ€™t boozy https://t.co/zkTOj0PT9R																					
12	CNN	Itâ€™s our job to #Gc â€œltâ€																					
13	CNN	Itâ€™s our job to #Gc The Biden administration has asked the Supreme Court to allow its controversial student loan relief program to go into effect while legal challenges play out https://t.co/dIez2LfgLJ																					
14	CNN	Itâ€™s our job to #Gc You could earn $30,000 by moving to this picturesque Italian town https://t.co/4izQMjAR0Q																					
15	CNN	Itâ€™s our job to #Gc President Joe Biden is turning 80 this weekend, but the big bash at the White House will be for an entirely different and more youthful occasion https://t.co/5ImgU6KRbv																					
16	CNN	Itâ€™s our job to #Gc The US																					
17	CNN	Itâ€™s our job to #Gc Academy Award-winning screenwriter and director Quentin Tarantino tells CNN's Chris Wallace that he plans to quit after his next film, but isn't sure what that film will be yet https://t.co/pCpgnKE5Ts https://t.co																					
18	CNN	Itâ€™s our job to #Gc In the interview, Ronaldo said he felt betrayed by the Premier League club, stated his lack of respect for manager Erik ten Hag, and cast doubt on whether he would return to the team https://t.co/K8Y0FCv51X																					
19	CNN	Itâ€™s our job to #Gc The Jackson, Mississippi, city council has voted to approve a legal agreement with the Environmental Protection Agency which aims to find a long-term solution to the cityâ€™s water crisis https://t.co/pY1Hj1cTZ																					
20	CNN	Itâ€™s our job to #Gc B. Smyth, an R&B artist who found success with his songs â€œWin Winâ€ and â€œTwerkaholic,â€ has died https://t.co/6A84hZzJYG																					
21	CNN	Itâ€™s our job to #Gc Former																					
22	CNN	Itâ€™s our job to #Gc How to get a year's worth of flights for $599 https://t.co/AXhXivyH4V																					
23	CNN	Itâ€™s our job to #Gc Investigators believe the man who drove a vehicle into a group of Los Angeles County law enforcement recruits, injuring 25 of them, committed a â€œdeliberate act,â€ and the case should go to prosecutors, the																					
24	CNN	Itâ€™s our job to #Gc The James Webb Space Telescope has spied one of the earliest galaxies formed after the big bang, about 350 million years after the universe began https://t.co/aE4fFsZfcD https://t.co/H5alrTRdcl																					
25	CNN	Itâ€™s our job to #Gc Home sales in the US declined for the ninth month in a row in October as surging mortgage rates and high prices pushed buyers out of the market https://t.co/pXx60Lhwcm																					

1.5 Implementation of data collection:

```python
import tweepy
import pandas as pd
```

Python

```python
def get_api():
    api_key = '8RQ5xPiDzAg1duMVpj9pWYEdX'
    api_key_secret = 'BqfcElqH5G6yAFHafuMQWO5ljp8Bosa402DzldWgc3mzCOSrSB'
    access_token = '1570146620653850629-erndFgwUuLXuK8prEhhrpkqlBUICih'
    access_token_secret = 'hwjy8IzsfECPVje0SAzdg3YorFifjAZbpBBgmLxBxnGVc'
    auth = tweepy.OAuthHandler(api_key, api_key_secret)
    auth.set_access_token(access_token, access_token_secret)
    return tweepy.API(auth,wait_on_rate_limit=True)
```

Python

```python
cols=['screenName','description','location','text']
```

Python

```python
def collect_User_TimelineData(username,start_date,end_date):
  api = get_api()
  db = pd.DataFrame(columns=cols, dtype=object)
  for tweet in tweepy.Cursor(api.user_timeline, screen_name=username, since=start_date,until=end_date,tweet_mode='extended').items(10000):
      username = tweet.user.screen_name
      description = tweet.user.description
      location = tweet.user.location
      tweet_text = tweet.full_text
      ith_tweet = [username, description,
                   location, tweet_text]
      db.loc[len(db)] = ith_tweet
  return db
```

```python
dates=[('2012-01-01','2012-12-31'),('2013-01-01','2013-12-31'),('2014-01-01','2014-12-31'),('2015-01-01','2015-12-31'),('2016-01-01','2016-12-31'),(
```

Python

```python
df= pd.DataFrame(columns=cols, dtype=object)
for x in dates:
  df=df.append(collect_User_TimelineData('CNN',x[0],x[1]),ignore_index=True)
```

Python

```python
df.to_csv('CNN_1000_data_with_dates2.csv')
```

Python

```python
df.shape
```

Python

```
(26160, 4)
```

```python
df_fox= pd.DataFrame(columns=cols, dtype=object)
for x in dates:
  df_fox=df_fox.append(collect_User_TimelineData('FoxNews',x[0],x[1]),ignore_index=True)
```

Python

```python
df_fox.to_csv('FOX_NEWS_1000_data_with_dates2.csv')
```

Python

```python
df_fox.shape
```

[16] Python

```
(35746, 4)
```

```python
df_Reuters= pd.DataFrame(columns=cols, dtype=object)
for x in dates:
  df_Reuters=df_Reuters.append(collect_User_TimelineData('Reuters',x[0],x[1]),ignore_index=True)
```

[17] Python

```python
df_Reuters.to_csv('Reuters_1000_data_with_dates2.csv')
```

[18] Python

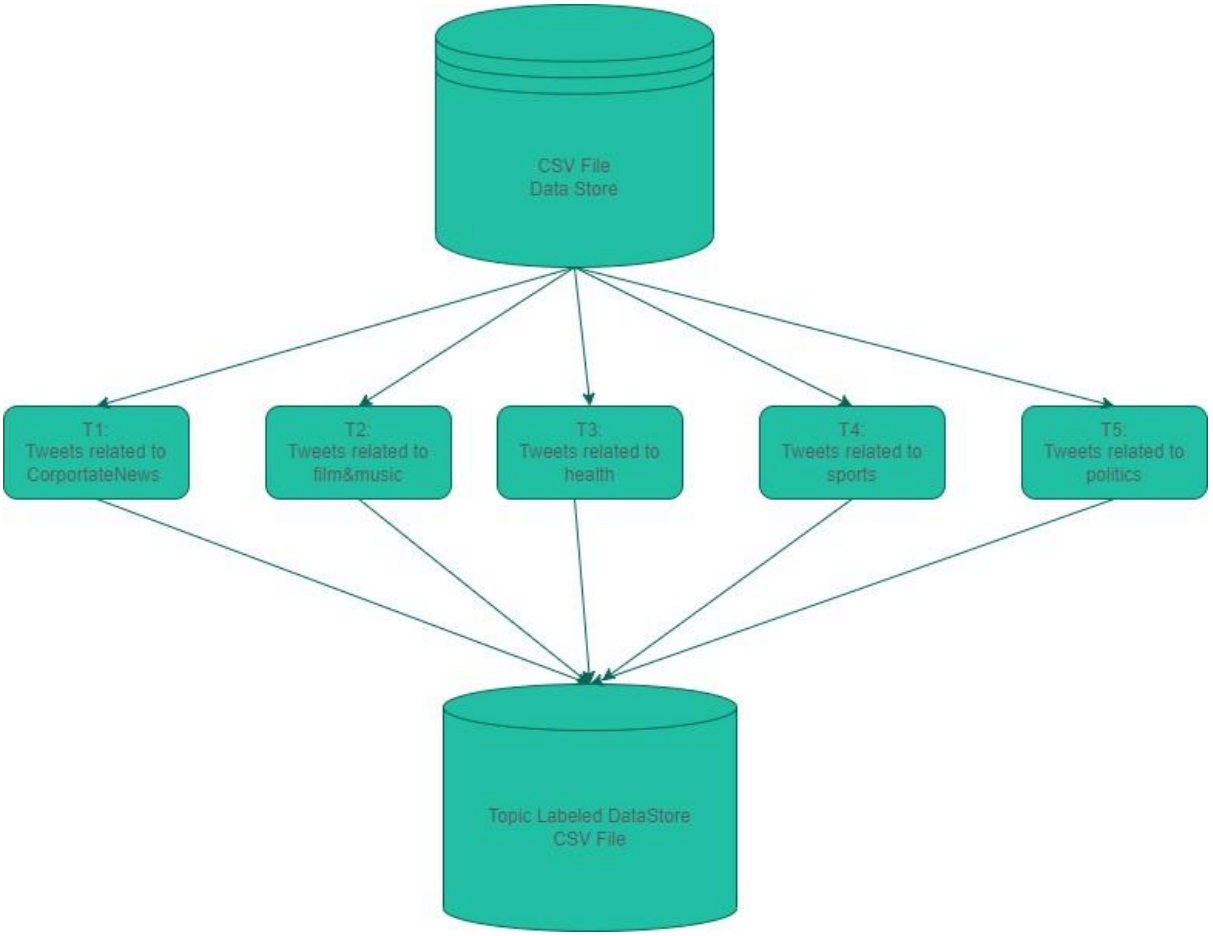
```python
df_Reuters.shape
```

[19] Python

```
(27134, 4)
```

Chapter 2. Topic Labeling:

The collected data need to be labeled into different topics in order to get ground truth which is an integral part of designing a NLP based classification model. For doing that we collected keywords from each of the datasets saved in csv files into five topics as depicted below in the figure:



Each tweet is labeled to that topic which does have the corresponding maximum number of keywords present in that tweet text. This is the strategy we are utilizing for topic labeling.

2.1 Keywords used for labeling the tweets:

```
health=['depression','disease','Alzheimer', 'medication','medical',
'hospital', 'vaccine','covid',  'healthcare', 'drug','overdose',
'doctors','patients','consult','ebola', 'treatment', 'covid-19',
'sperm','doctor','recipe','fast food','genealogist','Genetic','gene',
'blood' ,'hospitalized' , 'mental
health','prescriptions','hospitals','respitatory', 'smoking
weed','tobbaco smocker','ebola', 'virus','bacteria']
politics=['lapd','supreme court','Democrat', 'Biden','trump',
'federal','election','democratic','white house','congress','crown
prince','Bush','Obama','FBI','governor','government','senate','candidat
e', 'politicians', 'freedom', 'President', 'politics', 'governance',
'presidency','republicans','Democrats','legistative','prime
minister','federal','world
war','trial','lawsuit','attorney','missiles','sheriff','crime']
sports=['football','World cup','tournament', 'Raphael Warnock'
,'Herschel Walker','Olympics','game','postgame',
'game-winner','player','Bowlers','cricket','bowling','batsman','world
cup 2022', 'Fiat 500e','race' ,'tom brady','Nick Saban','trophey',
'Messi', 'Ronaldo' ,'soccer','Premier League','FIFA', 'world
champion','NBA' ,'sports car',"Shaquille O'Neal",'basketball']
filmandmusic=['filming','science-fiction','drama','web-series','taylor
swift','musician','country music','music','artists','jessie james
decker','Louis Tomlinson','one direction', 'harry
style','filmmaking','film', 'movie','screenwriter', 'Academy
Award-winning' ,'TV Special', 'nick jonas', 'jackie chan', 'tom hardy',
'avengers', 'marvel', 'nolan', 'superman', 'batman']
corporatenews=['Red Bull','AIR FORCE','companies','twitter','elon
musk', 'Airbnb','amazon','CEO','company', 'Disney World','Walmart',
'HBO']
```

2.2 Number of Tweets per Topic:

Screen Name	Topic Name	Number of Tweets
CNN	Corporate News	237
	Film	113
	Health	151
	Sports	184
	politics	832
FoxNews	CorporateNews	114
	Film	48
	Health	118
	Sports	253
	politics	1051
Reuters	CorporateNews	113
	Film	56
	Health	120
	Sports	249
	Politics	1013

2.3 Implementation of Topic Labeling:

```
import tweepy
import pandas as pd
```
[59] Python

+ Code + Markdown

```
cols=['screenName','description','location','text']
```
[61] Python

#Topic Labeling into five categories

```
health=['depression','disease','Alzheimer', 'medication','medical', 'hospital', 'vaccine','covid',  'healthcare', 'drug','overdose', 'doctors','pati
politics=['lapd','supreme court','Democrat', 'Biden','trump', 'federal','election','democratic','white house','congress','crown prince','Bush','Oban
sports=['football','World cup','tournament', 'Raphael Warnock' ,'Herschel Walker','Olympics','game','postgame', 'game-winner','player','Bowlers','cr
filmandmusic=['filming','science-fiction','drama','web-series','taylor swift','musician','country music','music','artists','jessie james decker','Lo
corporatenews=['Red Bull','AIR FORCE','companies','twitter','elon musk', 'Airbnb','amazon','CEO','company', 'Disney World','Walmart', 'HBO']
df_cnn_health= pd.DataFrame(columns=cols, dtype=object)
df_cnn_politics= pd.DataFrame(columns=cols, dtype=object)
df_cnn_sports= pd.DataFrame(columns=cols, dtype=object)
df_cnn_film=pd.DataFrame(columns=cols,dtype=object)
df_cnn_corporate=pd.DataFrame(columns=cols,dtype=object)
df_cnn_list=[df_cnn_corporate,df_cnn_film,df_cnn_health,df_cnn_sports,df_cnn_politics]
keywords=[corporatenews,filmandmusic,health,sports,politics]
topics_list=['corporatenews','film','health','sports','politics']
```
[67] Python

```
topicdict={}
for i in range(len(keywords)):
  topicdict[topics_list[i]]=[]
  for j in range(len(keywords[i])):
    topicdict[topics_list[i]].append(keywords[i][j].lower())
print(topicdict)
```

[68] Python

```
{'corporatenews': ['red bull', 'air force', 'companies', 'twitter', 'elon musk', 'airbnb', 'amazon', 'ceo', 'company', 'disney world', 'walmart',
'hbo'], 'film': ['filming', 'science-fiction', 'drama', 'web-series', 'taylor swift', 'musician', 'country music', 'music', 'artists', 'jessie james
decker', 'louis tomlinson', 'one direction', 'harry style', 'filmmaking', 'film', 'movie', 'screenwriter', 'academy award-winning', 'tv special',
'nick jonas', 'jackie chan', 'tom hardy', 'avengers', 'marvel', 'nolan', 'superman', 'batman'], 'health': ['depression', 'disease', 'alzheimer',
'medication', 'medical', 'hospital', 'vaccine', 'covid', 'healthcare', 'drug', 'overdose', 'doctors', 'patients', 'consult', 'ebola', 'treatment',
'covid-19', 'sperm', 'doctor', 'recipe', 'fast food', 'genealogist', 'genetic', 'gene', 'blood', 'hospitalized', 'mental health', 'prescriptions',
'hospitals', 'respitatory', 'smoking weed', 'tobbaco smocker', 'ebola', 'virus', 'bacteria'], 'sports': ['football', 'world cup', 'tournament',
'raphael warnock', 'herschel walker', 'olympics', 'game', 'postgame', 'game-winner', 'player', 'bowlers', 'cricket', 'bowling', 'batsman', 'world cup
2022', 'fiat 500e', 'race', 'tom brady', 'nick saban', 'trophey', 'messi', 'ronaldo', 'soccer', 'premier league', 'fifa', 'world champion', 'nba',
'sports car', "shaquille o'neal", 'basketball'], 'politics': ['lapd', 'supreme court', 'democrat', 'biden', 'trump', 'federal', 'election',
'democratic', 'white house', 'congress', 'crown prince', 'bush', 'obama', 'fbi', 'governor', 'government', 'senate', 'candidate', 'politicians',
'freedom', 'president', 'politics', 'governance', 'presidency', 'republicans', 'democrats', 'legistative', 'prime minister', 'federal', 'world war',
'trial', 'lawsuit', 'attorney', 'missiles', 'sheriff', 'crime']}
```

```
df_cnn=pd.read_csv('CNN_1000_data_with_dates2.csv')
```

[69] Python

```
print(len(keywords),len(df_cnn_list))
```

[70] Python

```
for x in df_cnn_list:
  print(x.shape[0])
```

[72] Python

```
1950
943
1212
1493
6662
```

#Data Cleaning

```
for i in range(len(df_cnn_list)):
  df_cnn_list[i] = df_cnn_list[i].drop_duplicates('text')
```

[73] Python

```
for i  in range(len(df_cnn_list)):
  print("the number of tweets related to topic "+topics_list[i]+" is " + str(df_cnn_list[i].shape[0]))
```

[74] Python

```
the number of tweets related to topic corporatenews is 237
the number of tweets related to topic film is 113
the number of tweets related to topic health is 151
the number of tweets related to topic sports is 184
the number of tweets related to topic politics is 832
```

#Adding Topic Column to the Dataframes

```
for i in range(len(df_cnn_list)):
  df_cnn_list[i]['topic']=topics_list[i]
```

[75] Python

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

#Combining all the topics to a single dataframe

```
df_cnn_topic_combined= pd.DataFrame(columns=cols, dtype=object)
for x in df_cnn_list:
  df_cnn_topic_combined=df_cnn_topic_combined.append(x,ignore_index=True)

print("the total number of rows after collecting all the topics into a single dataframe is "+str(df_cnn_topic_combined.shape[0]))
```

[77] Python

```
the total number of rows after collecting all the topics into a single dataframe is 1517
```

```
df_cnn_topic_combined.to_csv('nov_23_df_cnn_topic_combined.csv')
```

[79] Python

Topic Labeling of Fox news into 5 categories

```
df_fox=pd.read_csv('FOX_NEWS_1000_data_with_dates2.csv')
```

[93] Python

```
df_fox_health= pd.DataFrame(columns=cols, dtype=object)
df_fox_politics= pd.DataFrame(columns=cols, dtype=object)
df_fox_sports= pd.DataFrame(columns=cols, dtype=object)
df_fox_film=pd.DataFrame(columns=cols,dtype=object)
df_fox_corporate=pd.DataFrame(columns=cols,dtype=object)
df_fox_list=[df_fox_corporate,df_fox_film,df_fox_health,df_fox_sports,df_fox_politics]
```

[94] Python

```
for _, row in df_fox.iterrows():
    text=row['text'].lower()
    keywords_count={'0':0,'1':0,'2':0,'3':0,'4':0}
    for i in range(len(topicdict)):
      keywords=topicdict[topics_list[i]]
      for x in keywords:
        if x in text:
          keywords_count[str(i)]+=1
    key=max(keywords_count, key=keywords_count.get)
    if keywords_count[key]>0:
      print(key,topics_list[int(key)],keywords_count)
      df_fox_list[int(key)]=df_fox_list[int(key)].append(row,ignore_index=True)
```

[95] Python

```
Output exceeds the size limit. Open the full output data in a text editor
```

```
for i  in range(len(df_fox_list)):
  print("the number of tweets from foxnews related to topic "+topics_list[i]+" is " + str(df_fox_list[i].shape[0]))
```

[87] Python

```
the number of tweets from foxnews related to topic corporatenews is 114
the number of tweets from foxnews related to topic film is 48
the number of tweets from foxnews related to topic health is 118
the number of tweets from foxnews related to topic sports is 253
the number of tweets from foxnews related to topic politics is 1051
```

#Data Cleaning

```
for i in range(len(df_fox_list)):
  df_fox_list[i] = df_fox_list[i].drop_duplicates('text')
```

[88] Python

```
print("<---------------------- After Cleaning--------------------------")
for i  in range(len(df_fox_list)):
  print("the number of tweets from foxnews related to topic "+topics_list[i]+" is " + str(df_fox_list[i].shape[0]))
```

[89] Python

```
<---------------------- After Cleaning--------------------------
the number of tweets from foxnews related to topic corporatenews is 114
the number of tweets from foxnews related to topic film is 48
the number of tweets from foxnews related to topic health is 118
the number of tweets from foxnews related to topic sports is 253
the number of tweets from foxnews related to topic politics is 1051
```

#Adding topic column to the dataframes

#Adding topic column to the dataframes

```
for i in range(len(df_fox_list)):
  df_fox_list[i]['topic']=topics_list[i]
```

[90] Python

Combining the Dataframes

```
df_fox_topic_combined= pd.DataFrame(columns=cols, dtype=object)
for x in df_fox_list:
  df_fox_topic_combined=df_fox_topic_combined.append(x,ignore_index=True)
print(df_fox_topic_combined.shape)
```

[91] Python

```
(1584, 6)
```

```
df_fox_topic_combined.to_csv('nov_23_df_fox_topic_combined.csv')
```

[92] Python

Topic Labeling of Reuters news into 5 categories

In [12]:
```
df_reuters=pd.read_csv('Reuters_1000_data_with_dates2.csv')
```

In [22]:
```
health = ['covid', 'covid variants', 'pfizer', 'pandemics', 'WHO', 'world health organization', 'infections', 'dise
politics = ['biden', 'US Government', 'Supreme Court', 'Trump', 'Republican', 'Democrat', 'administration', 'senato
sports = ['soccer', 'FIFAWorldCup', 'national team', 'World Cup', 'FIFA', 'Qatar World Cup', 'Football', 'Skaters',
filmandmusic = ['disney', 'movie', 'singer', 'GRAMMY', 'artist', 'dance', 'choreography', 'concerts', 'film', 'holl
corporatenews = ['google', 'apple', 'laid-off', 'tech', 'co-founder', 'companies', 'workforce', 'sales', 'lay-offs'
df_reuters_health= pd.DataFrame(columns=cols, dtype=object)
df_reuters_politics= pd.DataFrame(columns=cols, dtype=object)
df_reuters_sports= pd.DataFrame(columns=cols, dtype=object)
df_reuters_film=pd.DataFrame(columns=cols,dtype=object)
df_reuters_corporate=pd.DataFrame(columns=cols,dtype=object)
df_reuters_list=[df_reuters_corporate,df_reuters_film,df_reuters_health,df_reuters_sports,df_reuters_politics]
keywords=[corporatenews,filmandmusic,health,sports,politics]
topics_list=['corporatenews','film','health','sports','politics']
```

In [23]:
```
topicdict={}
for i in range(len(keywords)):
  topicdict[topics_list[i]]=[]
  for j in range(len(keywords[i])):
    topicdict[topics_list[i]].append(keywords[i][j].lower())
print(topicdict)
```

```
{'corporatenews': ['google', 'apple', 'laid-off', 'tech', 'co-founder', 'companies', 'workforce', 'sales', 'lay-of
ffs', 'tesla', 'wall st', 'e-commerce', 'metaverse', 'technologies', 'recession', 'twitter', 'amazon', 'elon musk']
, 'film': ['disney', 'movie', 'singer', 'grammy', 'artist', 'dance', 'choreography', 'concerts', 'film', 'hollywoo
d', 'premiere', 'actors'], 'health': ['covid', 'covid variants', 'pfizer', 'pandemics', 'who', 'world health organ
ization', 'infections', 'disease', 'healthcare', 'vaccinations'], 'sports': ['soccer', 'fifaworldcup', 'national t
eam', 'world cup', 'fifa', 'qatar world cup', 'football', 'skaters', 'champion'], 'politics': ['biden', 'us govern
ment', 'supreme court', 'trump', 'republican', 'democrat', 'administration', 'senators', 'democratic', 'vice presi
dent', 'foreign ministry']}
```

In [24]:
```
for _, row in df_fox.iterrows():
    text=row['text'].lower()
    keywords_count={'0':0,'1':0,'2':0,'3':0,'4':0}
    for i in range(len(topicdict)):
      keywords=topicdict[topics_list[i]]
      for x in keywords:
        if x in text:
          keywords_count[str(i)]+=1
    key=max(keywords_count, key=keywords_count.get)
    if keywords_count[key]>0:
      print(key,topics_list[int(key)],keywords_count)
      df_reuters_list[int(key)]=df_reuters_list[int(key)].append(row,ignore_index=True)
```

```
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 2}
1 film {'0': 0, '1': 1, '2': 0, '3': 0, '4': 0}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
0 corporatenews {'0': 2, '1': 0, '2': 0, '3': 0, '4': 0}
2 health {'0': 0, '1': 0, '2': 1, '3': 0, '4': 0}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 2}
0 corporatenews {'0': 1, '1': 0, '2': 1, '3': 0, '4': 0}
3 sports {'0': 0, '1': 0, '2': 0, '3': 1, '4': 0}
1 film {'0': 0, '1': 1, '2': 0, '3': 0, '4': 0}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
1 film {'0': 0, '1': 1, '2': 0, '3': 0, '4': 0}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
0 corporatenews {'0': 1, '1': 0, '2': 0, '3': 0, '4': 0}
3 sports {'0': 0, '1': 0, '2': 0, '3': 2, '4': 0}
4 politics {'0': 0, '1': 0, '2': 0, '3': 0, '4': 1}
0 corporatenews {'0': 1, '1': 0, '2': 0, '3': 0, '4': 0}
0 corporatenews {'0': 1, '1': 0, '2': 0, '3': 0, '4': 0}
```

#Data Cleaning

In [27]:
```
print("<---------------------- After Cleaning--------------------------")
for i  in range(len(df_reuters_list)):
  print("the number of tweets from reuters related to topic "+topics_list[i]+" is " + str(df_reuters_list[i].s
```

```
<---------------------- After Cleaning--------------------------
the number of tweets from reuters related to topic corporatenews is 1265
the number of tweets from reuters related to topic film is 616
the number of tweets from reuters related to topic health is 1441
the number of tweets from reuters related to topic sports is 682
the number of tweets from reuters related to topic politics is 7438
```

#Adding topic column to the dataframes

In [28]:
```
for i in range(len(df_reuters_list)):
  df_reuters_list[i]['topic']=topics_list[i]
```

Combining the Dataframes

#fox news

In [91]:
```
df_fox_topic_combined= pd.DataFrame(columns=cols, dtype=object)
for x in df_fox_list:
  df_fox_topic_combined=df_fox_topic_combined.append(x,ignore_index=True)
print(df_fox_topic_combined.shape)
```

```
(1584, 6)
```

In [92]:
```
df_fox_topic_combined.to_csv('nov_23_df_fox_topic_combined.csv')
```

#reuters news

In [29]:
```
df_reuters_topic_combined= pd.DataFrame(columns=cols, dtype=object)
for x in df_reuters_list:
  df_reuter_topic_combined=df_reuters_topic_combined.append(x,ignore_index=True)
print(df_reuters_topic_combined.shape)
```

```
(0, 4)
```

```
/var/folders/y0/v83q7r_j77q28149_k_qs_sr0000gn/T/ipykernel_37384/1545716840.py:3: FutureWarning: The frame.ap
pend method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  df_reuter_topic_combined=df_reuters_topic_combined.append(x,ignore_index=True)
```

In [31]:
```
df_reuters_topic_combined.to_csv('nov_23_df_reuters_topic_combined.csv')
```

2.4 CSV File After Topic Labeling

2.4.1 CNN:

	screenNan	descriptior	location	text	Unnamed:	topic
0	CNN	Itâ€™s our job to #Gc		Another er	31	corporatenews
1	CNN	Itâ€™s our job to #Gc		Amazon Cl	36	corporatenews
2	CNN	Itâ€™s our job to #Gc		Max Verst	37	corporatenews
3	CNN	Itâ€™s our job to #Gc		Here are	45	corporatenews
4	CNN	Itâ€™s our job to #Gc		#RIPTwitte	53	corporatenews
5	CNN	Itâ€™s our job to #Gc		FTXâ€™s n	54	corporatenews
6	CNN	Itâ€™s our job to #Gc		Death is in	57	corporatenews
7	CNN	Itâ€™s our job to #Gc		The hot ne	62	corporatenews
8	CNN	Itâ€™s our job to #Gc		Airbnb CEC	65	corporatenews
9	CNN	Itâ€™s our job to #Gc		Disney Wc	67	corporatenews
10	CNN	Itâ€™s our job to #Gc		Another er	72	corporatenews
11	CNN	Itâ€™s our job to #Gc		Amazon's	91	corporatenews
12	CNN	Itâ€™s our job to #Gc		Another er	95	corporatenews
13	CNN	Itâ€™s our job to #Gc		The Tesla l	96	corporatenews
14	CNN	Itâ€™s our job to #Gc		NBA super	98	corporatenews
15	CNN	Itâ€™s our job to #Gc		"He is a	106	corporatenews
16	CNN	Itâ€™s our job to #Gc		Sen. Amy K	133	corporatenews
17	CNN	Itâ€™s our job to #Gc		Elon Musk	137	corporatenews
18	CNN	Itâ€™s our job to #Gc		The hot ne	139	corporatenews
19	CNN	Itâ€™s our job to #Gc		Airbnb CEC	145	corporatenews
20	CNN	Itâ€™s our job to #Gc		Brian Ches	146	corporatenews
21	CNN	Itâ€™s our job to #Gc		Senator Ar	147	corporatenews
22	CNN	Itâ€™s our job to #Gc		Disney Wc	148	corporatenews
23	CNN	Itâ€™s our job to #Gc		Thanksgivi	158	corporatenews
24	CNN	Itâ€™s our job to #Gc		Elon Musk	168	corporatenews
25	CNN	Itâ€™s our job to #Gc		Taylor Swi	178	corporatenews

2.4.2 Fox News:

	screenNan	descriptior	location	text	Unnamed:	topic
0	FoxNews	Follow Am	U.S.A.	Elon Musk	10	corporatenews
1	FoxNews	Follow Am	U.S.A.	Musk trolls	20	corporatenews
2	FoxNews	Follow Am	U.S.A.	Twitter rep	52	corporatenews
3	FoxNews	Follow Am	U.S.A.	Sen. Marke	63	corporatenews
4	FoxNews	Follow Am	U.S.A.	Surge in Tv	81	corporatenews
5	FoxNews	Follow Am	U.S.A.	Former Vo	115	corporatenews
6	FoxNews	Follow Am	U.S.A.	Olympics 2	136	corporatenews
7	FoxNews	Follow Am	U.S.A.	Air Force, I	157	corporatenews
8	FoxNews	Follow Am	U.S.A.	Sen. Marke	180	corporatenews
9	FoxNews	Follow Am	U.S.A.	Air Force t	289	corporatenews
10	FoxNews	Follow Am	U.S.A.	Chinese na	312	corporatenews
11	FoxNews	Follow Am	U.S.A.	AQUAMAN	317	corporatenews
12	FoxNews	Follow Am	U.S.A.	Clinton-lin	327	corporatenews
13	FoxNews	Follow Am	U.S.A.	FOX	334	corporatenews
14	FoxNews	Follow Am	U.S.A.	Justin Verla	447	corporatenews
15	FoxNews	Follow Am	U.S.A.	Dem senat	463	corporatenews
16	FoxNews	Follow Am	U.S.A.	Elon	467	corporatenews
17	FoxNews	Follow Am	U.S.A.	Ellen Greer	488	corporatenews
18	FoxNews	Follow Am	U.S.A.	BREAKIN	547	corporatenews
19	FoxNews	Follow Am	U.S.A.	Max Verst	599	corporatenews
20	FoxNews	Follow Am	U.S.A.	Twitter ske	606	corporatenews
21	FoxNews	Follow Am	U.S.A.	Virginia ma	607	corporatenews
22	FoxNews	Follow Am	U.S.A.	OPINION:	626	corporatenews
23	FoxNews	Follow Am	U.S.A.	Mexico's a	631	corporatenews
24	FoxNews	Follow Am	U.S.A.	'Huge nerv	691	corporatenews
25	FoxNews	Follow Am	U.S.A.	Liberal tec	710	corporatenews

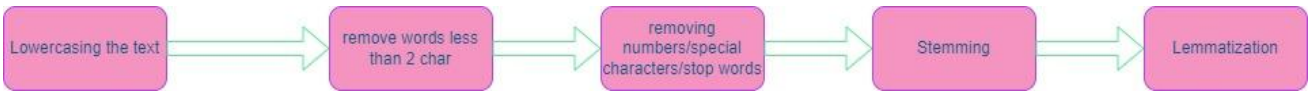
2.4.3 Reuters

	screenNan	descriptior	location	text	Unnamed:	topic
0	Reuters	Top and br	Around the	Italy court	0	corporatenews
1	Reuters	Top and br	Around the	Jaguar Lan	4	corporatenews
2	Reuters	Top and br	Around the	Indian foo	12	corporatenews
3	Reuters	Top and br	Around the	Companies	16	corporatenews
4	Reuters	Top and br	Around the	Twitter ha	17	corporatenews
5	Reuters	Top and br	Around the	Archer Avi	51	corporatenews
6	Reuters	Top and br	Around the	Tesla recal	58	corporatenews
7	Reuters	Top and br	Around the	Elon Musk	61	corporatenews
8	Reuters	Top and br	Around the	Italy court	68	corporatenews
9	Reuters	Top and br	Around the	RT @Reute	71	corporatenews
10	Reuters	Top and br	Around the	Jaguar Lan	74	corporatenews
11	Reuters	Top and br	Around the	Biden to m	83	corporatenews
12	Reuters	Top and br	Around the	Indian foo	92	corporatenews
13	Reuters	Top and br	Around the	Australian	103	corporatenews
14	Reuters	Top and br	Around the	Tesla recal	130	corporatenews
15	Reuters	Top and br	Around the	Ethiopia ha	137	corporatenews
16	Reuters	Top and br	Around the	Factbox: T	142	corporatenews
17	Reuters	Top and br	Around the	Factbox: A	156	corporatenews
18	Reuters	Top and br	Around the	FTXâ€™s n	163	corporatenews
19	Reuters	Top and br	Around the	From a dee	165	corporatenews
20	Reuters	Top and br	Around the	Australian	184	corporatenews
21	Reuters	Top and br	Around the	Factbox: T	198	corporatenews
22	Reuters	Top and br	Around the	Sources sa	220	corporatenews
23	Reuters	Top and br	Around the	ðŸ¢ Resea	236	corporatenews
24	Reuters	Top and br	Around the	With tech	278	corporatenews
25	Reuters	Top and br	Around the	Hundreds	292	corporatenews

Chapter 3. Preprocessing

Tweet text is preprocessed by following a sequential process that involves, lower casing, removing word less than 2 char, removing numbers, removing special characters, removing stopwords, stemming, lemmatizing.

3.1 Pipeline Structure for Preprocessing the Data:

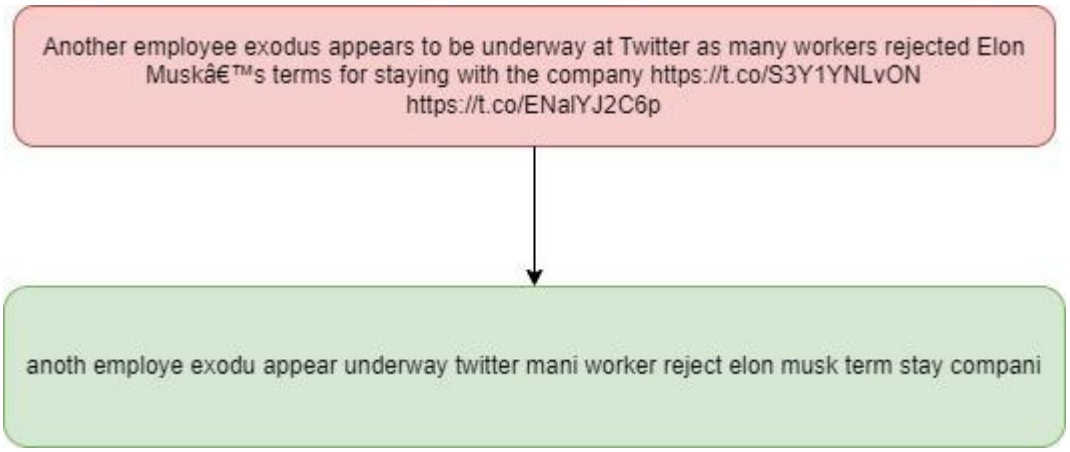


3.2 Implementation of text preprocess:

```python
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def text_preprocess(value):
    value['text'] = value['text'].str.lower() #lowercase all words
    value['text']=value['text'].str.split().map(lambda sl: " ".join(s for s in sl if len(s) > 2)) #remove word less than 2 char
    value['text'] = value['text'].str.replace(r'\d+','') #removing numbers
    value['text'] = value['text'].str.replace(r'[ ](?=[ ])|[^-_,A-Za-z0-9 ]+', "") #removing special characters
    value['tokenized_sentences'] = value.apply(lambda row: nltk.word_tokenize(row['text']), axis=1)
    value['tokenized_sentences']= value['tokenized_sentences'].apply(lambda x: [item for item in x if item not in stop_list]) #removing stopwords
    value['tokenized_sentences'] = value['tokenized_sentences'].apply(lambda x : [stemmer.stem(y) for y in x]) #stemming
    value['tokenized_sentences'] = value['tokenized_sentences'].apply(lambda x : [lemmatizer.lemmatize(y) for y in x]) #lemmatizing
    value['clean_text'] = value['tokenized_sentences'].str.join(' ') #converting back to normal string from tokenized string
    return value
```

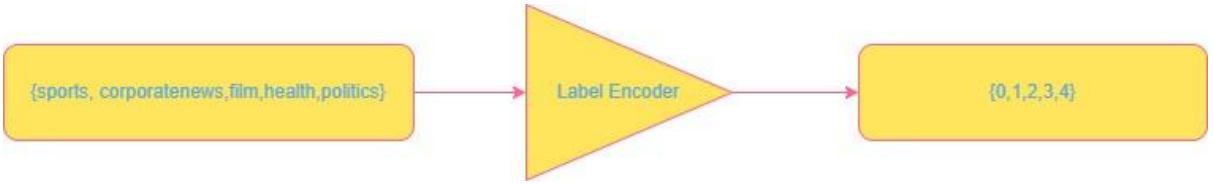
[346] Python

3.3 Results:



Chapter 4: Encoding

Encoding is done on the topic labels to convert that into categorical numbers. The labelencoder is utilized to encode in this project. After the encoding the ‘Y’ values are changed to numerical representation.



4.1 Implementation of Label Encoder:

```python
from sklearn.preprocessing import LabelEncoder

def encode_topic(dataframe):
    encoder = LabelEncoder()
    dataframe['topicEncoded'] = encoder.fit_transform(dataframe['topic'])
    dataframe['text'] = dataframe['text'].apply(lambda text: str(text).lower())
    return dataframe
```

Python

Chapter 5 : Random Forest Classifier

5.1 Vectorising:

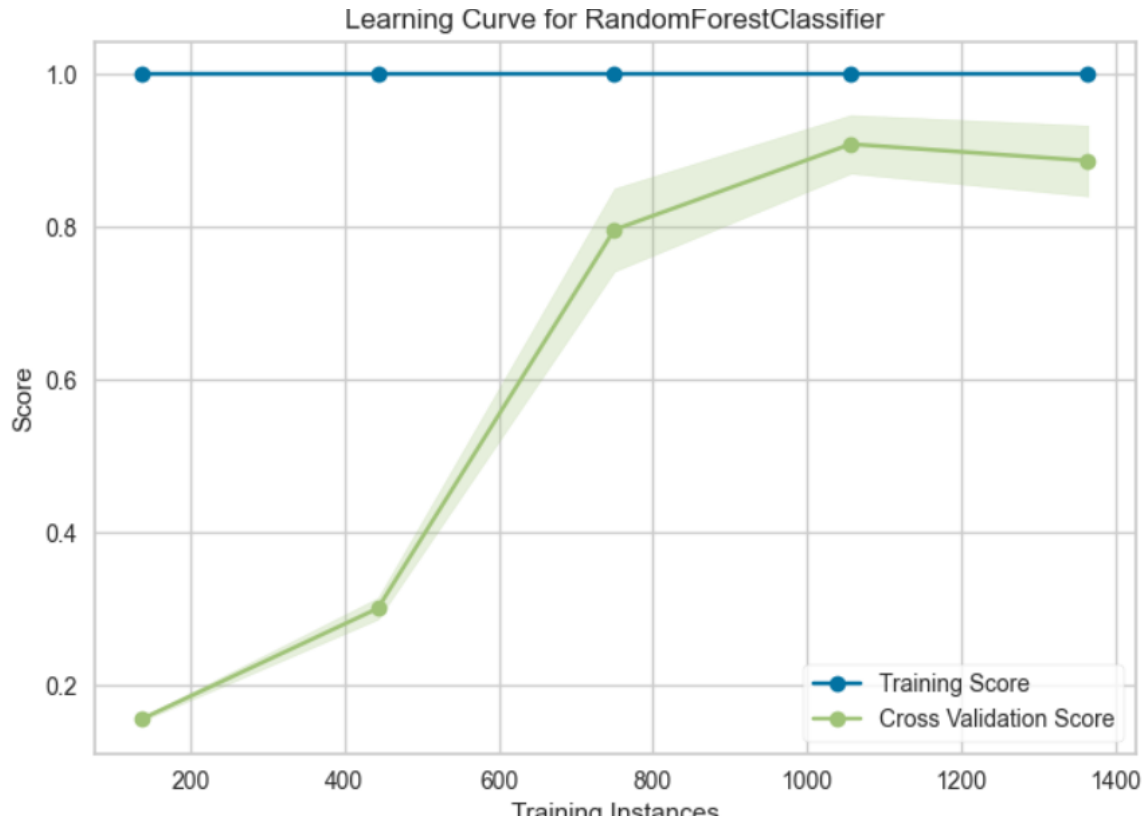
tfidfvectorizer with max_features as 2500 is used for vectoring the preprocessed tweet text to numerical representation(vector).

5.1.1 Implementation of Vectoriser:

```
#Vectoriser

    v = TfidfVectorizer(max_features=2500) #Initializnig the TF-IDF Vectorizer
]

#Defining the independent and dependent vectors for all three datasets

    X_left = v.fit_transform(df_cnn['clean_text']).toarray()
    X_right= v.fit_transform(df_fox['clean_text']).toarray()
    X_neutral=v.fit_transform(df_reuters['clean_text']).toarray()
    y_left=df_cnn['topicEncoded']
    y_right=df_fox['topicEncoded']
    y_nuetral=df_reuters['topicEncoded']
]

    print(X_left.shape)
    print(X_right.shape)
    print(X_neutral.shape)
]
(1517, 2500)
(1584, 2500)
(1332, 2500)
```

5.2 Training it on the left and Test it on the left:

30% of the collected data was splitted into test dataset, other 70% were utilized for training the dataset. learning curve for training on the left and testing on the same dataset is shown below:



Classification Report:

	precision	recall	f1-score	support
0	0.97	0.91	0.94	81
1	1.00	0.71	0.83	38
2	0.93	0.85	0.89	46
3	0.86	0.98	0.92	237
4	0.98	0.74	0.84	54
accuracy			0.91	456
macro avg	0.95	0.84	0.88	456
weighted avg	0.91	0.91	0.90	456

0.9057017543859649

Accuracy while training and testing on the left dataset using random forest model is 91%

5.3. Training on the left data set and test it on the right dataset:

The classification report tested on the right while train on the left is shown below. We can see a clear dip in the accuracy values from **91%** to **61%** . Ideological bias is affecting the random forest model is what we could infer from this accuracy dip.

	precision	recall	f1-score	support
0	0.10	0.02	0.03	114
1	0.05	0.06	0.06	48
2	0.10	0.03	0.05	118
3	0.66	0.90	0.76	1051
4	0.18	0.02	0.03	253
accuracy			0.61	1584
macro avg	0.22	0.21	0.19	1584
weighted avg	0.48	0.61	0.52	1584

0.6066919191919192

5.4 Train on left and test on neutral:

The classification report while trained on the left and tested on neutral. We see a significant decrease in the accuracy of the model from **91%** to **34%**

	precision	recall	f1-score	support
0	0.16	0.04	0.06	299
1	0.50	0.21	0.30	107
2	0.46	0.08	0.14	213
3	0.34	0.87	0.49	441
4	0.42	0.07	0.12	272
accuracy			0.34	1332
macro avg	0.38	0.25	0.22	1332
weighted avg	0.35	0.34	0.25	1332

0.3400900900900901

5.5 Train on Right and Test on Right:

Classification report for training and testing on the same dataset. The accuracy is **88%.**

```
[122]  print_evaluators(y_right_test,y_pred_right)
```

	precision	recall	f1-score	support
0	0.96	0.77	0.86	31
1	1.00	0.57	0.73	14
2	1.00	0.60	0.75	45
3	0.85	0.99	0.92	314
4	0.94	0.67	0.78	72
accuracy			0.88	476
macro avg	0.95	0.72	0.81	476
weighted avg	0.89	0.88	0.87	476

0.8781512605042017

5.6 Train on Right and Test on Left

	precision	recall	f1-score	support
0	0.20	0.03	0.04	237
1	0.00	0.00	0.00	113
2	0.17	0.06	0.09	151
3	0.55	0.90	0.68	832
4	0.12	0.04	0.06	184
accuracy			0.51	1517
macro avg	0.21	0.20	0.18	1517
weighted avg	0.36	0.51	0.40	1517

0.5062623599208965

5.7 Train on Right and Test on Neutral:

	precision	recall	f1-score	support
0	0.08	0.01	0.02	299
1	0.00	0.00	0.00	107
2	0.09	0.03	0.05	213
3	0.33	0.84	0.47	441
4	0.02	0.01	0.01	272
accuracy			0.29	1332
macro avg	0.11	0.18	0.11	1332
weighted avg	0.15	0.29	0.17	1332

0.2875375375375375

5.8 Train on Neutral and Test on Neutral

	precision	recall	f1-score	support
0	0.94	0.95	0.95	84
1	0.97	0.92	0.94	37
2	0.98	0.84	0.91	63
3	0.89	0.98	0.93	136
4	0.97	0.93	0.95	80
accuracy			0.94	400
macro avg	0.95	0.92	0.94	400
weighted avg	0.94	0.94	0.93	400

0.935

5.9 Train on Neutral and Test on left

	precision	recall	f1-score	support
0	0.16	0.03	0.06	237
1	0.52	0.20	0.29	113
2	0.14	0.13	0.14	151
3	0.56	0.81	0.66	832
4	0.08	0.04	0.05	184
accuracy			0.48	1517
macro avg	0.29	0.24	0.24	1517
weighted avg	0.40	0.48	0.41	1517

0.480553724456163 46

5.10 Train on Neutral and Test on Right

	precision	recall	f1-score	support
0	0.08	0.11	0.09	114
1	0.00	0.00	0.00	48
2	0.08	0.08	0.08	118
3	0.71	0.81	0.75	1051
4	0.29	0.07	0.11	253
accuracy			0.56	1584
macro avg	0.23	0.21	0.21	1584
weighted avg	0.53	0.56	0.53	1584

0.5625

Chapter 6 : Logistic Regression

Vectorisation:

tfidfvectorizer with max_features as 2500 as well as CounterVectoriser is used for vectoring the preprocessed tweet text to numerical representation(vector).

6.1 Train on left and test on left with tfidfvectorizer

	precision	recall	f1-score	support
0	1.00	0.81	0.89	72
1	0.95	0.77	0.85	26
2	0.96	0.60	0.74	40
3	0.81	1.00	0.90	255
4	0.95	0.57	0.71	63
accuracy			0.86	456
macro avg	0.93	0.75	0.82	456
weighted avg	0.88	0.86	0.85	456

0.8618421052631579

6.2 Train on left and test on left with Count Vectorizer

	precision	recall	f1-score	support
0	1.00	0.93	0.96	72
1	1.00	0.77	0.87	26
2	1.00	0.80	0.89	40
3	0.90	0.99	0.95	255
4	0.89	0.81	0.85	63
accuracy			0.93	456
macro avg	0.96	0.86	0.90	456
weighted avg	0.93	0.93	0.93	456

0.9276315789473685

6.3 Train on left and test on Right wit h tfidfvectorizer

	precision	recall	f1-score	support
0	1.00	0.72	0.84	114
1	1.00	0.52	0.68	48
2	0.94	0.42	0.58	118
3	0.83	1.00	0.91	1051
4	0.96	0.61	0.75	253
accuracy			0.86	1584
macro avg	0.95	0.65	0.75	1584
weighted avg	0.88	0.86	0.84	1584

0.8579545454545454

6.4 Train on left and test on Right with Count Vectorizer

	precision	recall	f1-score	support
0	1.00	0.86	0.92	114
1	0.94	0.71	0.81	48
2	0.96	0.67	0.79	118
3	0.90	0.99	0.94	1051
4	0.93	0.78	0.85	253
accuracy			0.91	1584
macro avg	0.95	0.80	0.86	1584
weighted avg	0.92	0.91	0.91	1584

0.9141414141414141

6.5 Train on left and test on Neutral with tfidfvectorizer

	precision	recall	f1-score	support
0	0.98	0.72	0.83	299
1	0.89	0.46	0.60	107
2	0.97	0.41	0.58	213
3	0.56	0.99	0.72	441
4	0.93	0.64	0.76	272
accuracy			0.72	1332
macro avg	0.86	0.64	0.70	1332
weighted avg	0.82	0.72	0.72	1332

0.7237237237237237

6.6 Train on left and test on Neutral with Count Vectorizer

	precision	recall	f1-score	support
0	0.94	0.77	0.85	299
1	0.79	0.62	0.69	107
2	0.93	0.54	0.68	213
3	0.66	0.98	0.79	441
4	0.91	0.74	0.82	272
accuracy			0.79	1332
macro avg	0.85	0.73	0.77	1332
eighted avg	0.83	0.79	0.78	1332

.786036036036036

6.7 Train on Right and test on Right with tfidfvectorizer

	precision	recall	f1-score	support
0	1.00	0.59	0.74	39
1	1.00	0.27	0.43	11
2	0.83	0.34	0.49	29
3	0.85	0.99	0.92	330
4	0.96	0.78	0.86	67
accuracy			0.87	476
macro avg	0.93	0.60	0.69	476
weighted avg	0.88	0.87	0.86	476

0.8739495798319328

6.8 Train on Right and test on Right with Count Vectorizer

	precision	recall	f1-score	support
0	1.00	0.69	0.82	39
1	1.00	0.55	0.71	11
2	0.89	0.59	0.71	29
3	0.91	0.99	0.95	330
4	0.92	0.85	0.88	67
accuracy			0.91	476
macro avg	0.94	0.73	0.81	476
weighted avg	0.92	0.91	0.91	476

0.9138655462184874

6.9 Train on Right and test on left with tfidfvectorizer

	precision	recall	f1-score	support
0	0.98	0.45	0.62	237
1	0.95	0.17	0.29	113
2	0.93	0.47	0.63	151
3	0.70	1.00	0.82	832
4	0.87	0.58	0.69	184
accuracy			0.75	1517
macro avg	0.89	0.53	0.61	1517
weighted avg	0.80	0.75	0.71	1517

0.7455504284772577

6.10 Train on Right and test on left with Count Vectorizer

	precision	recall	f1-score	support
0	0.96	0.67	0.79	237
1	0.88	0.40	0.55	113
2	0.90	0.74	0.81	151
3	0.82	0.97	0.89	832
4	0.78	0.82	0.80	184
accuracy			0.84	1517
macro avg	0.87	0.72	0.77	1517
weighted avg	0.85	0.84	0.83	1517

0.8378378378378378

6.11 Train on Right and test on Neutral with tfidfvectorizer

	precision	recall	f1-score	support
0	0.98	0.41	0.58	299
1	0.96	0.23	0.38	107
2	0.99	0.42	0.59	213
3	0.50	0.99	0.66	441
4	0.88	0.67	0.76	272
accuracy			0.64	1332
macro avg	0.86	0.55	0.59	1332
weighted avg	0.80	0.64	0.63	1332

0.6433933933933934

6.12 Train on Right and test on Neutral with Count Vectorizer

	precision	recall	f1-score	support
0	0.94	0.56	0.70	299
1	0.94	0.42	0.58	107
2	0.97	0.58	0.72	213
3	0.59	0.98	0.74	441
4	0.85	0.79	0.82	272
accuracy			0.74	1332
macro avg	0.86	0.66	0.71	1332
weighted avg	0.81	0.74	0.73	1332

0.7364864864864865

6.13 Train on Neutral and test on Neutral with tfidfvectorizer

	precision	recall	f1-score	support
0	0.87	0.96	0.91	71
1	1.00	0.67	0.80	30
2	0.98	0.64	0.78	70
3	0.82	0.99	0.89	142
4	0.94	0.92	0.93	87
accuracy			0.88	400
macro avg	0.92	0.83	0.86	400
weighted avg	0.90	0.88	0.88	400

0.8825

6.14 Train on Neutral and test on Neutral with Count Vectorizer

	precision	recall	f1-score	support
0	0.94	0.94	0.94	71
1	0.93	0.83	0.88	30
2	1.00	0.83	0.91	70
3	0.89	0.99	0.93	142
4	0.98	0.97	0.97	87
accuracy			0.94	400
macro avg	0.95	0.91	0.93	400
weighted avg	0.94	0.94	0.93	400

0.935

6.15 Train on Neutral and test on Right with tfidfvectorizer

	precision	recall	f1-score	support
0	0.75	0.74	0.74	114
1	1.00	0.44	0.61	48
2	0.71	0.45	0.55	118
3	0.83	0.98	0.90	1051
4	0.86	0.47	0.61	253
accuracy			0.82	1584
macro avg	0.83	0.61	0.68	1584
weighted avg	0.83	0.82	0.81	1584

0.8238636363636364

6.16 Train on Neutral and test on Right with Count Vectorizer

	precision	recall	f1-score	support
0	0.85	0.72	0.78	114
1	0.96	0.50	0.66	48
2	0.68	0.62	0.65	118
3	0.86	0.98	0.92	1051
4	0.92	0.57	0.70	253
accuracy			0.85	1584
macro avg	0.85	0.68	0.74	1584
weighted avg	0.86	0.85	0.84	1584

0.8535353535353535

6.17 Train on Neutral and test on left with tfidfvectorizer

	precision	recall	f1-score	support
0	0.79	0.84	0.81	237
1	0.98	0.49	0.65	113
2	0.74	0.54	0.62	151
3	0.81	0.98	0.89	832
4	0.81	0.41	0.54	184
accuracy			0.81	1517
macro avg	0.83	0.65	0.70	1517
weighted avg	0.81	0.81	0.79	1517

0.8075148319050758

6.18 Train on Neutral and test on left with Count Vectorizer

	precision	recall	f1-score	support
0	0.84	0.84	0.84	237
1	0.93	0.58	0.72	113
2	0.63	0.66	0.64	151
3	0.84	0.96	0.90	832
4	0.84	0.48	0.61	184
accuracy			0.83	1517
macro avg	0.82	0.71	0.74	1517
weighted avg	0.83	0.83	0.82	1517

0.8259723137771918

Chapter 7 : Convolutional Neural Network

For embeddings we are using glove based embeddings. Refer to the code to see more how its utilized.

7.1 Train on left and Test on left:

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print('Test Accuracy for training and testing on the cnn dataset: %f' % (test_acc*100))
```

```
8/8 - 1s - loss: 0.3551 - accuracy: 0.8860 - 1s/epoch - 126ms/step
Test Accuracy for training and testing on the cnn dataset: 88.596493
```

7.2 Train on left and test on right:

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
```

```
50/50 - 8s - loss: 1.4853 - accuracy: 0.5808 - 8s/epoch - 150ms/step
```

7.3 Train on right and test on right:

```
test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)
print('Test Accuracy for training and testing on the cnn dataset: %f' % (test_acc*100))
```

```
8/8 - 1s - loss: 0.8536 - accuracy: 0.7605 - 1s/epoch - 184ms/step
Test Accuracy for training and testing on the cnn dataset: 76.050419
```

7.4 Train on right and test on left:

```
test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)
print('Test Accuracy for training on the fox dataset and testing on the cnn dataset: %f' % (test_acc*100))
```

```
48/48 - 8s - loss: 1.9979 - accuracy: 0.5346 - 8s/epoch - 167ms/step
Test Accuracy for training on the fox dataset and testing on the cnn dataset: 53.460777
```

Chapter 8 : Support Vector Machine

tfidfvectorizer with max_features as 5000 is used for vectoring the preprocessed tweet text to numerical representation(vector).

8.1 Train on left and test on right:

Training on CNN and testing on FOX and Reuters

In [43]:
```
Train_X_CNN, Train_Y_CNN = preprocessedData_CNN['clean_text_final'], preprocessedData_CNN['topicEncoded']
Test_X_FOX, Test_Y_FOX = preprocessedData_Fox['clean_text_final'], preprocessedData_Fox['topicEncoded']
Test_X_Reuters, Test_Y_Reuters = preprocessedData_Reuters['clean_text_final'], preprocessedData_Reuters['topicEncode
```

Testing with FOX News

In [52]:
```
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(preprocessedData_CNN['clean_text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X_CNN)
Test_X_Tfidf = Tfidf_vect.transform(Test_X_FOX)

# print(Tfidf_vect.vocabulary_)
```

```
SVM Accuracy Score ->  87.75252525252525
```

In [54]:
```
print(classification_report(Test_Y_FOX,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       0.91      0.77      0.83       114
           1       0.97      0.58      0.73        48
           2       0.99      0.63      0.77       118
           3       0.86      0.99      0.92      1051
           4       0.94      0.63      0.76       253

    accuracy                           0.88      1584
   macro avg       0.93      0.72      0.80      1584
weighted avg       0.89      0.88      0.87      1584
```

8.2 Train on left and test on neutral:

```
SVM Accuracy Score ->  75.22522522522522
```

In [50]:
```
print(classification_report(Test_Y_Reuters,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       0.91      0.77      0.84       299
           1       0.85      0.63      0.72       107
           2       0.96      0.42      0.59       213
           3       0.61      0.97      0.75       441
           4       0.94      0.69      0.79       272

    accuracy                           0.75      1332
   macro avg       0.85      0.70      0.74      1332
weighted avg       0.82      0.75      0.75      1332
```

8.3 Train on right and test on left:

Training on FOX and testing on CNN and Reuters

In [51]:
```
Train_X_FOX, Train_Y_FOX = preprocessedData_Fox['clean_text_final'], preprocessedData_Fox['topicEncoded']
Test_X_CNN, Test_Y_CNN = preprocessedData_CNN['clean_text_final'], preprocessedData_CNN['topicEncoded']
Test_X_Reuters, Test_Y_Reuters = preprocessedData_Reuters['clean_text_final'], preprocessedData_Reuters['topicEncode
```

Testing with CNN News

In [55]:
```
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(preprocessedData_Fox['clean_text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X_FOX)
Test_X_Tfidf = Tfidf_vect.transform(Test_X_CNN)

# print(Tfidf_vect.vocabulary_)
```

```
SVM Accuracy Score ->  78.04878048780488
```

In [58]:
```
print(classification_report(Test_Y_CNN,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       0.97      0.49      0.66       237
           1       0.93      0.35      0.50       113
           2       0.96      0.56      0.71       151
           3       0.73      0.98      0.84       832
           4       0.83      0.70      0.76       184

    accuracy                           0.78      1517
   macro avg       0.88      0.62      0.69      1517
weighted avg       0.82      0.78      0.76      1517
```

8.4 Train on right and test on neutral:

```
SVM Accuracy Score ->  70.94594594594594
```

In [62]:
```
print(classification_report(Test_Y_Reuters,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       0.97      0.46      0.62       299
           1       0.94      0.42      0.58       107
           2       0.99      0.52      0.68       213
           3       0.55      0.98      0.71       441
           4       0.89      0.80      0.84       272

    accuracy                           0.71      1332
   macro avg       0.87      0.64      0.69      1332
weighted avg       0.82      0.71      0.70      1332
```

8.5 Train on neutral and test on left:

Training on Reuters and testing on CNN and FOX

In [63]:
```
Train_X_Reuters, Train_Y_Reuters = preprocessedData_Reuters['clean_text_final'], preprocessedData_Reuters['top
Test_X_CNN, Test_Y_CNN = preprocessedData_CNN['clean_text_final'], preprocessedData_CNN['topicEncoded']
Test_X_FOX, Test_Y_FOX = preprocessedData_Fox['clean_text_final'], preprocessedData_Fox['topicEncoded']
```

Testing with CNN News

In [64]:
```
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(preprocessedData_Reuters['clean_text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X_Reuters)
Test_X_Tfidf = Tfidf_vect.transform(Test_X_CNN)

# print(Tfidf_vect.vocabulary_)
```

In [65]:
```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X_Tfidf,Train_Y_Reuters)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y_CNN)*100)
```

```
SVM Accuracy Score ->  82.00395517468688
```

In [66]:
```
print(classification_report(Test_Y_CNN,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       0.80      0.81      0.81       237
           1       0.94      0.54      0.69       113
           2       0.68      0.60      0.64       151
           3       0.83      0.98      0.90       832
           4       0.86      0.45      0.59       184

    accuracy                           0.82      1517
   macro avg       0.82      0.68      0.72      1517
weighted avg       0.82      0.82      0.81      1517
```

8.6 Train on neutral and test on right:

```
SVM Accuracy Score ->  83.77525252525253
```

In [69]:
```
print(classification_report(Test_Y_FOX,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       0.81      0.73      0.76       114
           1       1.00      0.44      0.61        48
           2       0.67      0.60      0.63       118
           3       0.85      0.98      0.91      1051
           4       0.89      0.49      0.63       253

    accuracy                           0.84      1584
   macro avg       0.84      0.65      0.71      1584
weighted avg       0.84      0.84      0.82      1584
```

8.7 Train on neutral and test on neutral:

Training and Testing with Reuters News

In [81]:
```
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(preprocessedData_Reuters['clean_text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X_Reuters)
Test_X_Tfidf = Tfidf_vect.transform(Train_X_Reuters)

# print(Tfidf_vect.vocabulary_)
```

In [82]:
```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X_Tfidf,Train_Y_Reuters)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Train_Y_Reuters)*100)
```

```
SVM Accuracy Score ->  99.47447447447448
```

In [83]:
```
print(classification_report(Train_Y_Reuters,predictions_SVM))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       299
           1       0.98      0.99      0.99       107
           2       1.00      0.99      0.99       213
           3       1.00      1.00      1.00       441
           4       0.99      1.00      0.99       272

    accuracy                           0.99      1332
   macro avg       0.99      0.99      0.99      1332
weighted avg       0.99      0.99      0.99      1332
```

Chapter 9 : Recurrent Neural Network

For Embeddings, we used inbuilt-libraries. Please refer to the code for further reference, here is a small snippet of code.

Additional DataPreProcessing for RNN

In [92]:
```
def padding(data_frame):
    MAX_WORDS = 1000
    MAX_SEQUENCE_LENGTH=1000
    encoded_docs=[one_hot(item['clean_text'],vocab_size) for i,item in data_frame.iterrows()]
    text=data_frame['clean_text'].to_list()
    labels=data_frame['topicEncoded'].to_list()

    tokenizer  = Tokenizer(num_words = MAX_WORDS)
    tokenizer.fit_on_texts(text)
    sequences =  tokenizer.texts_to_sequences(text)

    word_index = tokenizer.word_index
#     print("unique words : {}".format(len(word_index)))

    data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)

    labels = to_categorical(np.asarray(labels))
    return data, labels
#     print('Shape of data tensor:', data.shape)
#     print('Shape of label tensor:', labels.shape)
#     print(labels)
```

9.1 Train on left and test on right:

Training on CNN

In [93]:
```
Train_X_CNN, Train_Y_CNN = preprocessedData_CNN['clean_text'], preprocessedData_CNN['topicEncoded']
Test_X_FOX, Test_Y_FOX = preprocessedData_Fox['clean_text'], preprocessedData_Fox['topicEncoded']
Test_X_Reuters, Test_Y_Reuters = preprocessedData_Reuters['clean_text'], preprocessedData_Reuters['topicEncoded']
x_train_cnn, y_train_cnn = padding(preprocessedData_CNN)
x_test_fox, y_test_fox = padding(preprocessedData_Fox)
x_test_reuters, y_test_reuters = padding(preprocessedData_Reuters)
```

In [115]:
```
word_size = 1000
embed_size = 500
imdb_model=tf.keras.Sequential()
imdb_model.add(tf.keras.layers.Embedding(word_size, embed_size, input_shape=(x_train_cnn.shape[1],)))
imdb_model.add(tf.keras.layers.LSTM(units=128, activation='tanh'))
# Output Layer
imdb_model.add(tf.keras.layers.Dense(units=5, activation='sigmoid'))
imdb_model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

In [116]:
```
"""#### Training the model"""

imdb_model.fit(x_train_cnn, y_train_cnn, epochs=25, batch_size=128)
```

Testing on FOX News

In [117]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_test_fox, y_test_fox)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_test_fox, batch_size=200, verbose=2)
report = classification_report(y_test_fox, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
50/50 [==============================] - 16s 315ms/step - loss: 0.3996 - accuracy: 0.5726
RNN Test accuracy score: 0.5726010203361511
8/8 - 10s - 10s/epoch - 1s/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       114
           1       0.00      0.00      0.00        48
           2       0.00      0.00      0.00       118
           3       0.77      0.67      0.72      1051
           4       0.22      0.01      0.02       253

   micro avg       0.74      0.45      0.56      1584
   macro avg       0.20      0.14      0.15      1584
weighted avg       0.54      0.45      0.48      1584
 samples avg       0.45      0.45      0.45      1584
```

click to expand output; double click to hide output

9.2 Train on left and test on neutral:

Testing on Reuters News

In [118]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_test_reuters, y_test_reuters)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_test_reuters, batch_size=200, verbose=2)
report = classification_report(y_test_reuters, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
42/42 [==============================] - 18s 423ms/step - loss: 0.7659 - accuracy: 0.2635
RNN Test accuracy score: 0.2635135054588318
7/7 - 8s - 8s/epoch - 1s/step
              precision    recall  f1-score   support

           0       0.02      0.00      0.01       299
           1       0.00      0.00      0.00       107
           2       0.00      0.00      0.00       213
           3       0.39      0.61      0.47       441
           4       0.00      0.00      0.00       272

   micro avg       0.36      0.20      0.26      1332
   macro avg       0.08      0.12      0.10      1332
weighted avg       0.13      0.20      0.16      1332
 samples avg       0.20      0.20      0.20      1332
```

9.3 Train on right and test on left:

Training on FOX News

In [119]:
```
Train_X_FOX, Train_Y_FOX = preprocessedData_Fox['clean_text'], preprocessedData_Fox['topicEncoded']
Test_X_CNN, Test_Y_CNN = preprocessedData_CNN['clean_text'], preprocessedData_CNN['topicEncoded']
Test_X_Reuters, Test_Y_Reuters = preprocessedData_Reuters['clean_text'], preprocessedData_Reuters['topicEncoded']
x_train_fox, y_train_fox = padding(preprocessedData_Fox)
x_test_cnn, y_test_cnn = padding(preprocessedData_CNN)
x_test_reuters, y_test_reuters = padding(preprocessedData_Reuters)
```

In [120]:
```
word_size = 1000
embed_size = 500
imdb_model=tf.keras.Sequential()
imdb_model.add(tf.keras.layers.Embedding(word_size, embed_size, input_shape=(x_train_fox.shape[1],)))
imdb_model.add(tf.keras.layers.LSTM(units=128, activation='tanh'))
# Output Layer
imdb_model.add(tf.keras.layers.Dense(units=5, activation='sigmoid'))
imdb_model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

In [121]:
```
"""#### Training the model"""

imdb_model.fit(x_train_fox, y_train_fox, epochs=25, batch_size=128)
```

Testing on CNN News

In [123]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_test_cnn, y_test_cnn)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_test_cnn, batch_size=200, verbose=2)
report = classification_report(y_test_cnn, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
48/48 [==============================] - 13s 278ms/step - loss: 0.6265 - accuracy: 0.4845
RNN Test accuracy score: 48.45089018344879
8/8 - 9s - 9s/epoch - 1s/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       237
           1       0.00      0.00      0.00       113
           2       0.00      0.00      0.00       151
           3       0.58      0.78      0.66       832
           4       0.17      0.07      0.09       184

   micro avg       0.56      0.43      0.49      1517
   macro avg       0.15      0.17      0.15      1517
weighted avg       0.34      0.43      0.38      1517
 samples avg       0.43      0.43      0.43      1517
```

9.4 Train on right and test on neutral:

Testing on Reuters News

In [124]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_test_reuters, y_test_reuters)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_test_reuters, batch_size=200, verbose=2)
report = classification_report(y_test_reuters, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
42/42 [==============================] - 13s 308ms/step - loss: 0.8277 - accuracy: 0.3348
RNN Test accuracy score: 33.483484387397766
7/7 - 8s - 8s/epoch - 1s/step
              precision    recall  f1-score   support

           0       0.50      0.00      0.01       299
           1       0.00      0.00      0.00       107
           2       0.00      0.00      0.00       213
           3       0.38      0.78      0.51       441
           4       0.14      0.02      0.04       272

   micro avg       0.37      0.26      0.31      1332
   macro avg       0.20      0.16      0.11      1332
weighted avg       0.27      0.26      0.18      1332
 samples avg       0.26      0.26      0.26      1332
```

9.5 Train on neutral and test on left:

Training on Reuters News

In [125]:
```
Train_X_Reuters, Train_Y_Reuters = preprocessedData_Reuters['clean_text'], preprocessedData_Reuters['topicEncoded']
Test_X_CNN, Test_Y_CNN = preprocessedData_CNN['clean_text'], preprocessedData_CNN['topicEncoded']
Test_X_FOX, Test_Y_FOX = preprocessedData_Fox['clean_text'], preprocessedData_Fox['topicEncoded']
x_train_reuters, y_train_reuters = padding(preprocessedData_Reuters)
x_test_cnn, y_test_cnn = padding(preprocessedData_CNN)
x_test_fox, y_test_fox = padding(preprocessedData_Fox)
```

In [126]:
```
word_size = 1000
embed_size = 500
imdb_model=tf.keras.Sequential()
imdb_model.add(tf.keras.layers.Embedding(word_size, embed_size, input_shape=(x_train_reuters.shape[1],)))
imdb_model.add(tf.keras.layers.LSTM(units=128, activation='tanh'))
# Output Layer
imdb_model.add(tf.keras.layers.Dense(units=5, activation='sigmoid'))
imdb_model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

In [127]:
```
"""#### Training the model"""

imdb_model.fit(x_train_reuters, y_train_reuters, epochs=25, batch_size=128)
```

Testing on CNN News

In [128]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_test_cnn, y_test_cnn)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_test_cnn, batch_size=200, verbose=2)
report = classification_report(y_test_cnn, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
48/48 [==============================] - 17s 356ms/step - loss: 0.7643 - accuracy: 0.3415
RNN Test accuracy score: 34.14634168148041
8/8 - 9s - 9s/epoch - 1s/step
              precision    recall  f1-score   support

           0       0.11      0.07      0.08       237
           1       0.00      0.00      0.00       113
           2       0.12      0.09      0.10       151
           3       0.67      0.43      0.52       832
           4       0.00      0.00      0.00       184

   micro avg       0.42      0.25      0.32      1517
   macro avg       0.18      0.12      0.14      1517
weighted avg       0.40      0.25      0.31      1517
 samples avg       0.25      0.25      0.25      1517
```

9.6 Train on neutral and test on right:

Testing on FOX News

In [129]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_test_fox, y_test_fox)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_test_fox, batch_size=200, verbose=2)
report = classification_report(y_test_fox, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
50/50 [==============================] - 14s 274ms/step - loss: 0.6823 - accuracy: 0.3813
RNN Test accuracy score: 38.131314516067505
8/8 - 9s - 9s/epoch - 1s/step
              precision    recall  f1-score   support

           0       0.07      0.10      0.08       114
           1       0.00      0.00      0.00        48
           2       0.07      0.05      0.06       118
           3       0.73      0.42      0.53      1051
           4       0.04      0.01      0.02       253

   micro avg       0.49      0.29      0.37      1584
   macro avg       0.18      0.12      0.14      1584
weighted avg       0.50      0.29      0.37      1584
 samples avg       0.29      0.29      0.29      1584
```

9.7 Train on neutral and test on neutral:

Training and Testing on Reuters News

In [130]:
```
test_loss, test_acurracy = imdb_model.evaluate(x_train_reuters, y_train_reuters)

print("RNN Test accuracy score: {}".format(test_acurracy*100))

y_pred=imdb_model.predict(x_train_reuters, batch_size=200, verbose=2)
report = classification_report(y_train_reuters, y_pred.round())
print(report)
# print(classification_report(Test_Y_Reuters,test_acurracy))
```

```
42/42 [==============================] - 13s 305ms/step - loss: 0.0878 - accuracy: 0.9482
RNN Test accuracy score: 94.81981992721558
7/7 - 8s - 8s/epoch - 1s/step
              precision    recall  f1-score   support

           0       1.00      0.95      0.97       299
           1       1.00      0.52      0.69       107
           2       0.94      0.87      0.90       213
           3       0.96      1.00      0.98       441
           4       0.96      0.87      0.91       272

   micro avg       0.97      0.90      0.93      1332
   macro avg       0.97      0.84      0.89      1332
weighted avg       0.97      0.90      0.93      1332
 samples avg       0.90      0.90      0.90      1332
```

Conclusion

The goal of this work was to identify the influence of ideological bias introduced by a social media handle creator by developing topic classification models using several classifiers that were trained and evaluated on various datasets. The NLP Tasks are severely harmed by the induction of ideological bias existing in the dataset, as seen by the classifier models' poor performance on the dataset they are not trained up on. As a result, according to the project's conclusions, we must do algorithmic bias detection and mitigation to reduce the bias in the trained model in order to train a robust AI model from a social media perspective. Inorder to detect the bias in the model, we now have bias police datasets at our disposal. When tested and trained on them, poor performance indicates bias in the model. There is also an emerging technique in machine learning called counterfactual exploitation which we can exploit to reduce the relationship between feature vectors and dependent Target Variables.

Github : Code Availability

The entire project is pushed to github under the repository name ideological bias. The link to the repository is **https://github.com/shanfer1/Idelogical-Bias**

References

- https://gist.github.com/pb111/88545fa33780928694388779af23bf58
- https://github.com/sajjadirn/Text-Classification-using-random-forest-classifier/blob/master/19200123_SajjadZulphekari_Proj2.ipynb
- https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9
- https://medium.com/dataseries/how-to-scrape-millions-of-tweets-using-snscrape-195ee3594721
- https://cnvrg.io/cnn-sentence-classification/
- https://github.com/kristina-arezina/Text-Classification-using-Logistic-Regression/blob/main/Text_Classification_using_Logistic_Regression.ipynb
- https://www.kaggle.com/code/au1206/text-classification-using-cnn
- https://github.com/saiharshithreddy/Text-classification-and-summarization/blob/master/TextClassification/TextClassification_LogisticRegression.ipynb
- https://github.com/aaryaab/Text-Classification-with-RNN/blob/main/rnn_with_imdb.py
- https://stackoverflow.com/questions/44273249/in-keras-what-exactly-am-i-configuring-when-i-create-a-stateful-lstm-layer-wi
- https://www.kaggle.com/code/foolofatook/news-classification-using-bert
- https://www.kaggle.com/code/sainijagjit/text-classification-using-svm/notebook