Project Charter

Project Name	A Text-Mining approach to Hashtag Optimization on social networks.
Project Guide	Mr. Agniva Das, Department of Statistics, SPU, Anand
Co-Leader	Jitesh Ramchandra Pawar
Project Description	To develop of functional of classifier of accurate an automatic sentiments classification of trending hashtag tweets on a twitter. Given a message, classifies whether the tweets is of positive, negative or neutral sentiments. Also, to predict next 10 days sentiments.

Report

SR.NO	Month	Week	Hours Spend	No. of meeting	Work
1.	Dec	Third	0	0	Project topic given
		Third	4	0	Project approval
		Fifth	10	1	Project topic discussion
2.	Jan	First	16	1	Project topic discussion
		Third	20	2	Study research paper & projects
3.	Feb	First	22	1	Study projects
		Second	20	2	Collecting information On Social Sites.
		Third	16	1	Data collection
4.	March	First	30	2	Data Scraping
		Second	28	3	Sentiment Analysis
		Forth	24	2	Time Series Analysis
5.	April	First	26	1	Time Series Analysis
		second	18	2	Study of ARIMA model
		Third	25	2	Create word file
6.	May	First	20	1	Final submission word file & ppt

Project Charter

Project Name	A Text-Mining approach to Hashtag Optimization on social networks.
Project Guide	Mr. Agniva Das, Department of Statistics, SPU, Anand
Co-Leader	Sanket Sunil Andalkar
Project Description	To develop of functional of classifier of accurate an automatic sentiments classification of trending hashtag tweets on a twitter. Given a message, classifies whether the tweets is of positive, negative or neutral sentiments. Also, to predict next 10 days sentiments.

Report

SR.NO	Month	Week	Hours Spend	No. of meeting	Work
1.	Dec	Third	0	0	Project topic given
		Third	4	0	Project approval
		Fifth	10	1	Project topic discussion
2.	Jan	First	16	1	Project topic discussion
		Third	20	2	Study research paper & projects
3.	Feb	First	22	1	Study projects
		Second	20	2	Collecting information On Social Sites.
		Third	16	1	Data collection
4.	March	First	30	2	Data Scraping
		Second	28	3	Sentiment Analysis
		Forth	24	2	Time Series Analysis
5.	April	First	26	1	Time Series Analysis
		second	18	2	Study of ARIMA model
		Third	25	2	Create word file
6.	May	First	20	1	Final submission word file & ppt

A Text-Mining approach to Hashtag Optimization on social networks.

ACKNOWLEDGEMENT

We are extremely thankful to our guide Mr. Agniva Das for his guidance and support. We are extremely glad to have as our project guide. We would like to thank Prof. Jyoti M. Divecha, Head of the Department of Statistics for extending her help and support.

We also thank the other professors of the department and the entire staff of Department of Statistics, Sardar Patel University, Vallabh Vidyanagar: - 388 120 for providing us with all the necessary infrastructure and cooperating with us at every stage of our project. Also genial thanks to our friends for providing us with the amenities and co-operation for our project.

Our special thanks to everyone who was included in our project for providing us with their vital information, precious time and their spirituous support.

A

PROJECT REPORT

(PS04CAST23)

On

A Text-Mining approach to Hashtag Optimization on social network

By

Jitesh R. Pawar

Sanket S. Andalkar

PROJECT GUIDE

Mr. Agniva Das

"MASTER OF APPLIED STATISTICS"

DEPARTMENT OF STATISTICS

SARDAR PATEL UNIVERSITY

VALLABH VIDYANAGAR

2020-2021

CERTIFICATE

This is to certify that Mr. Jitesh Ramchandra Pawar, Roll No. 17, Exam No. 25, "Master of Science in Applied Statistics", Semester-4 has successfully completed his project entitled "A Text-Mining approach to Hashtag Optimization on social networks" for PS04CAST23 in term 2020-2021.

Date:

Mr. Agniva Das **Prof. Jyoti M. Divecha**

(Project Guide) **(Head of Department)**

CERTIFICATE

This is to certify that Mr. Sanket Sunil Andalkar, Roll No. 34 , Exam No. 37 "Master of Science in Applied Statistics", Semester-4 has successfully completed his project entitled "A Text-Mining approach to Hashtag Optimization on social networks" for PS04CAST23 in term 2020-2021.

Date:

Mr. Agniva Das

(Project Guide)

Prof. Jyoti M. Divecha

(Head of Department)

INDEX

Abstract

Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, attitudes, an emotion expressed in written language. It is one of the most active research in natural language processing and text mining recent years.

The aim of this project is to develop of functional of classifier of accurate an automatic sentiments classification of trending hashtag tweets. Given a message, classifies whether the tweets is of positive, negative or neutral sentiments.

In recent years social media have emerged as popular platform for people to share their thoughts and opinions on all kind of topics. Tracking opinion over time is a powerful tool that can be used for sentiment prediction or to detect the possible reasons of a sentiment change. In this study, we explore conventional time series analysis methods and their applicability on topic and sentiment trend analysis.

Preface

"Experience is a best teacher" this saying had played a guiding role in including as a part of curricular of Master of Science in Statistics. Knowledge in itself is a continuous process. Getting practical knowledge is an important thing for an individual in a business concern. In the competition prevailing industry, a total awareness is the first and foremost thing for all aspects. Working smart seems to be as important as working harder and longer. Hence, practical expose at M.sc science after training in certain aspect has provide a boon for us.

This project provides an opportunity and platform to know the current situation and the behaviour of environment. The goal of the project is to give corporate exposure to the student as well as to give him an opportunity to apply theory into practice. The real business problems are drastically different from classroom case solving, project aims at providing little insight into working of an organization.

Objectives

- **To Analyze Sentiment for a particular Trending Hashtag by using Sentiment Analysis.**
- **To predict No. of tweets in daily bases using Time Series Analysis.**
- **To Analyze which factors, affect on Trending Hashtag.**
- **To predict the sentiments for next days in particular type.**


What the
#
#Hashtag?!

Hashtag

A hashtag is a number symbol **(#)** used to label keywords in a tweet.

The name "hashtag" was coined by Twitter and combines the word "hash" (another name for the number symbol) and "tag," since it is used to tag certain words.

In order to tag a keyword in a Twitter post, simply type a number symbol (Shift+3) immediately before the word.

For example: you can tag the word “Justice" in a tweet by typing **"#Justice."**

Text Mining

Text mining (also known as text analysis), is the process of transforming unstructured text into structured data for easy analysis.

Text mining uses **natural language processing (NLP),** allowing machines to understand the human language and process it automatically.

Text mining, also referred to as **text data mining**, similar to **text analytics**, is the process of deriving high-quality information from text. It involves "the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources." Written resources may include websites, books, emails, reviews, and articles. High-quality information is typically obtained by devising patterns and trends by means such as statistical pattern learning.

Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output.

'High quality' in text mining usually refers to some combination of relevance, novelty, and interest. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modelling (i.e., learning relations between named entities).

Text analysis involves information retrieval, lexical analysis to study word frequency distributions, pattern recognition, tagging/annotation, information extraction, data mining techniques including link and association analysis, visualization, and predictive analytics. The overarching goal is, essentially, to turn text into data for analysis, via application of natural language processing (NLP), different types of algorithms and analytical methods. An important phase of this process is the interpretation of the gathered information.

Applications

Text mining technology is now broadly applied to a wide variety of government, research, and business needs.

All these groups may use text mining for records management and searching documents relevant to their daily activities. Legal professionals may use text mining for e-discovery, for example. Governments and military groups use text mining for national security and intelligence purposes. Scientific researchers incorporate text mining approaches into efforts to organize large sets of text data (i.e., addressing the problem of unstructured data), to determine ideas communicated through text (e.g., sentiment analysis in social media) and to support scientific discovery in fields such as the life sciences and bioinformatics. In business, applications are used to support competitive intelligence and automated ad placement, among numerous other activities.

Natural language Processing (NLP).

Natural Language Processing is defined as understanding text or speech with the aid of any software program or machine.

An analogy is that humans can interact and understand each other views and reply with the perfect answer. In NLP, a computer made interactions, not a human.

Applications of NLP

- **Sentiment Analysis**
- **Chatbots**
- **Machine Translation**
- **Auto-Correct**
- **Spell-Checking**
- **Speech Recognition**
- **Keyword-Search**
- **Virtual Assistants**

What is Data Scraping?

- Data scraping is commonly defined as a system where a technology extracts data from a particular codebase or program. Data scraping provides results for a variety of uses and automates aspects of data aggregation.
- Data scraping, also known as web Scrapping.

Advantages of Web Scraping:

- Data extraction is automated.
- Copying and pasting the data manually is absolutely a pain. Actually, it is simply not possible to copy/paste a large amount of data when one needs to extract from millions of web pages on a regular basis.
- Web scraping can extract data automatically with zero human factors included.

I. The information collected is much more accurate:

Another advantage of web scraping is that greatly increases the accuracy of data extraction, as it eliminates human error in this process.

II. Get clean and structured data

After gathering data there usually follows cleaning and reorganizing it, because the data collected is not structured and ready to use.

Web scraping tools convert unstructured and semi-structured data into structured data, and web page information is reorganized into presentable formats.

Scraping Tweets Using Twitter API's

Web scraping is an essential skill for any data scientist to gather the data from various resources. Twitter is a great tool to gather tons of quality data. One thing where I find twitter data very helpful is in sentiment analysis. Twitter makes it really easy to gather publicly available data using its APIs.

API (Application Programming Interface)

An **API** is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange.

To create API's, follow below steps:

1. Please create first your Twitter developer Account. if you have don't have an account.

2. Create New Application.

3. After creating application you can copy your secret key and get access token.

Link:

https://developer.twitter.com/en/docs/apps/app-management

- We will be using tweepy library to extract data from Twitter.

  import tweepy

- Now we need to access our API keys:

  consumerKey = "your consumerKey "

  consumerSecret = "your consumerSecret"

  accessToken = "your accessToken"

  accessTokenSecret = "your accessTokenSecret"

- Next step is Authentication using OAuthHandler

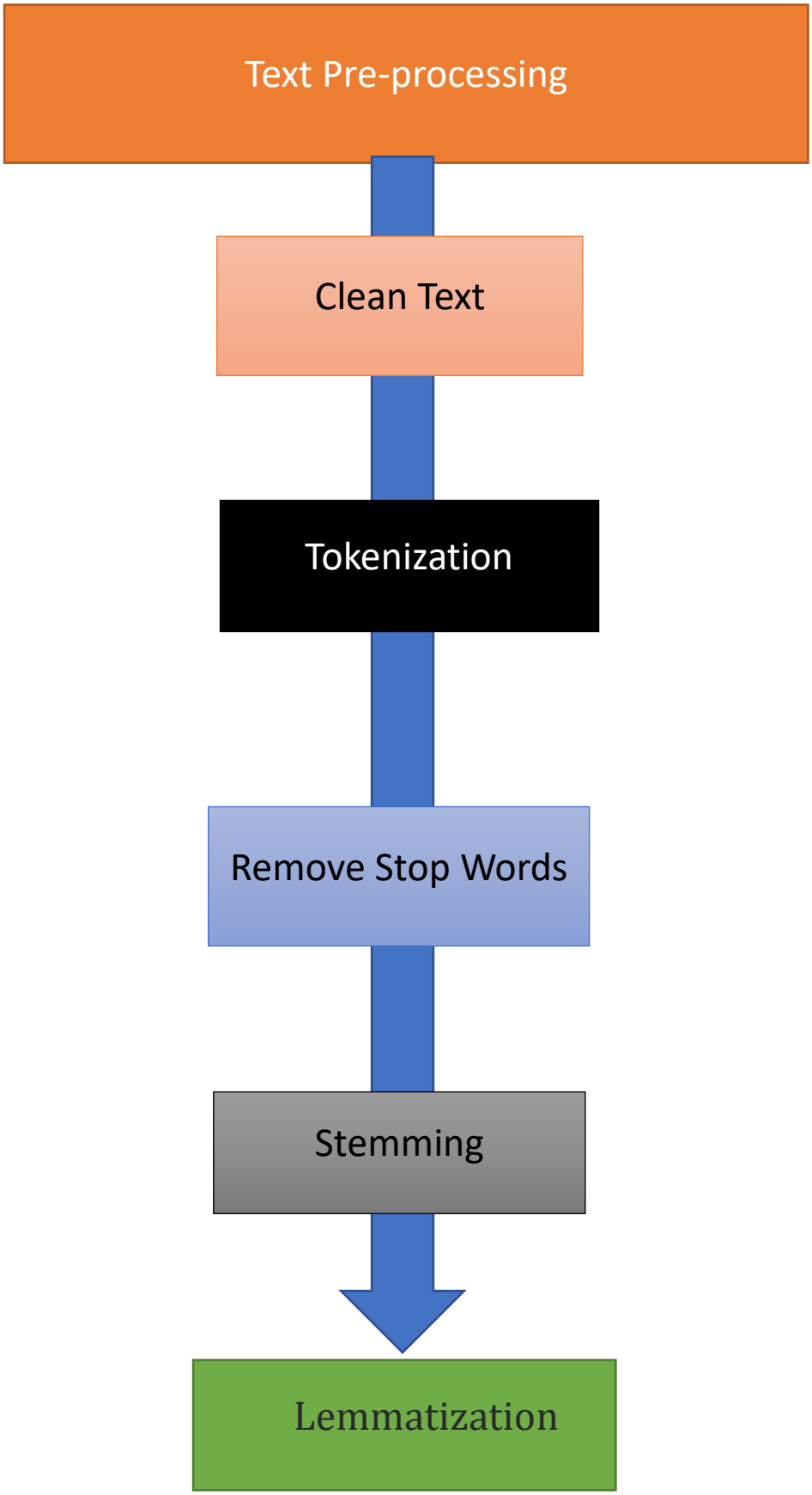
```
auth=tweepy.OAuthHandler(consumerKey,consumerSecret)
auth.set_access_token(accessToken,accessTokenSecret)
api=tweepy.API(auth,wait_on_rate_limit = True)
```

- **Tweets Scrapping Code**

```
search_word = ["#TAG"]
date_since = "2021-01-31"
tweets=tweepy.Cursor(api.search,q=search_word,
lang='en',since=date_since).items(100)
tweet_df=pd.DataFrame([[tweet.text] for tweet in
tweets],columns=['tweet'])
tweet_df.head
```

NLTK Library

- NLTK stands for Natural Language Tool Kit and is one of the most powerful NLP libraries.
- Natural Language Tool Kit is the mother of all NLP libraries.
- Machine Learning projects and data pre-processing use NLTK.
- It is used in data pre-processing.


Text Pre-processing
Clean Text
Tokenization
Remove Stop Words
Stemming
Lemmatization

Text Pre-processing in Natural Language Processing

- In NLP, text preprocessing is the first step in the process of building a model.
- The various text preprocessing steps are:
  - Clean Text
  - Tokenization
  - Lower casing
  - Stop words removal
  - Stemming
  - Lemmatization

```
#create a function to clean tweets
def cleanTxt(text):
text=re.sub('@[A-Za-z0-9]    +','',text)#Removing @mention

text = re.sub('#','',text) #Removing hashtag

text = re.sub('RT[\s]+', '', text)#Removing RT

text = re.sub('https?:\/\/\s+', '', text)#Removing hyperlink

text = ''.join([char.lower() for charin text if char not in string.punctuation])
#Removing punctuation and normalization
return text


# Clean the tweets
tweet_df['tweet_clean'] = tweet_df['tweet'].apply(cleanTxt)

# Show the clean Tweets
  tweet_df
```

Tokenization

- Splitting the sentence into words.
- Tokenization is the first step in text analytics. It is the process of breaking down a textual content paragraph into smaller chunks.
- A token is a single entity which is building block for sentence or paragraph. For example, each word is a token if a sentence “tokenized” in words.
- Every sentence is a token if a paragraph is tokenized into a sentence.

```
## Tokenization
    def tokenize(text):
    tokens = re.split('\W+'text)
    return tokens
    tweet_df['tweet_clean']=tweet_df['tweet_clean'].apply(tokenize)
    tweet_df
```

Lower casing

- Converting a word to lower case (NLP -> nlp).
- Words like Book and book mean the same but when not converted to the lower case those two are represented as two different words in the vector space model (resulting in more dimensions).

Stop words removal

- Stop words are very commonly used words (a, an, the, etc.) in the documents.
- These words do not really signify any importance as they do not help in distinguishing two documents.

```
## Remove stop words
nltk.download('stopwords')
stopwords = nltk.corpus.stop    words.words('english')
def remove_stopwords(text):
txt_clean = [word for word in text if word not in
stopwords]
 return txt_clean


tweet_df['tweet_clean'] = tweet_df['tweet_clean'].apply(remove_stopwords)
tweet_df
```

Stemming

- Stemming is the process of removing a part of a word, or reducing a word to its stem or root.
- A stemming algorithm might also reduce the words fishing, fished, and fisher to the stem fish.
- The stem need not be a word, for example the Porter algorithm reduces, argue, argued, argues, arguing, and argus to the stem argu.

```
## stemming

from nltk.stem import PorterStemmer
ps = PorterStemmer()
def stemming(text):
stem_text = [ps.stem(word)for word in text]
return stem_text

tweet_df['tweet_clean'] = tweet_df['tweet_clean'].apply(stemming)
tweet_df
```

Sentiment analysis

Sentiment analysis (or **opinion mining**) is a natural language processing technique used to determine whether data is positive, negative or neutral.

Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative our neutral.

Sentiment analysis is used in business intelligence to understand the subjective reasons why consumers are or are not responding to something (ex. why are consumers buying a product? What do they think of the user experience? Did customer service support meet their expectations?). Sentiment analysis can also be used in the areas of political science, sociology, and psychology to analyse trends, ideological bias, opinions, gauge reactions, etc.

Sentiment Analysis Algorithms

- Rule based (Lexicon-based)
- Automatic (ML based)

• Lexicon-based Techniques

Often, you may not have the convenience of a well labelled training dataset.
In those situations, you need to use unsupervised techniques for prediction the sentiment by using knowledgebases, ontologies databases and lexicon that have detailed information specially curated and prepared just for sentiment analysis.

As mention, lexicon is dictionary, vocabulary or a book of words. In our case lexicon are special dictionaries or vocabularies that have been created for analysing sentiment.

VADER Lexicon:

It is a lexicon with a Rule based sentiment analysis frame-work that was specially build for analysing sentiment from social media resources.

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

tweet_df['tweet_clean']=tweet_df['tweet_clean'].apply(lambda x: ' '.join(map(str
, x)))

tweet_df['score']=tweet_df['tweet_clean'].apply(lambda tweet: sid.polarity_scor
es(tweet))
tweet_df.head

tweet_df['compound']=tweet_df['score'].apply(lambda score_dict: score_dict['co
mpound'])
tweet_df.head()


def getAnalysis(score):

  if score < 0:
  return 'Negative'
  elif score == 0:
  return 'Neutral'
  else:
  return 'Positive'
tweet_df['Analysis'] = tweet_df['compound'].apply(getAnalysis)
tweet_df['Analysis'].value_counts()
```

Exploratory Data Analysis (EDA)

- In statistics, **exploratory data analysis** is an approach to analysing datasets to summarize their main characteristics, often using statistical graphics and other data visualization methods.
- A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.
- Exploratory data analysis (EDA) is used by data scientists to analyse and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.
- EDA is primarily used to see what data can reveal beyond the formal modelling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

  The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

- **Described Data**

```
In [8]: data.describe()
```

Out[8]:

	No_OF_Tweets	Insta_Post	Fac_Post	Pos	Neg	Neu	News_Agency	Organization	Naturally	trend_by_Creator
count	230.000000	2.300000e+02	230.000000	230.000000	230.000000	230.000000	230.000000	230.000000	230.000000	230.000000
mean	29616.086957	7.172017e+04	11126.308696	421.234783	173.678261	405.295652	0.104348	0.239130	0.752174	0.252174
std	49237.201936	3.318070e+05	58845.842964	194.019057	174.420941	189.318842	0.306378	0.427483	0.432692	0.435208
min	1100.000000	0.000000e+00	0.000000	12.000000	2.000000	6.000000	0.000000	0.000000	0.000000	0.000000
25%	3900.000000	2.650000e+01	30.250000	283.000000	46.000000	269.000000	0.000000	0.000000	1.000000	0.000000
50%	11100.000000	5.875000e+02	80.500000	413.000000	100.000000	405.500000	0.000000	0.000000	1.000000	0.000000
75%	32325.000000	1.472500e+04	3775.000000	547.250000	265.000000	520.000000	0.000000	0.000000	1.000000	0.750000
max	325000.000000	4.200000e+06	831000.000000	951.000000	862.000000	908.000000	1.000000	1.000000	1.000000	1.000000

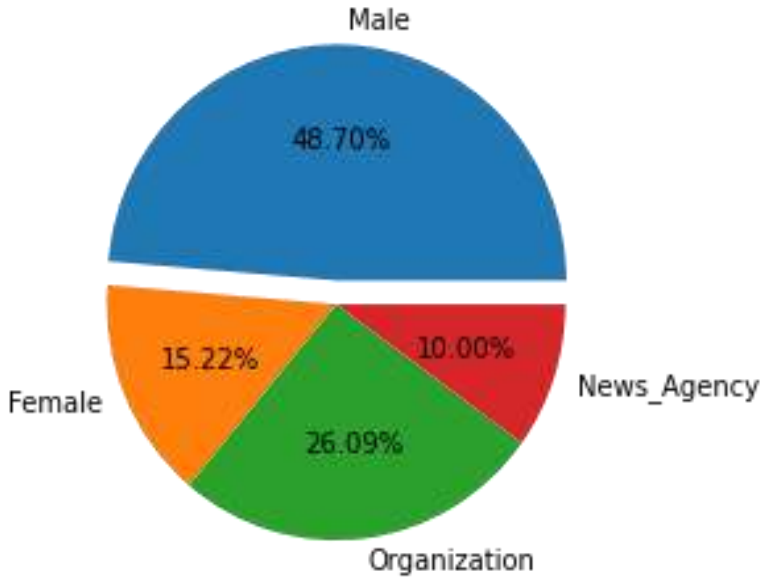
- **Checking NA Values**

```
In [11]: pd.isna(data).sum()

Out[11]: Date              0
         Hashtag           0
         Type              0
         No_OF_Tweets      0
         Insta_Post        0
         Fac_Post          0
         Pos               0
         Neg               0
         Neu               0
         creator_name      3
         creator_Id        0
         creator           0
         create_date       0
         News _Agency      0
         Organization      0
         Naturally         0
         Sentiment         0
         trend_by_Creator  0
         dtype: int64
```

Who are creators?

```
In [9]: Creator_Count = data.creator.value_counts()
        Creator_Count

Out[9]: Male            112
        Organization     60
        Female           35
        News_Agency      23
        Name: creator, dtype: int64
```

Pie Chart



As above pie chart we can say that, the most of Hashtag creators are **Male.**

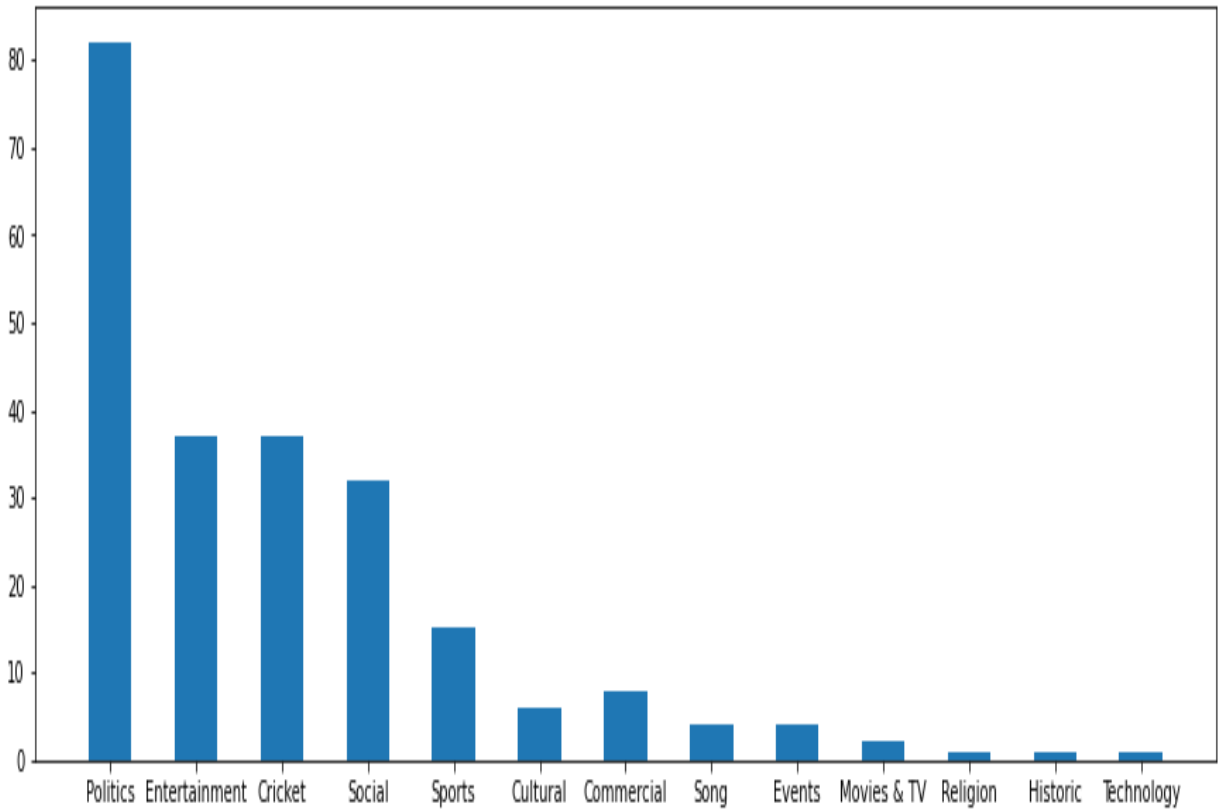
Which Type has a more frequency?

```
In [12]: Type_Count = data.Type.value_counts()
         Type_Count
```

```
Out[12]: Politics         82
         Entertainment    37
         Cricket          37
         Social           32
         Sports           15
         Commercial        8
         Cultural          6
         Song              4
         Events            4
         Movies & TV       2
         Historic          1
         Religion          1
         Technology        1
         Name: Type, dtype: int64
```

From the above table we can conclude that the politics type has more number of frequencies.

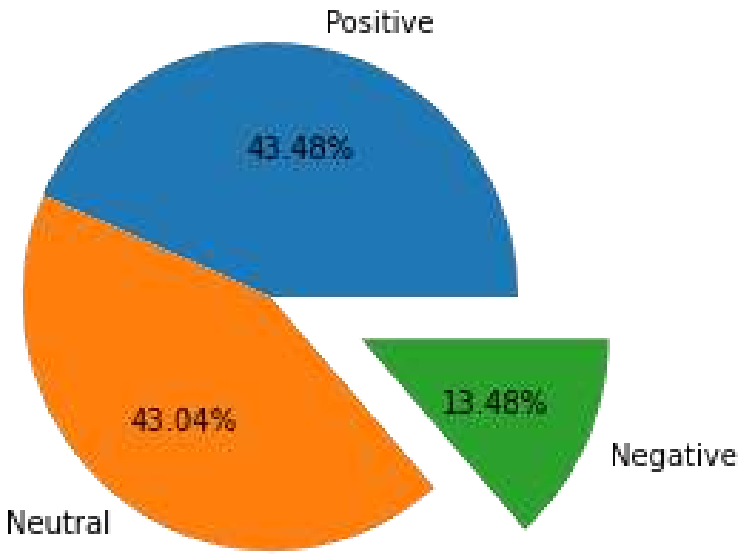
Bar Plot



Above Bar plot says that, **Politics, Entertainment Cricket** and **Social** are more trending topics while **Religion Historic and Technology** are less trending topics in India.

Hashtag Sentiments

```
In [19]: data.Sentiment.value_counts()

Out[19]: Neutral     100
         Positive     99
         Negative     31
         Name: Sentiment, dtype: int64
```

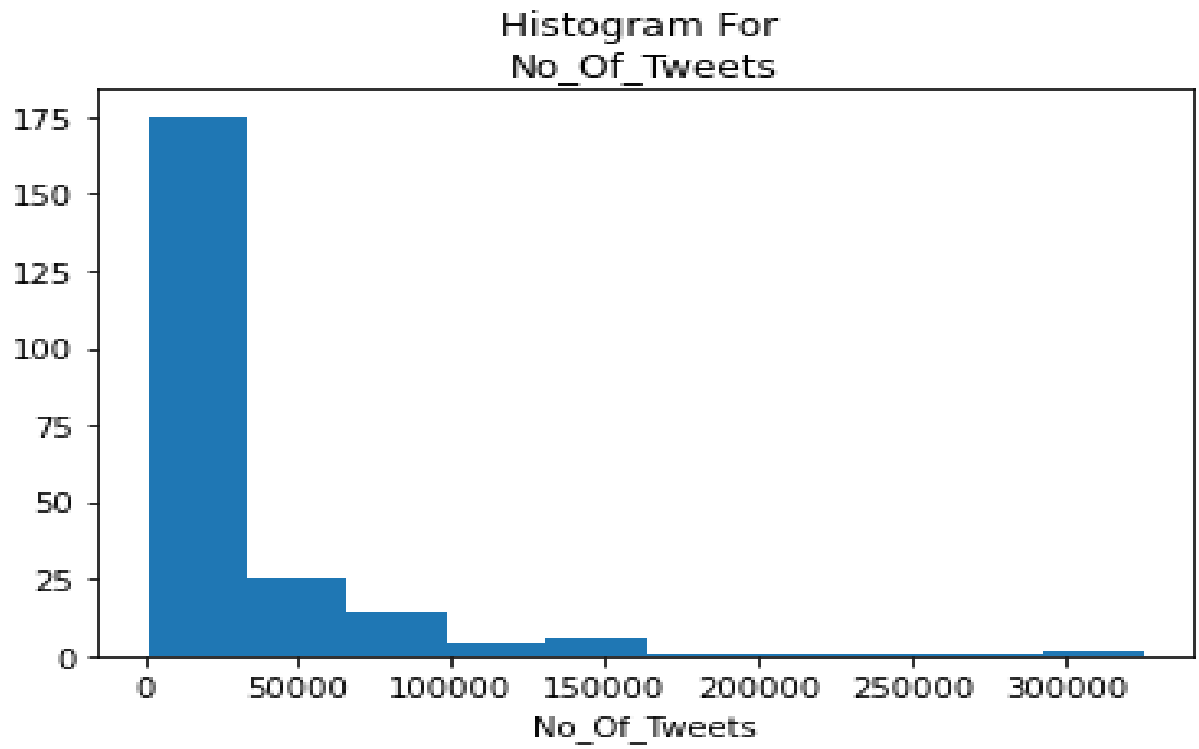


43.48 % Hashtags are Positive.

43.04 % Hashtags are Neutral.

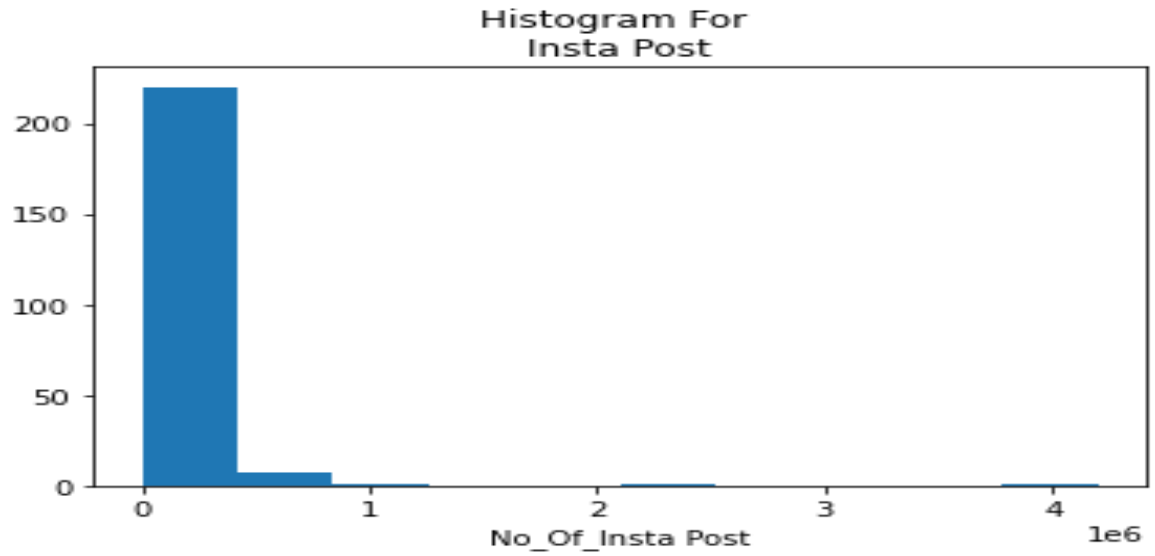
13.48 % hashtags are Negative.

Histograms for No. of Tweets



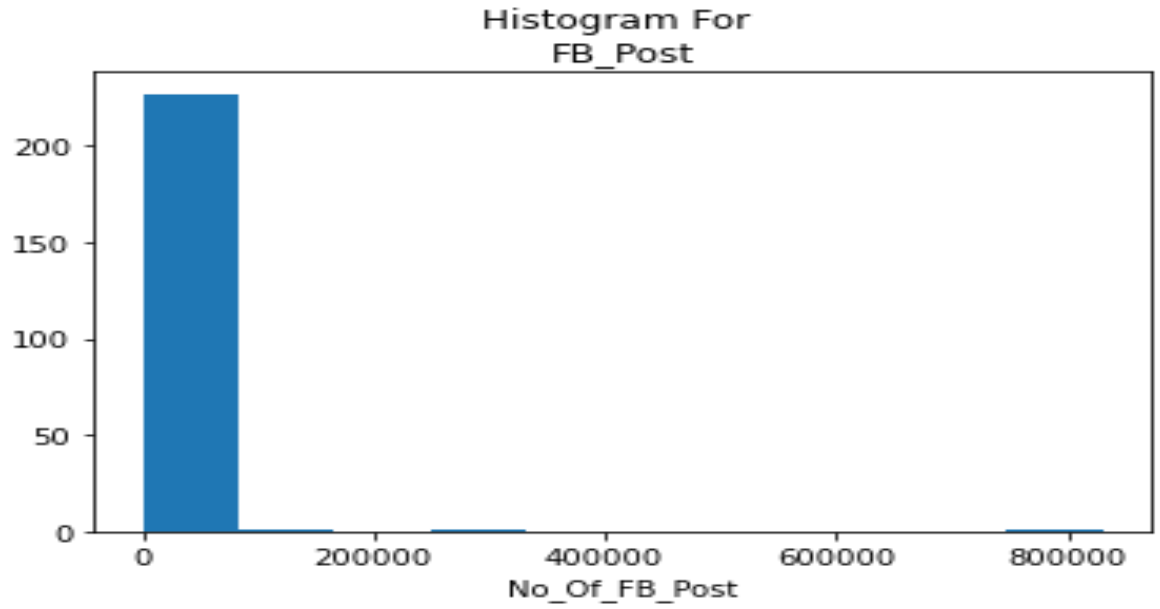
There are 76 % Hashtags lies between 0 to 30000 No of tweets.

Histogram for Instagram Post



There are almost 95 % Hashtags lies between 0 to 3.3 lac Instagram posts.

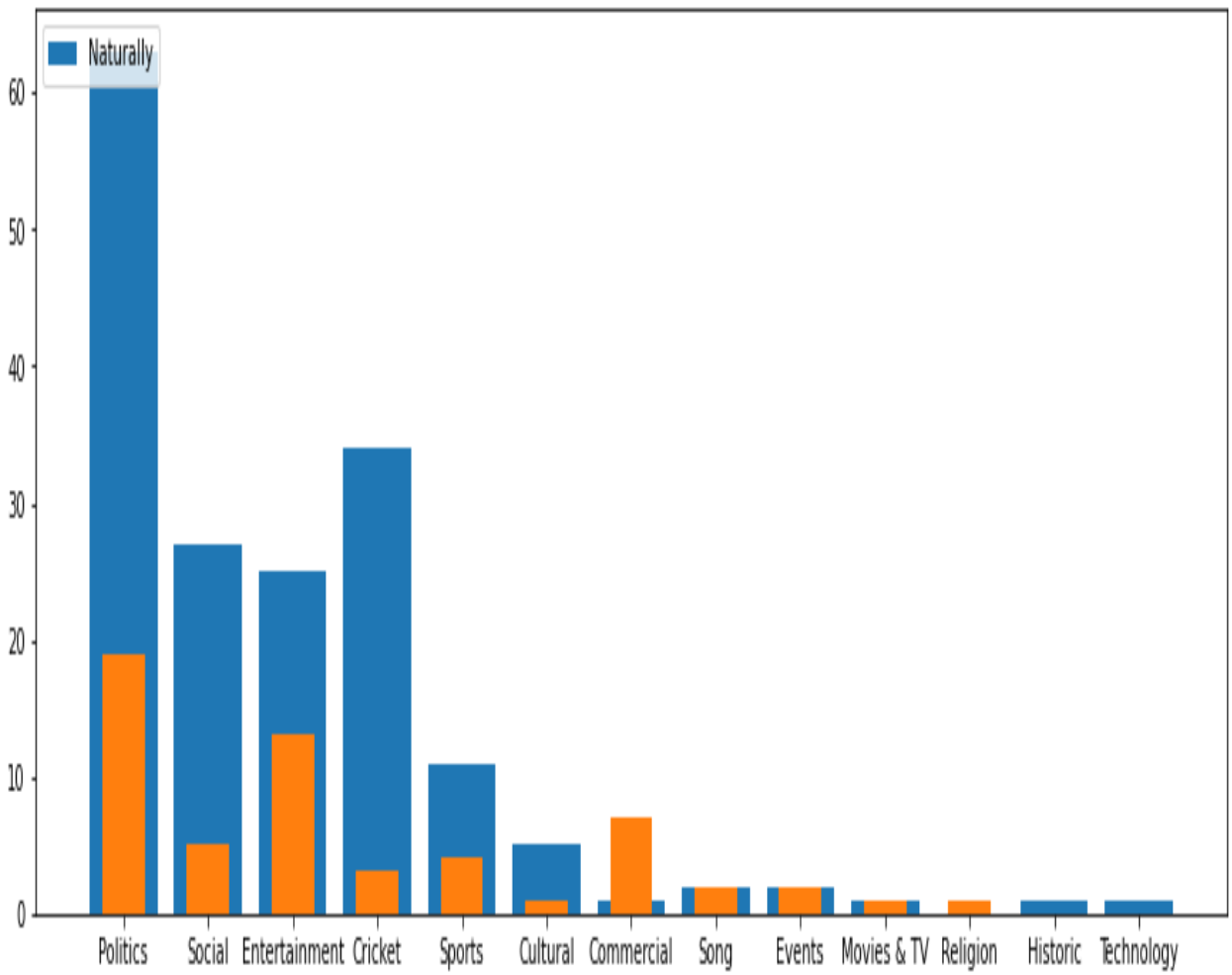
Histogram for Facebook Post



There are almost 98 % Hashtags lies between 0 to 1 lac Facebook posts.

How many Hashtags trend by creator?

Type	Naturally	Trend_by_Creator	Total
Commercial	1	7	8
Cricket	34	3	37
Cultural	5	1	6
Entertainment	25	13	37
Events	2	2	4
Historic	1	0	1
Movies & TV	1	1	2
Politics	63	19	82
Religion	0	1	1
Social	27	5	32
Song	2	2	4
Sports	11	4	15
Technology	1	0	1



Cricket, Cultural and Social these Hashtags types are trend by Naturally.

Entertainment and Politics are grown by Naturally as well as by creator.

Commercial Hashtags are often created by creator.

Who creates that trend?

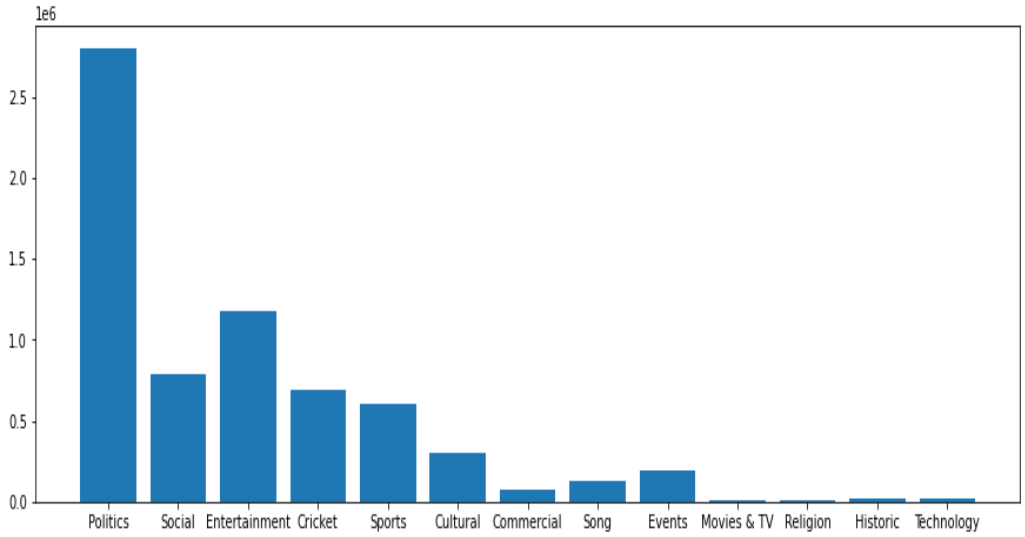
creator Type	Female	Male	News_Agency	Organization
Commercial	NaN	3.0	NaN	4.0
Cricket	1.0	1.0	0.0	1.0
Cultural	0.0	1.0	NaN	0.0
Entertainment	2.0	7.0	1.0	3.0
Events	NaN	2.0	NaN	NaN
Historic	NaN	0.0	NaN	NaN
Movies & TV	NaN	1.0	NaN	0.0
Politics	2.0	12.0	1.0	4.0
Religion	NaN	1.0	NaN	NaN
Social	1.0	1.0	0.0	3.0
Song	0.0	2.0	NaN	NaN
Sports	0.0	1.0	1.0	2.0
Technology	NaN	0.0	NaN	NaN

55 % trend by creator's Hashtags are made by **Male**.

No. of Tweets with respect to Type

Type	No_OF_Tweets
Commercial	70600
Cricket	694600
Cultural	300100
Entertainment	1177700
Events	189100
Historic	16800
Movies & TV	8600
Politics	2801500
Religion	8400
Social	791000
Song	129900
Sports	602200
Technology	21200

Bar Plot

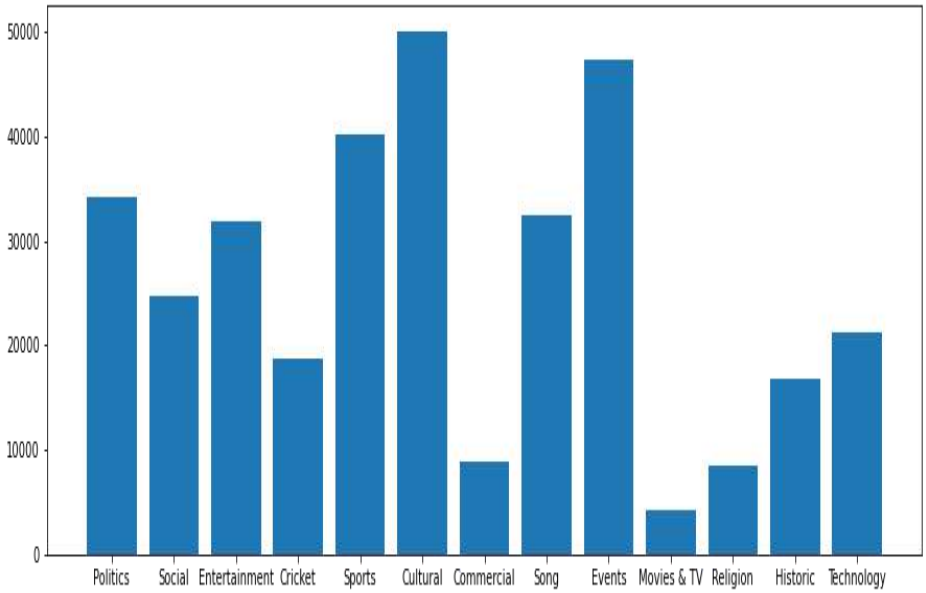


Most of tweets are on **Politics** and **Entertainment**.

Average No. of Tweets with respect to Type

Type	No_OF_Tweets
Commercial	8825.000000
Cricket	18772.972973
Cultural	50016.666667
Entertainment	31829.729730
Events	47275.000000
Historic	16800.000000
Movies & TV	4300.000000
Politics	34164.634146
Religion	8400.000000
Social	24718.750000
Song	32475.000000
Sports	40146.666667
Technology	21200.000000

Bar Plot



By average most of tweets are on **Cultural Events** and **Sports**.

Frequency table for creators

creator Type	Female	Male	News_Agency	Organization	All
Commercial	0	3	0	5	8
Cricket	5	18	7	7	37
Cultural	2	3	0	1	6
Entertainment	3	19	2	13	37
Events	0	4	0	0	4
Historic	0	1	0	0	1
Movies & TV	0	1	0	1	2
Politics	8	47	11	16	82
Religion	0	1	0	0	1
Social	15	4	1	12	32
Song	1	3	0	0	4
Sports	1	7	2	5	15
Technology	0	1	0	0	1
All	35	112	23	60	230

Crosstab for sentiment

Sentiment Type	Negative	Neutral	Positive	All
Commercial	0	3	5	8
Cricket	4	12	21	37
Cultural	1	3	2	6
Entertainment	4	14	19	37
Events	0	3	1	4
Historic	1	0	0	1
Movies & TV	0	1	1	2
Politics	20	38	24	82
Religion	0	1	0	1
Social	0	16	16	32
Song	0	2	2	4
Sports	1	6	8	15
Technology	0	1	0	1
All	31	100	99	230

Crosstab for creator's sentiment

Sentiment	Negative				Neutral				Positive				All
creator	Female	Male	News_Agency	Organization	Female	Male	News_Agency	Organization	Female	Male	News_Agency	Organization	
Type													
Commercial	0	0	0	0	0	1	0	2	0	2	0	3	8
Cricket	0	3	0	1	2	6	2	2	3	9	5	4	37
Cultural	1	0	0	0	0	2	0	1	1	1	0	0	6
Entertainment	0	2	1	1	1	8	0	5	2	9	1	7	37
Events	0	0	0	0	0	3	0	0	0	1	0	0	4
Historic	0	1	0	0	0	0	0	0	0	0	0	0	1
Movies & TV	0	0	0	0	0	0	0	1	0	1	0	0	2
Politics	2	10	3	5	6	19	6	7	0	18	2	4	82
Religion	0	0	0	0	0	1	0	0	0	0	0	0	1
Social	0	0	0	0	10	1	1	4	5	3	0	8	32
Song	0	0	0	0	1	1	0	0	0	2	0	0	4
Sports	0	0	1	0	1	3	1	1	0	4	0	4	15
Technology	0	0	0	0	0	1	0	0	0	0	0	0	1
All	3	16	5	7	21	46	10	23	11	50	8	30	230

Chi-square test for independence.

The Chi-Square test of independence is used to determine if there is a significant relationship between two nominal (categorical) variables.

The frequency of each category for one nominal variable is compared across the categories of the second nominal variable.

The data can be displayed in a contingency table where each row represents a category for one variable and each column represents a category for the other variable.

Hypothesis

H_0: There is no significance relationship between Creator and type.

Vs

H_1: There is significance relationship between Creator and type.

researchpy Library:

Using this library, we perform a chi-square test.

	Chi-square test	results
0	Pearson Chi-square (36.0) =	65.1092
1	p-value =	0.0021
2	Cramer's V =	0.3072

From the above table P-value corresponds to chi-square test is **0.0021,** indicate that there is significant relationship between creator and type.

Hypothesis

H_0: There is no significance relationship between Creator and Sentiment.

Vs

H_1: There is significance relationship between Creator and Sentiment**.**

	Chi-square test	results
0	Pearson Chi-square (6.0) =	6.9047
1	p-value =	0.3298
2	Cramer's V =	0.1225

From the above table P-value corresponds to chi-square test is **0.3298,** indicate that there is no significant relationship between creator and Sentiment.

Time Series Analysis

- Time series analysis is a statistical technique that deals with time series data, or trend analysis.

- Time series data means that data is in a series of particular time periods or intervals.

Time series analysis comprises methods for analysing time series data in order to extract meaningful statistics and other characteristics of the data. **Time series forecasting** is the use of a model to predict future values based on previously observed values. While regression analysis is often employed in such a way as to test relationships between one more different time series, this type of analysis is not usually called "time series analysis," which refers in particular to relationships between different points in time within a single series. Interrupted time series analysis is used to detect changes in the evolution of a time series from before to after some intervention which may affect the underlying variable.

Prediction and Forecasting

In statistics, prediction is a part of statistical inference. One particular approach to such inference is known as predictive inference, but the prediction can be undertaken within any of the several approaches to statistical inference. Indeed, one description of statistics is that it provides a means of transferring knowledge about a sample of a population to the whole population, and to other related populations, which is not necessarily the same as prediction over time.

When information is transferred across time, often to specific points in time, the process is known as forecasting.

The data is considered in three types:

Time series data: A set of observations on the values that a variable takes at different times.

Cross-sectional data: Data of one or more variables, collected at the same point in time.

Pooled data: A combination of time series data and cross-sectional data.

Importance of Time Series Analysis

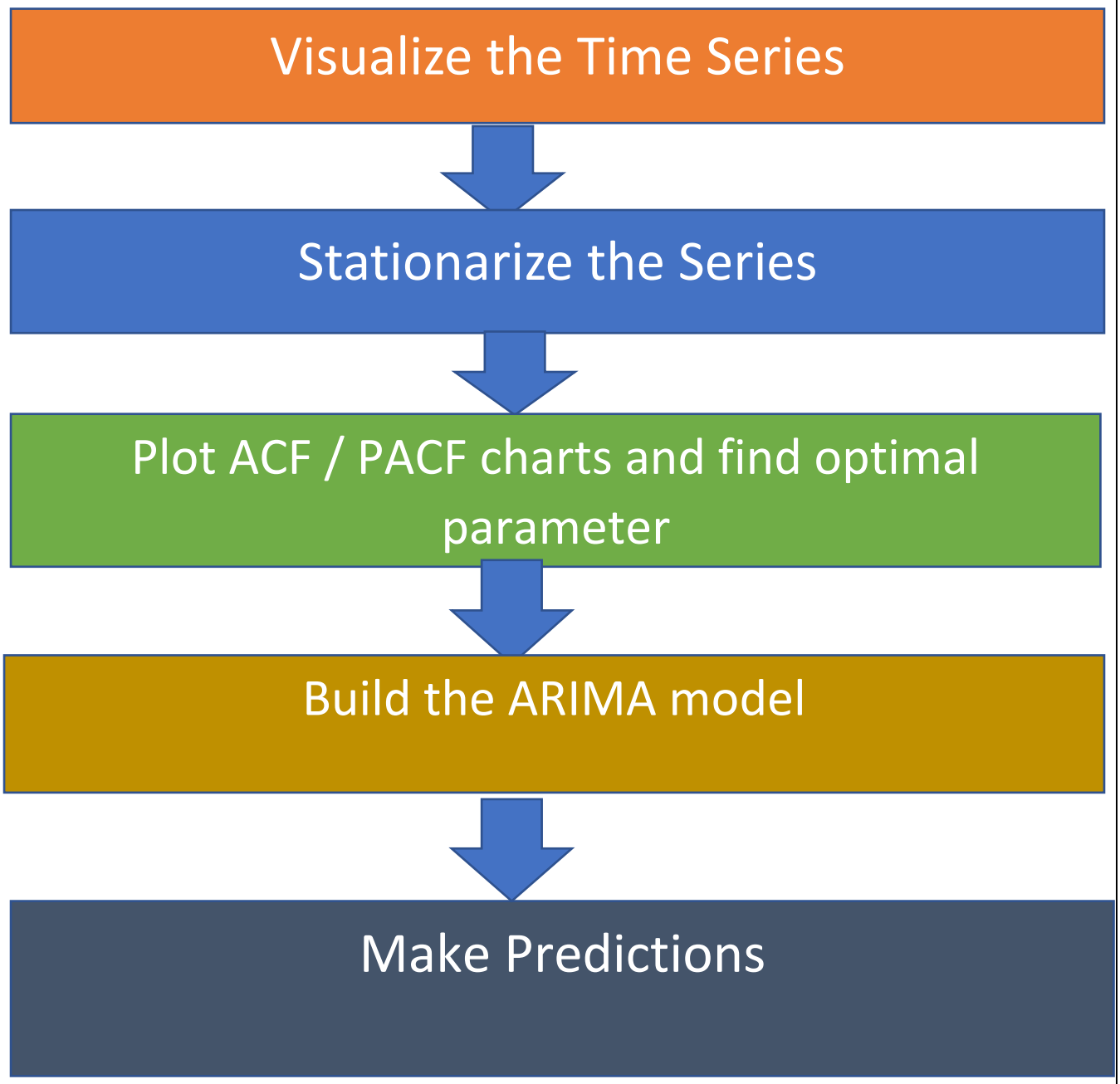
Business forecasting.

Understanding past behaviour.

Comparison of related time series.

Applications

- **Financial Analysis** − It includes sales forecasting, inventory analysis, stock market analysis, price estimation.
- **Weather Analysis** − It includes temperature estimation, climate change, seasonal shift recognition, weather forecasting.
- **Network Data Analysis** − It includes network usage prediction, anomaly or intrusion detection, predictive maintenance.
- **Healthcare Analysis** − It includes census prediction, insurance benefits prediction, patient monitoring.

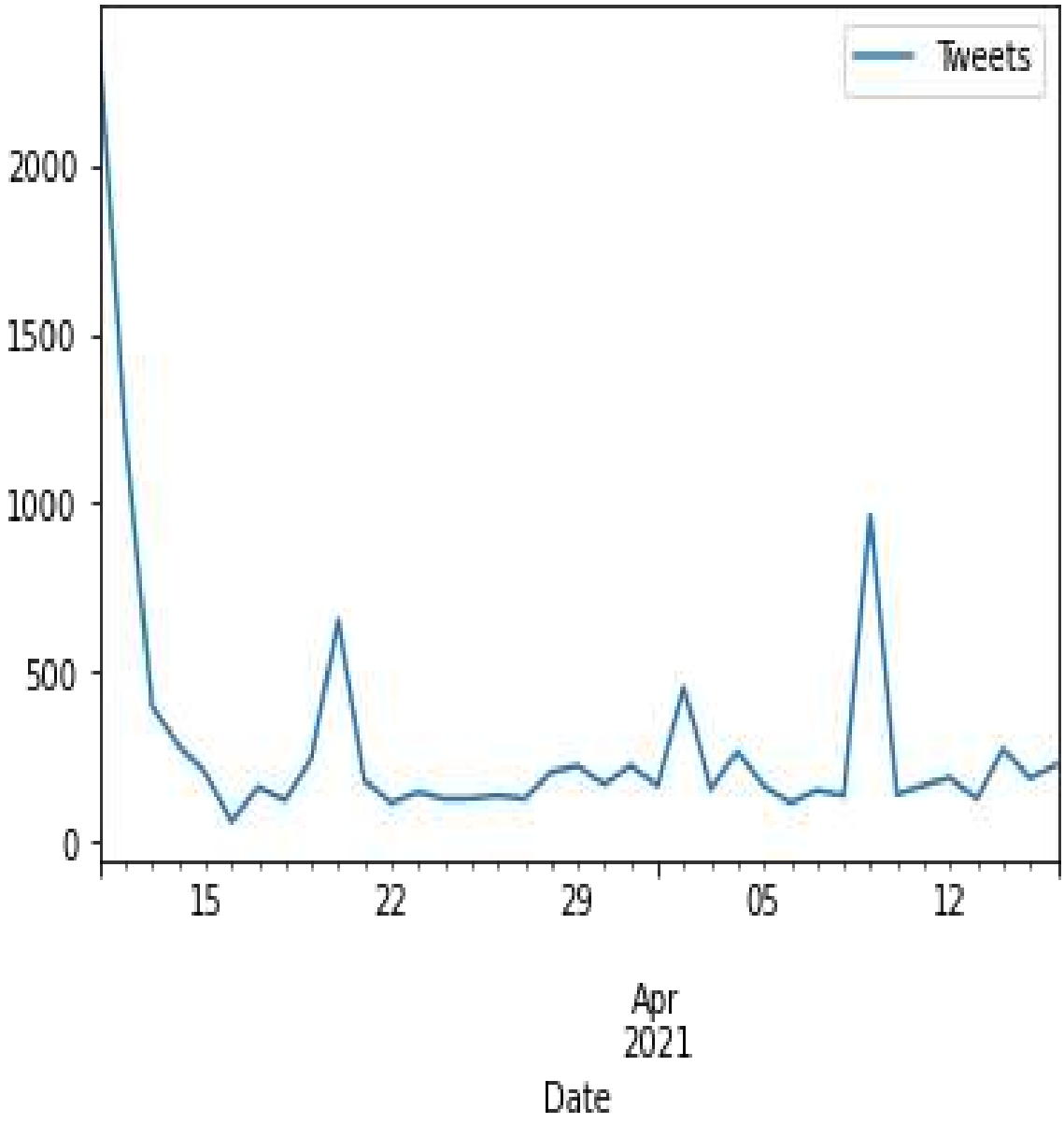

Visualize the Time Series
Stationarize the Series
Plot ACF / PACF charts and find optimal parameter
Build the ARIMA model
Make Predictions

```
df=pd.read_csv("Mahakal_TimeSeriesData.csv")
df.head()
```

	Date	Tweets
0	11-Mar-21	2365
1	12-Mar-21	1211
2	13-Mar-21	400
3	14-Mar-21	285
4	15-Mar-21	203

```
df['Date']=pd.to_datetime(df['Date'])
df.head()
```

	Date	Tweets
0	2021-03-11	2365
1	2021-03-12	1211
2	2021-03-13	400
3	2021-03-14	285
4	2021-03-15	203


Tweets
2000
1500
1000
500
0
15
22
29
05
12
Apr
2021
Date

Stationarity

- It is the process in which mean and variance remains constant over the time.
- For ARMA process Stationarity is required.
- White noise is a simplest example of Stationarity Time Series, white noise Time Series has zero mean, constant variance and zero covariance with lagged Time Series.

Testing for Stationarity

H_0: it is non stationary.

Vs

H_1: it is stationary.

```
adfuller_test(df["Tweets"])
```

```
ADF Test statistics : -8.772784249101026
P-value : 2.494858049465755e-14
#Lags Used : 0
Number of Observation Used : 36
Strong evidence against null hypothesis
```

From the above table P-value corresponds to ADF test is **2.49e-16,** indicate that it is stationary data.

ARIMA

ARIMA stands for Autoregressive integrated moving average model, which is a type of regression analysis that measures the influence of one dependent variable corresponding to changing variable.

An ARIMA model is characterised by three terms:

p, q, d

where,

p is the order of the AR term.

q is the order of the MA term.

d is the number of differencing required to make the Time Series stationary.

Summary for ARMA model

```
model_fit1.summary()
```

ARMA Model Results

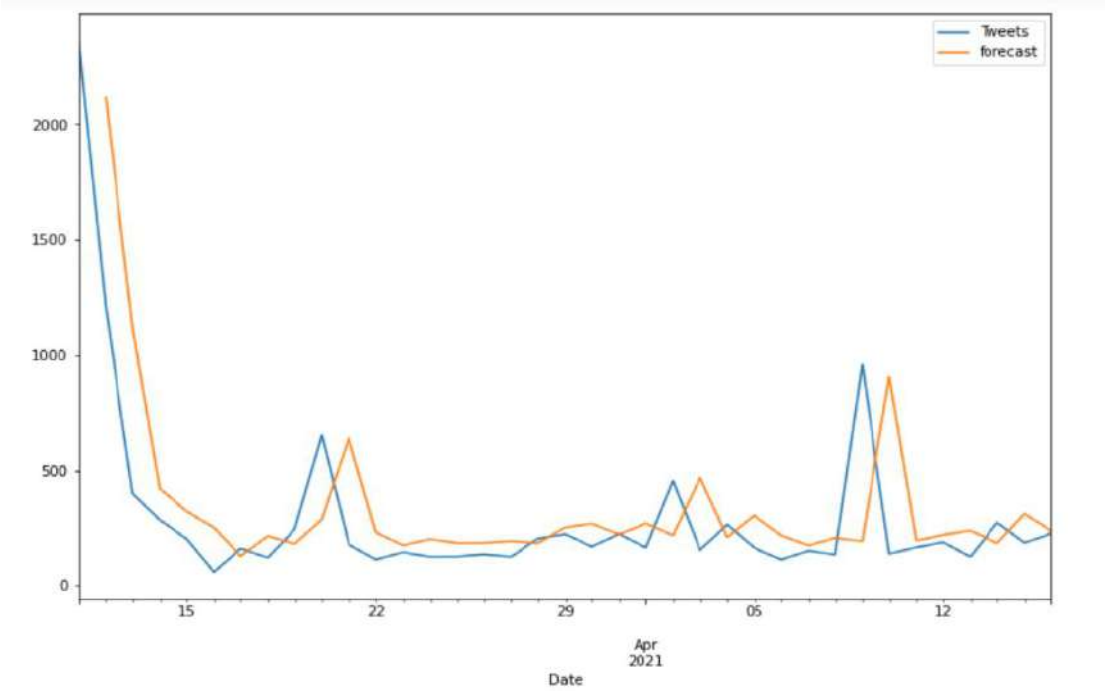
Dep. Variable:	Tweets	**No. Observations:**	37
Model:	ARMA(1, 0)	**Log Likelihood**	-267.954
Method:	css-mle	**S.D. of innovations**	331.826
Date:	Tue, 04 May 2021	**AIC**	541.908
Time:	21:06:00	**BIC**	546.741
Sample:	03-11-2021	**HQIC**	543.612
	- 04-16-2021		

	coef	std err	z	P>|z|	[0.025	0.975]
const	557.5694	407.441	1.368	0.171	-241.000	1356.139
ar.L1.Tweets	0.8630	0.138	6.261	0.000	0.593	1.133

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.1587	+0.0000j	1.1587	0.0000

Prediction plot



Forecasting for next Two days

```
model_fit1.forecast(steps=2)[0]
```

```
array([271.42104794, 310.61995422])
```

Time Series on Politics

Using **ARIMA** model we can predict what will be the sentiment of particular type of hashtags for upcoming 10 days.

We collect data of type "**Politics"** from 18'th March 2021 to 20'th April 2021.

On daily basis we scraped 1000 tweets of each hashtag and then we perform sentiment analysis on them. Now, here we classified no. of positive, negative, neutral tweets from them. We made 3 different ARIMA models for positive, negative and neutral.

Using these models, we forecast no. of positive, negative, neutral tweets for next 10 days. Using forecasted value, we classify them into categories positive, negative and neutral.

```
In [3]: data = pd.read_csv("PoliticsData.csv")
        data.head()
```

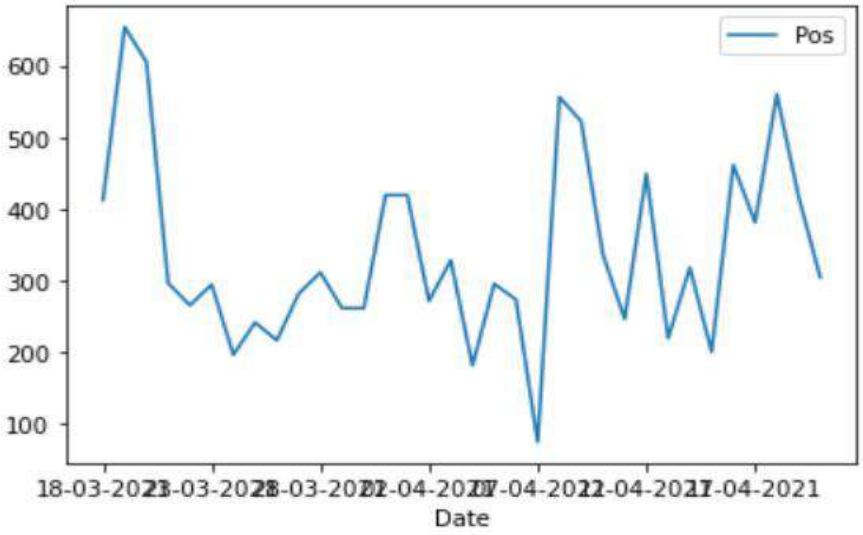
Out[3]:

	Date	Type	Pos	Neg	Neu
0	18-03-2021	Politics	412	152	429
1	19-03-2021	Politics	654	121	225
2	20-03-2021	Politics	605	110	285
3	21-03-2021	Politics	296	505	198
4	22-03-2021	Politics	265	287	447

Models we obtained for positive, negative and neutral tweets.

Positive Time Series:

```
In [5]: data1.plot();
```



Plot for positive Tweets

Using "**auto_arima"** function we got values of p, d, q as (0,0,1). Then we fit the model having summary as

```
In [16]: model_fit1.summary()
```

Out[16]:

ARMA Model Results

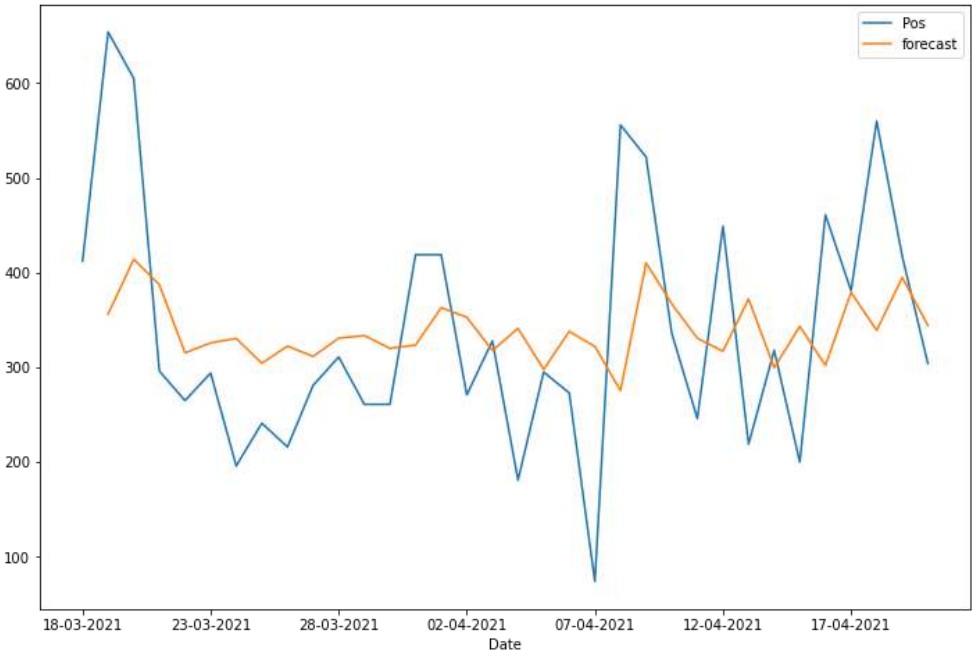
Dep. Variable:	Pos	**No. Observations:**	34
Model:	ARMA(0, 1)	**Log Likelihood**	-212.497
Method:	css-mle	**S.D. of innovations**	125.209
Date:	Wed, 05 May 2021	**AIC**	430.994
Time:	09:10:10	**BIC**	435.573
Sample:	0	**HQIC**	432.555

	coef	std err	z	P>|z|	[0.025	0.975]
const	338.6595	26.790	12.641	0.000	286.151	391.167
ma.L1.Pos	0.2550	0.152	1.674	0.094	-0.044	0.553

Roots

	Real	Imaginary	Modulus	Frequency
MA.1	-3.9219	+0.0000j	3.9219	0.5000

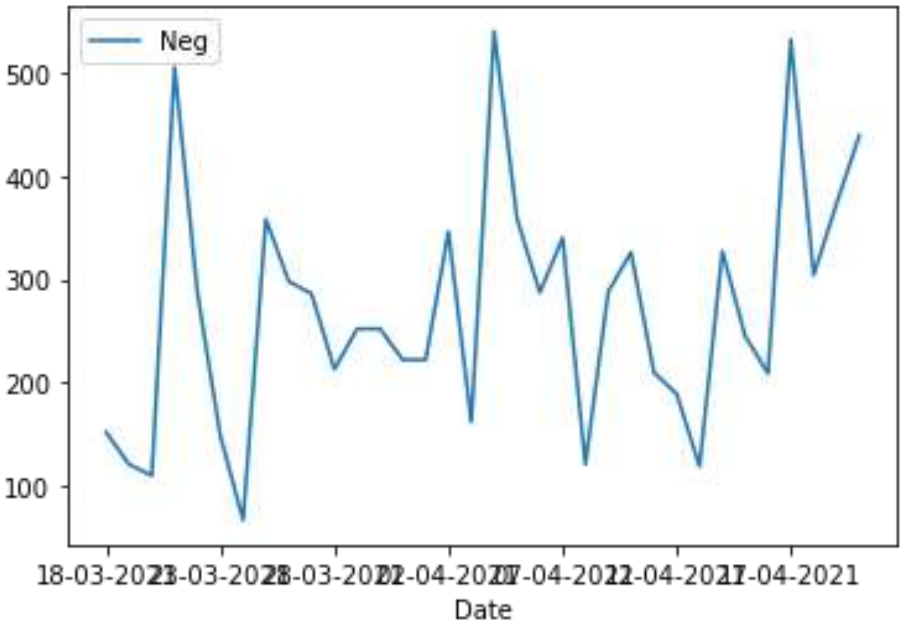
Our model gives forecast plot as



Forecasting for next 10 days

```
In [18]: Positive = model_fit1.forecast(steps=10)[0]

In [19]: Positive

Out[19]: array([328.39021182, 338.65947277, 338.65947277, 338.65947277,
                338.65947277, 338.65947277, 338.65947277, 338.65947277,
                338.65947277, 338.65947277])
```

Negative Time Series:



Plot for Negative Tweets

Using "**auto_arima"** function we got values of p, d, q as (0,1,1). Then we fit the model having summary as

```
In [47]: model_fit2.summary()
```

Out[47]:

ARIMA Model Results

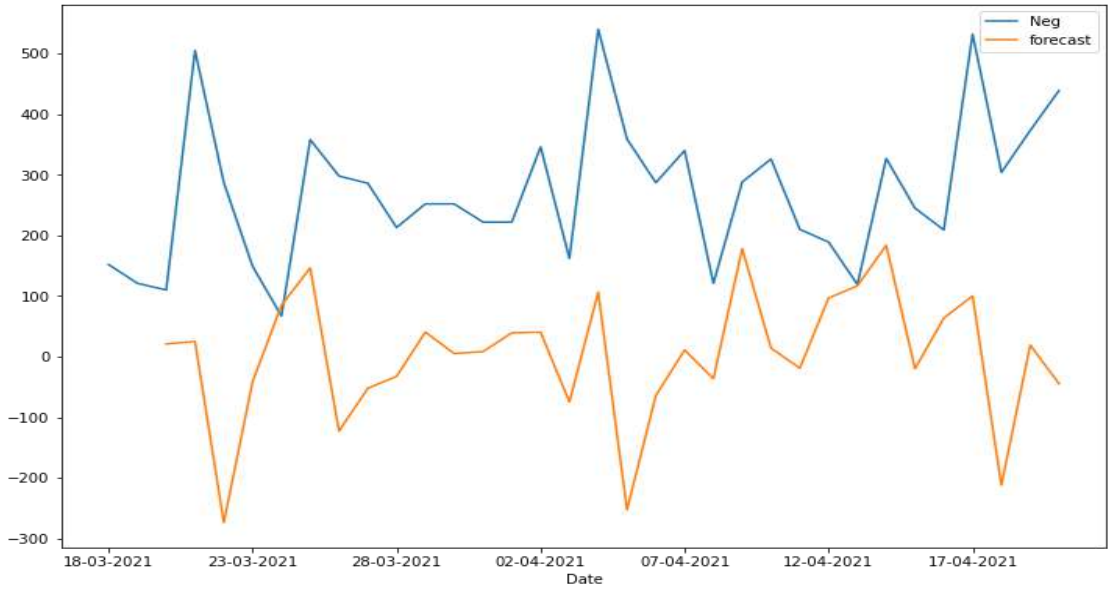
Dep. Variable:	D.Neg	**No. Observations:**	33
Model:	ARIMA(0, 1, 1)	**Log Likelihood**	-204.722
Method:	css-mle	**S.D. of innovations**	113.446
Date:	Sat, 01 May 2021	**AIC**	415.444
Time:	08:57:28	**BIC**	419.933
Sample:	1	**HQIC**	416.954

	coef	std err	z	P>|z|	[0.025	0.975]
const	3.6785	1.983	1.855	0.064	-0.208	7.565
ma.L1.D.Neg	-1.0000	0.089	-11.207	0.000	-1.175	-0.825

Roots

	Real	Imaginary	Modulus	Frequency
MA.1	1.0000	+0.0000j	1.0000	0.0000

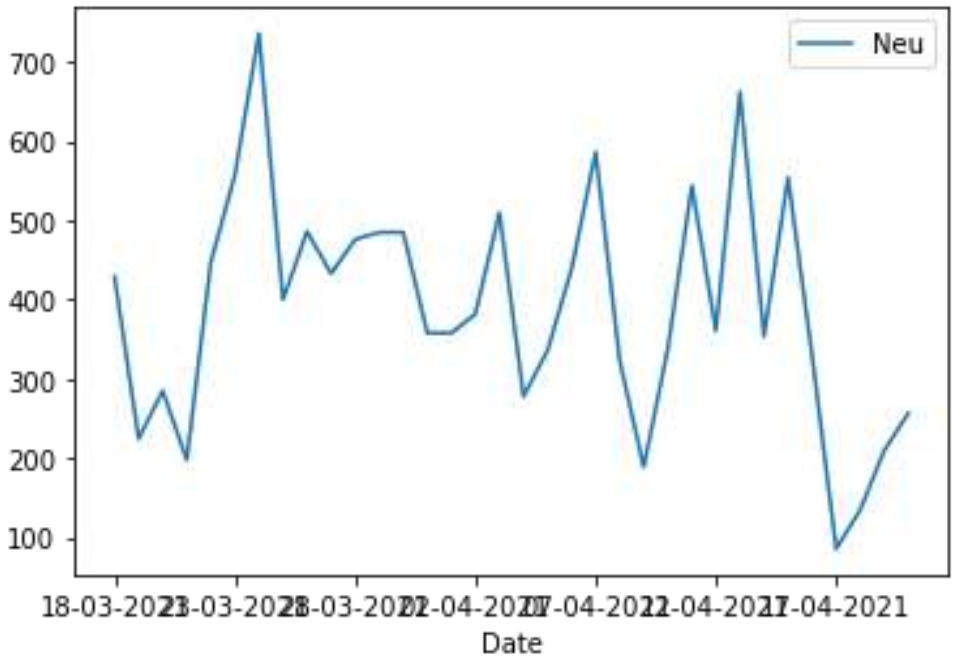
Our model gives forecast plot as



Forecasting for next 10 days

```
In [67]: Negative = model_fit2.forecast(steps=10)[0]

In [68]: Negative

Out[68]: array([332.00156835, 335.68010632, 339.35864429, 343.03718225,
                346.71572022, 350.39425819, 354.07279615, 357.75133412,
                361.42987209, 365.10841006])
```

Neutral Time Series:



Plot for Neutral Tweets

Using "**auto_arima**" function we got values of p, d, q as (0,1,1). Then we fit the model having summary as

```
In [62]: model_fit3.summary()
```

Out[62]:

ARIMA Model Results

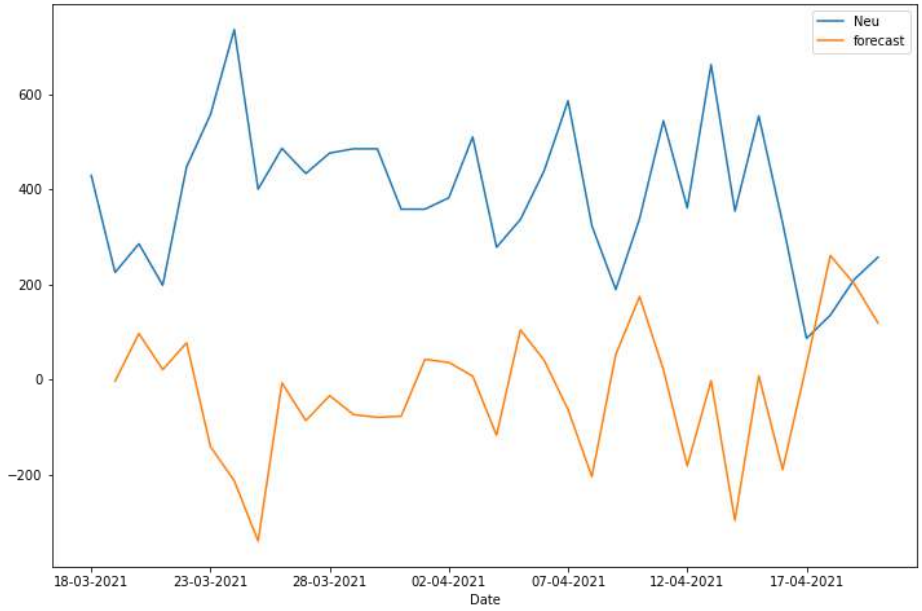
Dep. Variable:	D.Neu	**No. Observations:**	33
Model:	ARIMA(0, 1, 1)	**Log Likelihood**	-212.367
Method:	css-mle	**S.D. of innovations**	143.022
Date:	Sat, 01 May 2021	**AIC**	430.734
Time:	08:58:50	**BIC**	435.223
Sample:	1	**HQIC**	432.245

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.8067	2.500	-1.523	0.128	-8.707	1.093
ma.L1.D.Neu	-1.0000	0.148	-6.745	0.000	-1.291	-0.709

Roots

	Real	Imaginary	Modulus	Frequency
MA.1	1.0000	+0.0000j	1.0000	0.0000

Our model gives forecast plot as



Forecasting for next 10 days

```
In [69]: Neutral = model_fit3.forecast(steps=10)[0]

In [94]: Neutral

Out[94]: array([324.63050974, 320.82378754, 317.01706533, 313.21034313,
                309.40362093, 305.59689872, 301.79017652, 297.98345431,
                294.17673211, 290.37000991])
```

Here we concatenate our forecasted results from 21st April to 30th April, and we predict the sentiment of these days.

```
In [128]: Sentiment
```

Out[128]:

	Positive	Negative	Neutral	Sentiment
0	328.391188	332.001568	324.630510	Negative
1	338.660653	335.680106	320.823787	Positive
2	338.660653	339.358644	317.017065	Negative
3	338.660653	343.037182	313.210343	Negative
4	338.660653	346.715720	309.403621	Negative
5	338.660653	350.394258	305.596899	Negative
6	338.660653	354.072796	301.790177	Negative
7	338.660653	357.751334	297.983454	Negative
8	338.660653	361.429872	294.176732	Negative
9	338.660653	365.108410	290.370010	Negative

Time Series on Cricket

Here also we perform ARIMA model to predict the sentiment same as politics.

```
In [2]: data = pd.read_csv("CricketData.csv")
        data.head()
```

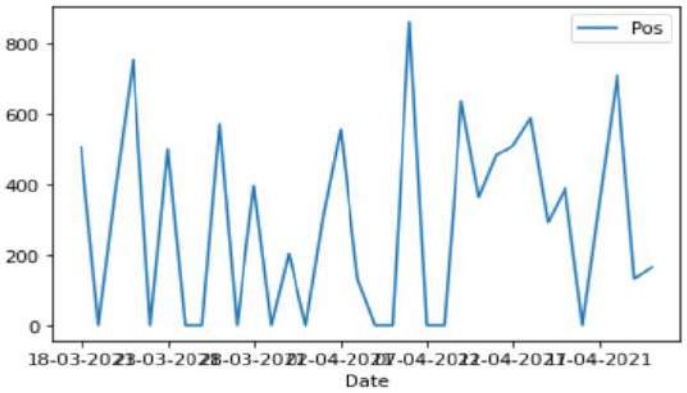
Out[2]:

	Date	Type	Pos	Neg	Neu
0	18-03-2021	Cricket	506	88	406
1	19-03-2021	Cricket	449	88	463
2	20-03-2021	Cricket	392	88	520
3	21-03-2021	Cricket	754	10	236
4	22-03-2021	Cricket	627	30	343

We have a data of type "**Cricket"** from 18'th March 2021 to 20'th April 2021.

Positive Time Series:

```
In [6]: data1.plot();
```



This is the plot for positive tweet with respect to time.

Using "**auto_arima"** function we got values of p, d, q as (4,1,0). Then we fit the model having summary as

```
In [17]: model_fit1.summary()
```

Out[17]:

ARIMA Model Results

Dep. Variable:	D.Pos	**No. Observations:**	33
Model:	ARIMA(4, 1, 0)	**Log Likelihood**	-231.090
Method:	css-mle	**S.D. of innovations**	258.156
Date:	Sat, 01 May 2021	**AIC**	474.180
Time:	10:07:56	**BIC**	483.160
Sample:	1	**HQIC**	477.202

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.0050	11.062	-0.091	0.928	-22.686	20.676
ar.L1.D.Pos	-1.0957	0.159	-6.899	0.000	-1.407	-0.784
ar.L2.D.Pos	-1.0803	0.219	-4.934	0.000	-1.510	-0.651
ar.L3.D.Pos	-0.6967	0.224	-3.107	0.002	-1.136	-0.257
ar.L4.D.Pos	-0.4107	0.167	-2.463	0.014	-0.738	-0.084

Roots

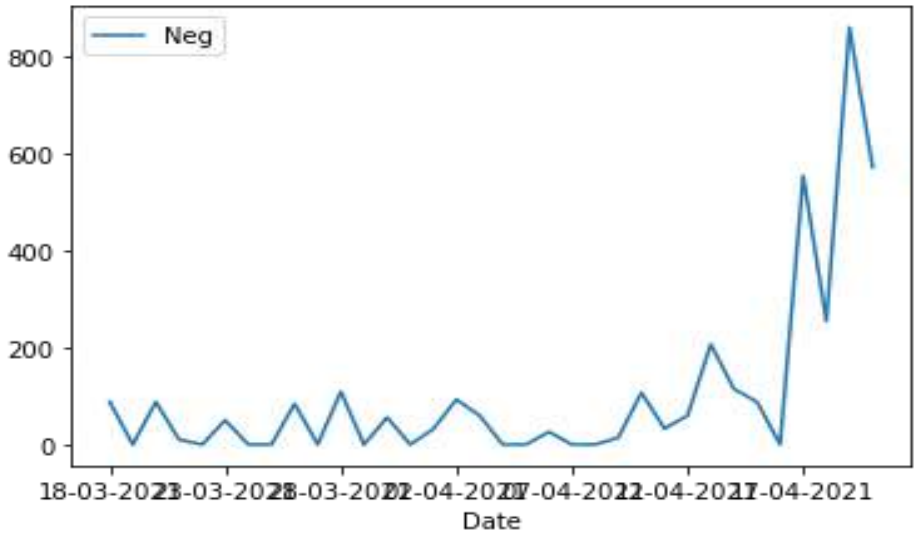
	Real	Imaginary	Modulus	Frequency
AR.1	0.1259	-1.2466j	1.2530	-0.2340
AR.2	0.1259	+1.2466j	1.2530	0.2340
AR.3	-0.9741	-0.7760j	1.2454	-0.3929
AR.4	-0.9741	+0.7760j	1.2454	0.3929

Our model gives forecast plot as

Forecasting for next 10 days

```
In [19]: Positive = model_fit1.forecast(steps=10)[0]

In [20]: Positive

Out[20]: array([356.97828313, 365.2086339 , 358.87925946, 205.31137367,
                291.53411825, 359.68517523, 297.14781657, 290.73969215,
                278.12379846, 310.14620252])
```

Negative Time Series:



We can see here increasing trend in negative tweets with respect to time.

Using "**auto_arima"** function we got values of p, d, q as (4,1,0). Then we fit the model having summary as

Out[35]:

ARIMA Model Results

Dep. Variable:	D2.Neg	**No. Observations:**	32
Model:	ARIMA(4, 2, 0)	**Log Likelihood**	-205.143
Method:	css-mle	**S.D. of innovations**	111.922
Date:	Sat, 01 May 2021	**AIC**	422.286
Time:	10:08:53	**BIC**	431.080
Sample:	2	**HQIC**	425.201

	coef	std err	z	P>|z|	[0.025	0.975]
const	5.7534	nan	nan	nan	nan	nan
ar.L1.D2.Neg	-1.6175	2.4e-05	-6.74e+04	0.000	-1.618	-1.617
ar.L2.D2.Neg	-1.1409	2.33e-05	-4.9e+04	0.000	-1.141	-1.141
ar.L3.D2.Neg	-0.8166	1.36e-05	-6.03e+04	0.000	-0.817	-0.817
ar.L4.D2.Neg	-0.2932	2.73e-06	-1.07e+05	0.000	-0.293	-0.293

Roots

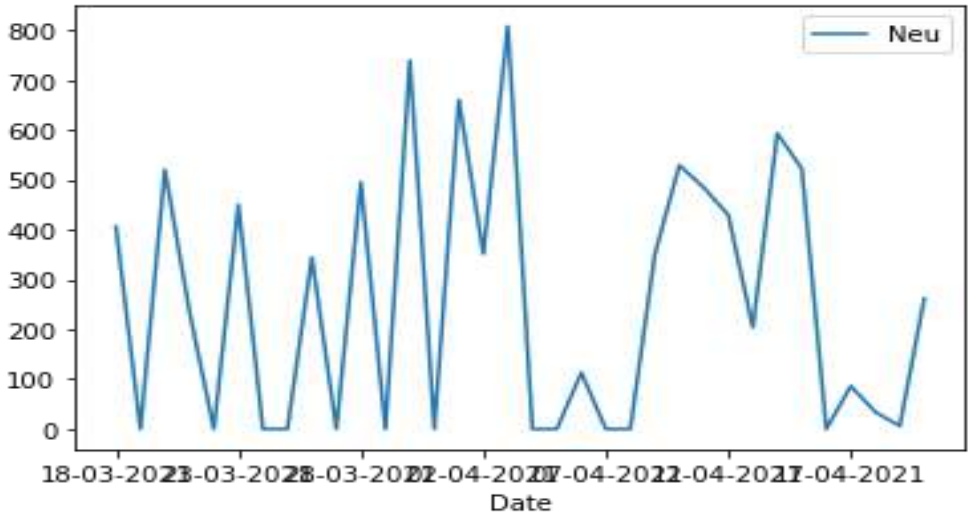
	Real	Imaginary	Modulus	Frequency
AR.1	-0.0346	-1.4094j	1.4098	-0.2539
AR.2	-0.0346	+1.4094j	1.4098	0.2539
AR.3	-1.0000	-0.0000j	1.0000	-0.5000
AR.4	-1.7161	-0.0000j	1.7161	-0.5000

Our model gives forecast plot as

Forecasting for next 10 days

```
In [37]: Negative = model_fit2.forecast(steps=10)[0]

In [38]: Negative

Out[38]: array([1236.15452354,  919.48350922, 1595.11836034, 1296.7685297 ,
                1990.67890472, 1695.80404657, 2400.64050468, 2119.85047861,
                2837.26048786, 2566.13952526])
```

Neutral Time Series:



This is the time series plot for neutral tweets with respect to date.

Using "**auto_arima"** function we got values of p, d, q as (3,2,0). Then we fit the model having summary as

```
In [51]: model_fit3.summary()
```

Out[51]:

ARIMA Model Results

Dep. Variable:	D2.Neu	**No. Observations:**	32
Model:	ARIMA(3, 2, 0)	**Log Likelihood**	-232.296
Method:	css-mle	**S.D. of innovations**	333.212
Date:	Sat, 01 May 2021	**AIC**	474.593
Time:	10:10:10	**BIC**	481.921
Sample:	2	**HQIC**	477.022

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.1946	16.760	0.131	0.896	-30.654	35.043
ar.L1.D2.Neu	-1.3752	0.162	-8.479	0.000	-1.693	-1.057
ar.L2.D2.Neu	-0.9170	0.248	-3.703	0.000	-1.402	-0.432
ar.L3.D2.Neu	-0.3623	0.165	-2.191	0.028	-0.686	-0.038

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-1.2580	-0.0000j	1.2580	-0.5000
AR.2	-0.6367	-1.3375j	1.4813	-0.3207
AR.3	-0.6367	+1.3375j	1.4813	0.3207

Our model gives forecast plot as

Forecasting for next 10 days

```
In [53]: Neutral = model_fit3.forecast(steps=10)[0]

In [54]: Neutral

Out[54]: array([163.44410743, 291.36702935, 338.10722447, 425.26853858,
                457.26260792, 565.48641487, 612.84490712, 702.00686292,
                769.8998621 , 858.77710552])
```

Here we concatenate our forecasted results from 21’st April to 30’th April, and we predict the sentiment of these days.

```
In [4]: Sentiment
```

Out[4]:

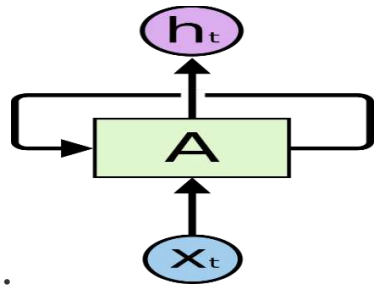
	Positive	Negative	Neutral	Sentiment
0	356.978283	1236.154524	163.444107	Negative
1	365.208634	919.483509	291.367029	Negative
2	358.879260	1595.118360	338.107225	Negative
3	205.311374	1296.768530	425.268539	Negative
4	291.534118	1990.678905	457.262608	Negative
5	359.685175	1695.804047	565.486415	Negative
6	297.147817	2400.640505	612.844907	Negative
7	290.739692	2119.850479	702.006863	Negative
8	278.123798	2837.260488	769.899862	Negative
9	310.146203	2566.139525	858.777105	Negative

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

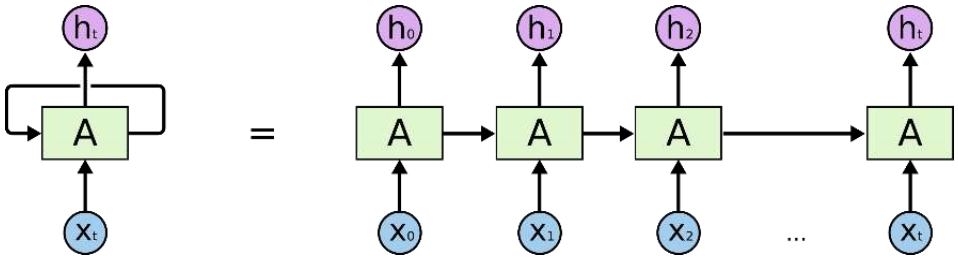
Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



Recurrent Neural Networks have loops.

In the above diagram, a chunk of neural network, A, looks at some input xt and outputs a value ht. A loop allows information to be passed from one step of the network to the next.

These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop:



An unrolled recurrent neural network.

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data.

And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning… The list goes on. I'll leave discussion of the amazing feats one can achieve with RNNs to Andrej Karpathy's excellent blog post, The Unreasonable Effectiveness of Recurrent Neural Networks. But they really are pretty amazing.

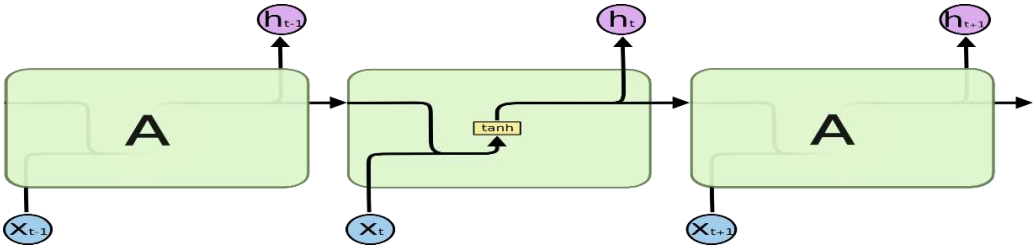
Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much much better than the standard version. Almost all exciting results based on recurrent neural networks are achieved with them. It's these LSTMs that this essay will explore.

LSTM Networks

Long Short-Term Memory networks usually just called "LSTMs" are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used.

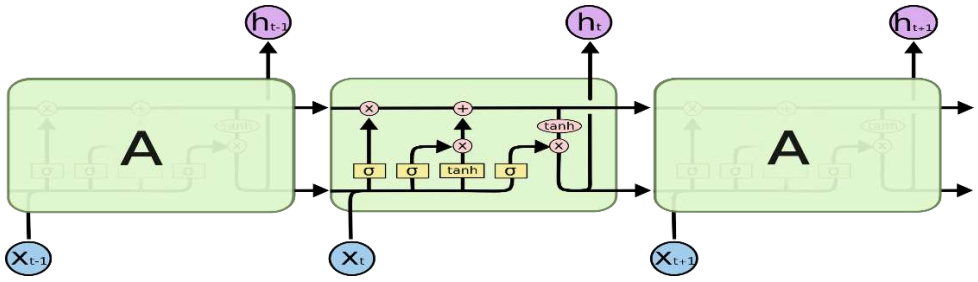
LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

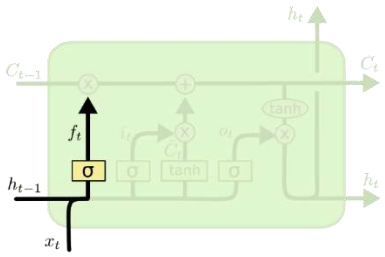


The repeating module in an LSTM contains four interacting layers

Step-by-Step LSTM Walk Through

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at ht−1 and x_t and outputs a number between 0 and 1 for each number in the cell state Ct−1. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

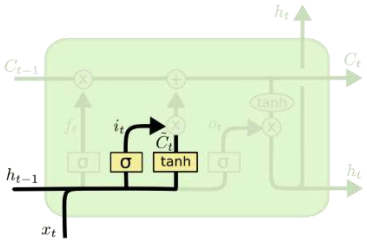
Let's go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $C\tilde{}_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.

In the example of our language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.

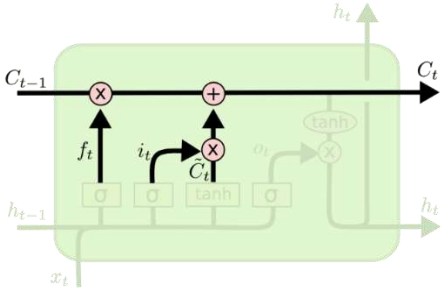


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

It's now time to update the old cell state, Ct−1, into the new cell state Ct. The previous steps already decided what to do, we just need to actually do it.

We multiply the old state by ft forgetting the things we decided to forget earlier. Then we add $i_t * C\tilde{}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

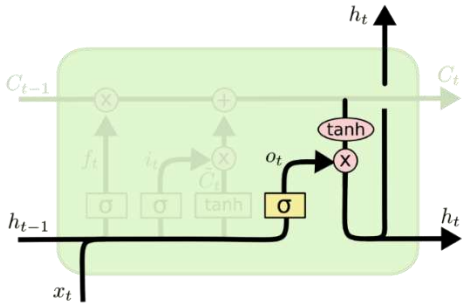
In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM on Politics:

Here, we developed 3 model using LSTM same as it is developed in ARIMA**.**

Positive LSTM:

We Prepare the data for the modelling

```
In [5]: def prepare_data(data,n_step):
            x,y=[],[]
            for i in range(len(data)):
                end_ix = i + n_step
                if end_ix > len(data)-1:
                    break
                seq_x, seq_y = data[i:end_ix], data[end_ix]
                x.append(seq_x)
                y.append(seq_y)
            return np.array(x), np.array(y)
```

```
In [6]: Pos_data = data.Pos
        n_step = 3
        x,y = prepare_data(Pos_data, n_step)
```

```
In [8]: print(x)
```

```
[[412 654 605]
 [654 605 296]
 [605 296 265]
 [296 265 294]
 [265 294 196]
 [294 196 241]
 [196 241 216]
 [241 216 281]
 [216 281 311]]
```

```
In [9]: print(y)
```

```
[296 265 294 196 241 216 281 311 261 261 419 419 271 328 181 295 273  74
 556 522 336 246 449 219 318 200 461 381 560 417 304]
```

Fitted LSTM Model using RELU activation function

```
In [15]: model = Sequential()
         model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(n_step, n_features)))
         model.add(LSTM(50, activation='relu'))
         model.add(Dense(2))
         model.compile(optimizer='adam', loss='mse')
         model.fit(x,y,epochs=300, verbose=2)
```

Prediction for Next 10 days

```
In [18]: Pos_pred

Out[18]: [422.57336,
          198.56592,
          366.07407,
          236.75346,
          360.02014,
          190.46527,
          341.3197,
          282.8282,
          171.38878,
          281.79218]
```

Negative LSTM:

We Prepare the data for the modelling.

```
In [11]: print(x)

[[152 121 110]
 [121 110 505]
 [110 505 287]
 [505 287 149]
 [287 149  67]
 [149  67 358]
 [ 67 358 298]]
```

```
In [12]: print(y)

[505 287 149  67 358 298 286 213 252 252 222 222 346 162 540 359 287 340
 121 288 326 210 189 119 327 245 209 532 304 373 439]
```

Using RELU Activation function developed the model for negative tweets and forecast for Next 10 days.

```
In [31]: Neg_pred

Out[31]: [138.86958,
          389.64886,
          384.5437,
          201.89235,
          247.06026,
          210.27017,
          330.78317,
          160.48041,
          536.4602,
          360.36566]
```

Neutral LSTM:

We Prepare the data for the modelling.

```
In [14]: print(x)

[[429 225 285]
 [225 285 198]
 [285 198 447]
 [198 447 557]
 [447 557 736]
 [557 736 400]
 [736 400 486]
 [400 486 433]]
```

```
In [15]: print(y)

[198 447 557 736 400 486 433 476 485 485 358 358 382 510 278 336 439 586
 323 189 338 544 361 662 354 554 329  86 135 210 257]
```

Using RELU Activation function developed the model for neutral tweets and forecast for Next 10 days.

```
In [46]: Neu_pred = lst_output
         Neu_pred

Out[46]: [395.65732,
          513.3883,
          626.427,
          363.55826,
          340.81174,
          399.29034,
          575.51495,
          281.76904,
          274.46936,
          185.25266]
```

Forecasting for 21’st April to 30’th April

```
In [17]: Sentiment
```

Out[17]:

	Pos_pred	Neg_pred	Neu_pred	Sentiment
0	422.573364	138.869583	395.657318	Positive
1	198.565918	389.648865	513.388306	Neutral
2	366.074066	384.543701	626.427002	Neutral
3	236.753464	201.892349	363.558258	Neutral
4	360.020142	247.060257	340.811737	Positive
5	190.465271	210.270172	399.290344	Neutral
6	341.319702	330.783173	575.514954	Neutral
7	282.828186	160.480408	281.769043	Positive
8	171.388779	536.460205	274.469360	Negative
9	281.792175	360.365662	185.252655	Negative

Actual Sentiment for 21’st April to 30’th April

```
In [19]: Actual_sentiment
```

Out[19]:

	Date	Pos	Neg	Neu	Sentiment
0	21-Apr-21	472	156	372	Positive
1	22-Apr-21	367	302	331	Positive
2	23-Apr-21	230	298	472	Neutral
3	24-Apr-21	384	104	512	Neutral
4	25-Apr-21	467	156	377	Positive
5	26-Apr-21	281	267	452	Neutral
6	27-Apr-21	330	102	568	Neutral
7	28-Apr-21	302	306	392	Neutral
8	29-Apr-21	212	427	361	Negative
9	30-Apr-21	522	256	222	Positive

LSTM for Cricket

Here, we developed 3 model using LSTM same as it is developed in ARIMA.

Positive LSTM:

We Prepare the data for the modelling as show in Politics.

```
In [6]: print(x)

[[506 449 392]
 [449 392 754]
 [392 754 627]
 [754 627 500]
 [627 500 535]
 [500 535 535]
 [535 535 571]
```

```
In [7]: print(y)

[754 627 500 535 535 571 483 396 300 204 170 308 556 132 496 496 861 748
 748 636 363 484 510 588 292 389 374 659 710 578 465]
```

Using RELU Activation function developed the model for positive tweets and forecast for Next 10 days.

```
In [94]: Pos_pred = lst_output
         Pos_pred

Out[94]: [456.6195,
          563.30646,
          436.01117,
          407.79837,
          483.24246,
          393.73618,
          610.54614,
          558.81537,
          520.19543,
          473.1392]
```

Negative LSTM:

We Prepare the data for the modelling.

```
In [9]: print(x)

[[ 88  88  88  10  30]
 [ 88  88  10  30  50]
 [ 88  10  30  50  67]
 [ 10  30  50  67  67]
 [ 30  50  67  67  84]
 [ 50  67  67  84  96]
 [ 67  67  84  96 109]
 [ 67  84  96 109  82]]
```

```
In [10]: print(y)

[ 50  67  67  84  96 109  82  56  29  32  93  60  43  43  26  20  20  14
 107  33  60 207 115  88 321  55 105  62  73]
```

Using RELU Activation function developed the model for negative tweets and forecast for Next 10 days.

```
In [106]: Neg_pred = lst_output
          Neg_pred

Out[106]: [290.81378,
           64.176605,
           45.684715,
           508.02658,
           129.1446,
           180.17352,
           838.10846,
           411.70114,
           772.20514,
           1397.2137]
```

Neutral LSTM:

We Prepare the data for the modelling.

```
In [12]: print(x)

[[406 463 520]
 [463 520 236]
 [520 236 343]
 [236 343 450]
 [343 450 397]
 [450 397 397]
 [397 397 344]
 [397 344 419]]
```

```
In [13]: print(y)

[236 343 450 397 397 344 419 495 617 739 466 660 351 808 460 460 113 231
 231 350 529 484 429 204 593 522 304 286 184 360 462]
```

Using RELU Activation function developed the model for neutral tweets and forecast for Next 10 days.

```
In [117]: Neu_pred = lst_output
          Neu_pred

Out[117]: [417.01584,
           358.232,
           402.14514,
           464.46533,
           387.69647,
           344.64227,
           360.93027,
           456.55478,
           595.30206,
           654.3228]
```

Forecasting for 21'st April to 30'th April

```
In [15]: Sentiment
```

Out[15]:

	Pos_pred	Neg_pred	Neu_pred	Sentiment
0	456.619507	290.813782	417.015839	Positive
1	563.306457	64.176605	358.231995	Positive
2	436.011169	45.684715	402.145142	Positive
3	407.798370	508.026581	464.465332	Negative
4	483.242462	129.144608	387.696472	Positive
5	393.736176	180.173523	344.642273	Positive
6	610.546143	838.108459	360.930267	Negative
7	558.815369	411.701141	456.554779	Positive
8	520.195435	772.205139	595.302063	Negative
9	473.139191	1397.213745	654.322815	Negative

Actual Sentiment for 21'st April to 30'th April

```
In [17]: Actual_sentiment
```

Out[17]:

	date	Pos	Neg	Neu	Sentiment
0	21-Apr-21	589	103	308	Positive
1	22-Apr-21	512	89	399	Positive
2	23-Apr-21	659	58	283	Positive
3	24-Apr-21	240	157	603	Neutral
4	25-Apr-21	702	198	100	Positive
5	26-Apr-21	489	212	299	Positive
6	27-Apr-21	270	207	523	Neutral
7	28-Apr-21	499	107	394	Positive
8	29-Apr-21	427	137	436	Neutral
9	30-Apr-21	324	278	398	Neutral

Conclusion:

- We perform Time series after analysis we found that prediction of time series is not satisfactory. So further we perform LSTM then we found that LSTM is more accurate as compare to time series analysis.
- We Perform LSTM on Politics type, then out of 10 days sentiments, 7 days sentiments are more precise.
- We Perform LSTM on Cricket type, then out of 10 days sentiments, 6 days sentiments are more precise.

References

Scraping Tweets using Twitter API's

https://medium.com/analytics-vidhya/scraping-tweets-using-twitter-apis-d497fd97d5f6

https://developer.twitter.com/en/docs/apps/app-management

Sentiment Analysis on a Twitter Account

https://betterprogramming.pub/twitter-sentiment-analysis-15d8892c0082

https://www.quora.com/How-can-I-search-the-first-tweet-that-started-a-specific-Hashtag

Time Series Forecasting in Python

https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

LSTM Networks

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Appendix

We uploaded all python codes on Github. Here we provided link to Github repository.

https://github.com/Jitesh619/Twitter-/blob/main/Time_Series.pdf

https://github.com/Jitesh619/Twitter-/blob/main/Politics_Time_Series.pdf

https://github.com/Jitesh619/Twitter-/blob/main/Politics%20LSTM.pdf

https://github.com/Jitesh619/Twitter-/blob/main/1-Cricket%20LSTM.pdf

https://github.com/Jitesh619/Twitter-/blob/main/Cricket_Time_Series.pdf

https://github.com/Jitesh619/Twitter-.git

Project Learning Remarks :

- Discussion on particular subject.
- Got new concepts and ideas.
- We learnt different statistical software as well as various statistical techniques.
- Also, we understood that every day on social sites people use to talk which trend is going on in market.