

Opinion Mining + Sentiment Classification :

For the Top 10 Indian Web Series(Thriller Genre)

Getting The Data

We have Web Scraped the user reviews from different OTT platforms(Amazon Prime,Netflix,ALT Balaji,ZEE5,Disney+Hotstar) for the top 10 Indian Web Series in Thriller Genre, on which our further analysis are done.

In [2]:

```
import pandas as pd #for working with dataframes
```

In [3]:

```
#Reading the webscraped reviews of all the the top 10 webseries of THRILLER genre.  
r1_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r2_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r3_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r4_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r5_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r6_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r7_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r8_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r9_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER REV  
r10_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\THRILLER RE
```

In [4]:

```
#printing the dataframes to see the reviews
r1_df
```

Out[4]:

Unnamed: 0	
0	REVIEWS OF SACRED GAMES (THRILLER)
1	NaN
2	NaN
3	Had read the sprawling world building novel by...
4	Simply brilliant content to binge watch. Stell...
...	...
1607	It is just an Havoc
1608	Beginning of new era in India
1609	Best Netflix series!
1610	Mysterious show
1611	It's worth a watch.

1612 rows × 1 columns

In [5]:

```
r2_df
```

Out[5]:

Unnamed: 0	
0	REVIEWS OF DELHI CRIME (THRILLER)
1	NaN
2	The incident that shook the country to its cor...
3	Probably the best Indian crime series in Netfl...
4	#Delhi_Crime - Finally Done Watch 7-EP Total :...
...	...
554	The Delhi Crime Web Series 18+ or under 18?
555	Well written and acted showsad about the...
556	Worth for Watching.....
557	Raw.. Rough.. Real!
558	Delhi Crime Branch master Minecraft branch

559 rows × 1 columns

In [6]:

r3_df

Out[6]:

Unnamed: 0

0	REVIEWS OF MIRZAPUR (THRILLER)
1	NaN
2	Excellent Acting, dragging storyline and defam...
3	The series was excellent, everyone had a uniqu...
4	MIRZAPUR the mighty show which actually increa...
...	...
1651	When will Season 2 be released in other langua...
1652	Cenema is the more better Tv
1653	Mirzapur is shows the power of purwanchal.
1654	Superbb web series and all actor perfect.
1655	Just... Influenced me every single character..

1656 rows × 1 columns

In [7]:

r4_df

Out[7]:

Unnamed: 0

0	REVIEWS OF BREATHE (THRILLER)
1	NaN
2	After 2017's Inside Edge, about the politics a...
3	It was a great Season 1 and can't wait for the...
4	Breathe is a class on its own. I hv started co...
...	...
716	Nice web series but..... ,.
717	R Madhava awesome negative role
718	Series of the series 🍷🍷🍷🍷👍👍👍👍👍👍👍👍
719	Penomminal acting done by Abhishek, amit 🍷
720	Another level of web series.....

721 rows × 1 columns

In [8]:

r5_df

Out[8]:


Unnamed: 0	
0	REVIEWS OF APAHARAN (THRILLER)
1	Absolutely worth watching it pls see this web ...
2	Outstanding series by Balaji telefilms... stor...
3	rocking thrilling entertaining superb series, ...
4	Best thriller series I've seen so far, too man...
...	...
412	Mind blowing jobApharan team ...great w...
413	It was really an 70s journey and you should wa...
414	Wonderful dirsction and powerpack acting by ar...
415	Such a great, excellent story, plot, and scrip...
416	Ab filme chhod kar ye series dekhne me jyada m...

417 rows × 1 columns

In [9]:

r6_df

Out[9]:

Unnamed: 0	
0	REVIEWS OF CRIMINAL JUSTICE (THRILLER)
1	NaN
2	Well this was the first Crime Thriller Web Ser...
3	Amazing show. 2nd season was even better than ...
4	What a fabulous and thrilling series it is. Fr...
...	...
541	Ditto copy of english tv series of the night of
542	haven't watched yet
543	Very Good Movie  ☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
544	Copy of "The whole truth" movie.
545	Nice movie thanks.....

546 rows × 1 columns

In [10]:

r7_df

Out[10]:

Unnamed: 0

0	REVIEWS OF INSIDE EDGE (THRILLER)
1	NaN
2	Amazing... \n\nKaran Anshuman has done a splen...
3	Deserves 10 Stars!\n\nInside Edge keeps you ri...
4	This is an another best Indian web series I ha...
...	...
328	When the 3rd Season will come?
329	Vivek oberoï has still something left in him..!!
330	I loved all the episodes waiting for further.....
331	I am just waiting for season 3.
332	Maybe the best by India

333 rows × 1 columns

In [11]:

r8_df

Out[11]:

Unnamed: 0

0	REVIEWS OF PAATAL LOK (THRILLER)
1	NaN
2	Gripping from start to end. Simply wow!\n\nMin...
3	TLDR:Not for the weak hearted...not even for t...
4	Totally worth the hype!!\n\nPaatal lok comes s...
...	...
1957	awosomeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee...
1958	Hinduphobic ideology is represented.
1959	More cuss words than dialogues 😊
1960	Just 3 Star ☆☆☆\n.....
1961	Intresting series to watch thriller.

1962 rows × 1 columns

In [12]:

r9_df

Out[12]:

Unnamed: 0

0	REVIEWS OF THE FINAL CALL (THRILLER)
1	NaN
2	Series deserve praise for presenting spiritual...
3	A moving tale about life and death!\n\nThe Fin...
4	Simply Awesome. It is a modern depiction of ag...
...	...
303	awesome start of the story but unfortunately p...
304	Bessssttt series ever in Zee5
305	I just loved it man!! 🐶🐶🐶🐶
306	What happened to episode#5??
307	Ek number show \nZee5 release another eposides...

308 rows × 1 columns

In [13]:

r10_df

Out[13]:

Unnamed: 0

0	REVIEWS OF HOSTAGES (THRILLER)
1	NaN
2	Hostages 2: Bigger & Better!\n\nHostages 2 is ...
3	Hostages season 2 one of the well made Indian ...
4	Hostages season 2 : GRIPPING AND WORTH WATCHIN...
...	...
329	Good one,but was it necessary to use the dirty...
330	Wating for s3season 2 better then s1 \n...
331	Worth watching but they could have done better.
332	Series is good but too many loop holes...
333	Very Intresting and Full of Twist show...

334 rows × 1 columns

In [14]:

```
#combining all the review dataframes into one dataframe
combined_df = pd.concat([r1_df, r2_df,r3_df,r4_df,r5_df,r6_df,r7_df,r8_df,r9_df,r10_df], ig
```

In [15]:

```
combined_df
```

Out[15]:

Unnamed: 0	
0	REVIEWS OF SACRED GAMES (THRILLER)
1	NaN
2	NaN
3	Had read the sprawling world building novel by...
4	Simply brilliant content to binge watch. Stell...
...	...
8443	Good one,but was it necessary to use the dirty...
8444	Wating for s3seasion 2 better then s1 \n...
8445	Worth watching but they could have done better.
8446	Series is good but too many loop holes...
8447	Very Intresting and Full of Twist show...

8448 rows × 1 columns

In [16]:

```
#naming the columns
combined_df.columns=['transcript']
```

In [17]:

```
# Let's take a look at the updated df
combined_df
```

Out[17]:

	transcript
0	REVIEWS OF SACRED GAMES (THRILLER)
1	NaN
2	NaN
3	Had read the sprawling world building novel by...
4	Simply brilliant content to binge watch. Stell...
...	...
8443	Good one,but was it necessary to use the dirty...
8444	Wating for s3season 2 better then s1 \n...
8445	Worth watching but they could have done better.
8446	Series is good but too many loop holes...
8447	Very Intresting and Full of Twist show...

8448 rows × 1 columns

In [25]:

```
combined_df.sample(10)
```

Out[25]:

	transcript
7652	Highly abusive, created by a sick mind,
664	Awsome must watch only those who like story an...
4437	One of the best series..fantastic
1604	To download
6129	Indian webseries on Netflix and primevideo see...
4122	This series is fantastic and fabulous 🥰🥰🥰🥰🥰🥰🥰❤...
7875	Crappy story, pathetic acting, poor direction ...
3269	Those who feel this as unrealistic are either ...
2667	Excellent web series.... Totally full of all t...
2757	Mirzapur is the true definition of what pankaj...

Cleaning The Data

When dealing with numerical data, data cleaning often involves removing null values and duplicate data, dealing with outliers, etc. With text data, there are some common data cleaning techniques, which are also known as text pre-processing techniques.

With text data, this cleaning process can go on forever. There's always an exception to every cleaning step. So, we're going to follow the MVP (minimum viable product) approach - start simple and iterate. Here are a bunch of things you can do to clean your data. We're going to execute just the common cleaning steps here and the rest can be done at a later point to improve our results.

Common data cleaning steps on all text:

- Make text all lower case
- Remove punctuation
- Remove numerical values
- Remove common non-sensical text (\n-new lines,\t-whitespaces etc)
- Tokenize text
- Remove stop words

More data cleaning steps after tokenization:

- Stemming / lemmatization
- Parts of speech tagging
- Create bi-grams or tri-grams
- Deal with typos
- And more...

In [19]:

```
# Applying a first round of text cleaning techniques
import re
import string

def clean_text_round1(text):
    '''Make text lowercase, remove text in square brackets, remove punctuation and remove w

    text = str(text)
    text = text.lower()

    text = re.sub('\[.*?\]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```


In [22]:

```
# Let's take a look at the updated text
data_clean_df = pd.DataFrame(data_clean_df.transcript.apply(clean_text_round2))
data_clean_df
```

Out[22]:

	transcript
0	reviews of sacred games thriller
1	nan
2	nan
3	had read the sprawling world building novel by...
4	simply brilliant content to binge watch stella...
...	...
8443	good onebut was it necessary to use the dirty ...
8444	wating for seasion better then 👍👍👍👍👍❤️
8445	worth watching but they could have done better
8446	series is good but too many loop holes
8447	very intresting and full of twist show

8448 rows × 1 columns

In [24]:

```
# Applying a third round of cleaning

import re
import string

text_translator = str.maketrans({ord(c): " " for c in string.punctuation})
def clean_text_round3(text, remove_punctuation_all=False):
    if not text:
        return ''
    try:
        text = text.replace(chr(160), " ")
        text = ''.join([i if ord(i) < 128 else ' ' for i in text])
    except Exception as e:
        try:
            text = text.encode('utf-8')
            text = text.decode('utf-8')
        except Exception as e:
            return ""
    try:
        text = text.encode('ascii', 'ignore').decode("utf-8")
        text = text.translate(text_translator)
    except Exception as e:
        return ""
    while ' ' in text:
        text = text.replace(' ', ' ')
    text = text.strip()
    return text
```

In [25]:

```
# Let's take a look at the updated text
data_clean_df= pd.DataFrame(data_clean_df.transcript.apply(clean_text_round3))
```

In [26]:

```
#Updated dataframe after three rounds of data cleaning
data_clean_df
```

Out[26]:

	transcript
0	reviews of sacred games thriller
1	nan
2	nan
3	had read the sprawling world building novel by...
4	simply brilliant content to binge watch stella...
...	...
8443	good onebut was it necessary to use the dirty ...
8444	wating for seasion better then
8445	worth watching but they could have done better
8446	series is good but too many loop holes
8447	very intresting and full of twist show

8448 rows × 1 columns

In [27]:

```
data_clean_df.sample(6)
```

Out[27]:

	transcript
7188	i cant explain i was amazing web series just g...
2481	best web series on amazon prime must watch thi...
1690	a well made serial giving fascinating glimpse ...
2454	this should be banned for its filthy dialogues...
7835	absolutely stunningso many characters so many ...
7503	not a very promising show to watch in these times

NOTE:

This data cleaning aka text pre-processing step could go on for a while, but we are going to stop for now. After going through some analysis techniques, if you see that the results don't make sense or could be improved, you can come back and make more edits such as:

- Mark 'cheering' and 'cheer' as the same word (stemming / lemmatization)
- Combine 'thank you' into one term (bi-grams)
- And a lot more...

Exploratory Data Analysis

Introduction

After the data cleaning step where we put our data into a few standard formats, the next step is to take a look at the data and see if what we're looking at makes sense. Before applying any fancy algorithms, it's always important to explore the data first.

When working with numerical data, some of the exploratory data analysis (EDA) techniques we can use include finding the average of the data set, the distribution of the data, the most common values, etc. The idea is the same when working with text data. We are going to find some more obvious patterns with EDA before identifying the hidden patterns with machines learning (ML) techniques. Let's look at the

- Most common words - find these and create word clouds

Organizing The Data

The output of this notebook will be clean, organized data which can be done in two standard text formats:

1. Corpus - a collection of text
2. Document-Term Matrix - word counts in matrix format

Corpus

The definition of a corpus is a collection of texts, and they are all put together.

In [28]:

```
# Python program to generate WordCloud

# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the combined dataframe file
for val in data_clean_df.transcript:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()
    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                      background_color = 'white',
                      stopwords = stopwords,
                      min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For example, the words like the, he, have etc.

NOTE:

At this point, we could go on and continue with this word clouds. However, by looking at these top words, you can see that some of them have very little meaning and could be added to a stop words list, so let's do just that.

In [29]:

```
#present dictionary of stop words
print(stopwords)
```

```
{'didn't', 'more', 'each', 'shouldn't', 'any', 'aren't', 'http', 'these', 'n
o', 'com', 'it's', 'same', 'in', 'were', 'we'll', 'did', 'we've', 'against',
'however', 'can', 'being', 'wasn't', 'i', 'the', 'yourself', 'a', 'your', 't
o', 'he', 'with', 'then', 'yourselves', 'you'd', 'you've', 'other', 'her
e's', 'doesn't', 'she'd', 'there's', 'above', 'i'd', 'until', 'its', 'coul
d', 'he'd', 'can't', 'weren't', 'also', 'they', 'or', 'up', 'they're', 'yo
u're', 'am', 'me', 'won't', 'he's', 'into', 'during', 'she'll', 'haven't',
'they'd', 'is', 'just', 'since', 'they've', 'about', 'been', 'our', 'we're',
'than', 'k', 'hers', 'that', 'where's', 'www', 'myself', 'would', 'don't',
'he'll', 'had', 'else', 'couldn't', 'too', 'where', 'be', 'those', 'ours',
'here', 'as', 'all', 'you'll', 'such', 'so', 'once', 'therefore', 'after',
'her', 'she's', 'not', 'very', 'what's', 'we', 'because', 'herself', 'down',
'like', 'while', 'nor', 'we'd', 'ever', 'do', 'what', 'get', 'hence', 'ove
r', 'wouldn't', 'yours', 'by', 'most', 'mustn't', 'whom', 'both', 'hasn't',
'let's', 'why', 'you', 'himself', 'an', 'him', 'my', 'how', 'own', 'this',
'she', 'themselves', 'out', 'between', 'off', 'ought', 'that's', 'his', 'o
n', 'when's', 'from', 'why's', 'further', 'are', 'theirs', 'was', 'again',
'under', 'it', 'does', 'otherwise', 'they'll', 'of', 'shan't', 'have', 'bu
t', 'for', 'when', 'should', 'some', 'there', 'shall', 'cannot', 'who's', 'h
aving', 'at', 'them', 'only', 'ourselves', 'if', 'below', 'few', 'and', 'r',
'doing', 'hadn't', 'has', 'itself', 'isn't', 'through', 'their', 'i'm', 'whi
ch', 'i've', 'i'll', 'before', 'how's', 'who'}
```

In [30]:

```
#corpus of our reviews
comment_words
```

Out[30]:

```
'reviews of sacred games thriller nan nan had read the sprawling world bui
lding novel by vikram chandra nearly a year back so was waiting with bated
breath for the tv adaptation also met the author at a literary event and a
sked him about the samechandra said that netflix is an incredibly closeted
organization and despite him being the author he had no idea when it would
be on air to tell you the truth i was wondering how a softspoken individua
l like him had conjured up this macabre story but as writers we conjure up
differen simply brilliant content to binge watch stellar performances phen
omenal scorebeautiful writing it takes you to the core essence of the seri
es very quickthe story is very tightno loose ends at all mind boggling dir
ectionscreen play nice visualssets the flavour of indian elements are port
rayed very well through out seriesfirst episodes outdoor clean visualsmusi
cscreenplay complement with each other travels on very good pacelast episo
des twists and turns in to end ar it is a must watch series and for me it
was the first time i was watching an indian series after i watched mahabha
rat when i was a kid in the the approach to the series seemed raw and grit
tywith all the vagaries of the old gangster filled bombay and all the corr
uption of the current mumbai which got carried over from old times the hon
```

In [31]:

```
# Python program to find the most frequent words from data set
from collections import Counter

# split() returns list of all the words in the string
split_it = comment_words.split(" ")

# Pass the split_it list to instance of Counter class.
Counter = Counter(split_it)

# most_common() produces k frequently encountered
# input values and their respective counts.
most_occur = Counter.most_common()

print(most_occur)
```

```
[('the', 9214), ('and', 6022), ('of', 5098), ('is', 4331), ('a', 4247),
('series', 4117), ('to', 3886), ('it', 3148), ('i', 2977), ('in', 2767),
('this', 2485), ('for', 1932), ('watch', 1665), ('best', 1517), ('was', 1498),
('story', 1498), ('good', 1360), ('all', 1332), ('but', 1326), ('you', 1320),
('one', 1276), ('web', 1276), ('very', 1270), ('acting', 1252),
('season', 1230), ('with', 1224), ('its', 1219), ('are', 1196), ('have', 1140),
('that', 1132), ('by', 1126), ('show', 1119), ('not', 1056), ('as', 923),
('just', 904), ('on', 858), ('great', 845), ('so', 808), ('like', 795),
('amazing', 760), ('must', 755), ('indian', 724), ('has', 692), ('be', 677),
('well', 657), ('from', 636), ('awesome', 621), ('an', 583), ('watching', 576),
('will', 570), ('what', 545), ('time', 518), ('really', 515), ('which', 501),
('more', 500), ('can', 487), ('at', 470), ('every', 470), ('thriller', 469),
('actors', 464), ('ever', 456), ('my', 455), ('they', 455), ('watched', 449),
('much', 445), ('crime', 437), ('characters', 429), ('episode', 426),
('if', 419), ('no', 418), ('there', 418), ('such', 408), ('india', 407),
('too', 407), ('about', 404), ('waiting', 403), ('his', 401), ('after', 398),
('also', 397), ('direction', 388), ('loved', 385), ('some', 384), ('superb', 380),
('character', 379), ('excellent', 378), ('how', 376), ('dont', 374),
('brilliant', 369), ('their', 364), ('we', 362), ('...')]

```

In [32]:

```
# Excluding few words from the list
# Look at the most common top words --> add them to the stop word list

add_stop_words = [word for word, count in Counter.most_common() if count > 1666]
add_stop_words
```

Out[32]:

```
['the', 'and', 'of', 'is', 'a', 'series', 'to', 'it', 'i', 'in', 'this', 'for']
```

In [33]:

```
#adding more stopwords for better analysis
from sklearn.feature_extraction import text
additional_stop_words = text.ENGLISH_STOP_WORDS
print (additional_stop_words)
```

```
frozenset({'more', 'each', 'often', 'nowhere', 'same', 'in', 'nine', 'beyon
d', 'i', 'yourself', 'among', 'with', 'cant', 'eleven', 'other', 'above', 'w
hereas', 'alone', 'or', 'along', 'am', 'become', 'into', 'bill', 'everythin
g', 'three', 'since', 'formerly', 'name', 'myself', 'would', 'thereby', 'tw
o', 'too', 'seems', 'done', 'fifteen', 'seemed', 'her', 'eg', 'do', 'take',
'besides', 'found', 'yours', 'fire', 'whom', 're', 'why', 'an', 'him', 'pe
r', 'off', 'whereby', 'on', 'elsewhere', 'wherever', 'it', 'hereupon', 'ont
o', 'anyhow', 'perhaps', 'when', 'amongst', 'twenty', 'and', 'behind', 'giv
e', 'already', 'still', 'made', 'always', 'thru', 'first', 'before', 'un',
'may', 'please', 'somehow', 'can', 'seeming', 'thin', 'another', 'con', 'thi
rd', 'sincere', 'afterwards', 'nothing', 'becoming', 'least', 'together', 'a
mount', 'could', 'whence', 'they', 'around', 'toward', 'is', 'fifty', 'fil
l', 'whole', 'will', 'those', 'here', 'everywhere', 'such', 'once', 'therefo
re', 'inc', 'after', 'etc', 'five', 'we', 'because', 'down', 'ltd', 'never',
'call', 'get', 'across', 'even', 'wherein', 'well', 'sometimes', 'how', 'any
way', 'out', 'between', 'either', 'beside', 'much', 'are', 'was', 'again',
'someone', 'us', 'of', 'but', 'anywhere', 'there', 'somewhere', 'at', 'mostl
y', 'almost', 'if', 'go', 'has', 'itself', 'many', 'bottom', 'their', 'whic
h', 'anyone', 'due', 'these', 'no', 'now', 'were', 'however', 'upon', 'withi
n', 'the', 'might', 'thereafter', 'a', 'whenever', 'your', 'yourselves', 'cr
y', 'top', 'until', 'its', 'hereafter', 'interest', 'next', 'me', 'indeed',
'our', 'six', 'than', 'that', 'couldnt', 'whereupon', 'whether', 'every', 'h
ad', 'ours', 'as', 'all', 'so', 'moreover', 'ten', 'very', 'whoever', 'whil
e', 'ever', 'less', 'others', 'serious', 'thick', 'thence', 'by', 'show', 'h
imself', 'herein', 'my', 'anything', 'empty', 'themselves', 'his', 'detail',
'from', 'sixty', 'towards', 'though', 'under', 'hundred', 'have', 'describ
e', 'mill', 'ourselves', 'amoungst', 'beforehand', 'see', 'whereafter', 'exc
ept', 'sometime', 'whose', 'side', 'ie', 'none', 'part', 'de', 'few', 'who',
'meanwhile', 'any', 'without', 'system', 'hereby', 'against', 'enough', 'bei
ng', 'everyone', 'yet', 'full', 'he', 'seem', 'then', 'find', 'forty', 'hasn
t', 'namely', 'nobody', 'co', 'also', 'up', 'nevertheless', 'during', 'via',
'about', 'been', 'hers', 'must', 'back', 'something', 'throughout', 'else',
'where', 'be', 'front', 'thus', 'not', 'herself', 'nor', 'whatever', 'what',
'therein', 'whither', 'hence', 'rather', 'over', 'most', 'both', 'four', 'yo
u', 'twelve', 'latter', 'own', 'this', 'mine', 'she', 'became', 'thereupon',
'further', 'although', 'eight', 'several', 'otherwise', 'for', 'latterly',
'should', 'some', 'keep', 'cannot', 'noone', 'them', 'becomes', 'only', 'on
e', 'put', 'below', 'neither', 'through', 'last', 'move', 'to', 'former'})
```

In [34]:

```
# Python program to generate WordCloud

# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

comment_words = ''

# Add new stop words

selected_stop_words=['show','season','one','season','watch','story','bajpayee','webserie','
stopwords = list(additional_stop_words) + selected_stop_words + add_stop_words + list(STOPW

# iterate through the file
for val in data_clean_df.transcript:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()
    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title('Sentiment WorldCloud\n',fontsize=35)
plt.show()
```


In [35]:

```
#all the stopwords that were used
print (stopwords)
```

```
['more', 'each', 'often', 'nowhere', 'same', 'in', 'nine', 'beyond', 'i', 'y', 'ourself', 'among', 'with', 'cant', 'eleven', 'other', 'above', 'whereas', 'a', 'lone', 'or', 'along', 'am', 'become', 'into', 'bill', 'everything', 'three', 'since', 'formerly', 'name', 'myself', 'would', 'thereby', 'two', 'too', 'se', 'ems', 'done', 'fifteen', 'seemed', 'her', 'eg', 'do', 'take', 'besides', 'fo', 'und', 'yours', 'fire', 'whom', 're', 'why', 'an', 'him', 'per', 'off', 'wher', 'eby', 'on', 'elsewhere', 'wherever', 'it', 'hereupon', 'onto', 'anyhow', 'pe', 'rhaps', 'when', 'amongst', 'twenty', 'and', 'behind', 'give', 'already', 'st', 'ill', 'made', 'always', 'thru', 'first', 'before', 'un', 'may', 'please', 's', 'omewhat', 'can', 'seeming', 'thin', 'another', 'con', 'third', 'sincere', 'af', 'terwards', 'nothing', 'becoming', 'least', 'together', 'amount', 'could', 'w', 'hence', 'they', 'around', 'toward', 'is', 'fifty', 'fill', 'whole', 'will', 'those', 'here', 'everywhere', 'such', 'once', 'therefore', 'inc', 'after', 'etc', 'five', 'we', 'because', 'down', 'ltd', 'never', 'call', 'get', 'acro', 'ss', 'even', 'wherein', 'well', 'sometimes', 'how', 'anyway', 'out', 'betwee', 'n', 'either', 'beside', 'much', 'are', 'was', 'again', 'someone', 'us', 'o', 'f', 'but', 'anywhere', 'there', 'somewhere', 'at', 'mostly', 'almost', 'if', 'go', 'has', 'itself', 'many', 'bottom', 'their', 'which', 'anyone', 'due', 'these', 'no', 'now', 'were', 'however', 'upon', 'within', 'the', 'might', 'thereafter', 'a', 'whenever', 'your', 'yourselves', 'cry', 'top', 'until', 'its', 'hereafter', 'interest', 'next', 'me', 'indeed', 'our', 'six', 'tha', 'n', 'that', 'couldnt', 'whereupon', 'whether', 'every', 'had', 'ours', 'as', 'all', 'so', 'moreover', 'ten', 'very', 'whoever', 'while', 'ever', 'less', 'others', 'serious', 'thick', 'thence', 'by', 'show', 'himself', 'herein', 'my', 'anything', 'empty', 'themselves', 'his', 'detail', 'from', 'sixty', 'towards', 'though', 'under', 'hundred', 'have', 'describe', 'mill', 'oursel', 'ves', 'amongst', 'beforehand', 'see', 'whereafter', 'except', 'sometime', 'whose', 'side', 'ie', 'none', 'part', 'de', 'few', 'who', 'meanwhile', 'an', 'y', 'without', 'system', 'hereby', 'against', 'enough', 'being', 'everyone', 'yet', 'full', 'he', 'seem', 'then', 'find', 'forty', 'hasnt', 'namely', 'no', 'body', 'co', 'also', 'up', 'nevertheless', 'during', 'via', 'about', 'been', 'hers', 'must', 'back', 'something', 'throughout', 'else', 'where', 'be', 'f', 'ront', 'thus', 'not', 'herself', 'nor', 'whatever', 'what', 'therein', 'whit', 'her', 'hence', 'rather', 'over', 'most', 'both', 'four', 'you', 'twelve', 'l', 'atter', 'own', 'this', 'mine', 'she', 'became', 'thereupon', 'further', 'alt', 'hough', 'eight', 'several', 'otherwise', 'for', 'latterly', 'should', 'som', 'e', 'keep', 'cannot', 'noone', 'them', 'becomes', 'only', 'one', 'put', 'bel', 'ow', 'neither', 'through', 'last', 'move', 'to', 'former', 'show', 'season', 'one', 'season', 'watch', 'story', 'bajpayee', 'webserie', 'webseries', 'baj', 'pai', 'manoj', 'episode', 'review', 'actor', 'actors', 'the', 'and', 'of', 'is', 'a', 'series', 'to', 'it', 'i', 'in', 'this', 'for', "didn't", 'more', 'each', "shouldn't", 'any', "aren't", 'http', 'these', 'no', 'com', "it's", 'same', 'in', 'were', "we'll", 'did', "we've", 'against', 'however', 'can', 'being', "wasn't", 'i', 'the', 'yourself', 'a', 'your', 'to', 'he', 'with', 'then', 'yourselves', "you'd", "you've", 'other', "here's", "doesn't", "sh", 'e'd', "there's", 'above', "i'd", 'until', 'its', 'could', "he'd", "can't", "weren't", 'also', 'they', 'or', 'up', "they're", "you're", 'am', 'me', "wo", 'n't', "he's", 'into', 'during', "she'll", "haven't", "they'd", 'is', 'just', 'since', "they've", 'about', 'been', 'our', "we're", 'than', 'k', 'hers', 't', 'hat', "where's", 'www', 'myself', 'would', "don't", "he'll", 'had', 'else', "couldn't", 'too', 'where', 'be', 'those', 'ours', 'here', 'as', 'all', "yo", 'u'll', 'such', 'so', 'once', 'therefore', 'after', 'her', "she's", 'not', 'v', 'ery', "what's", 'we', 'because', 'herself', 'down', 'like', 'while', 'nor', "we'd", 'ever', 'do', 'what', 'get', 'hence', 'over', "wouldn't", 'yours', 'by', 'most', "mustn't", 'whom', 'both', "hasn't", "let's", 'why', 'you', 'h', 'imself', 'an', 'him', 'my', 'how', 'own', 'this', 'she', 'themselves', 'ou
```

```
t', 'between', 'off', 'ought', "that's", 'his', 'on', "when's", 'from', "wh
y's", 'further', 'are', 'theirs', 'was', 'again', 'under', 'it', 'does', 'ot
herwise', "they'll", 'of', "shan't", 'have', 'but', 'for', 'when', 'should',
'some', 'there', 'shall', 'cannot', "who's", 'having', 'at', 'them', 'only',
'ourselves', 'if', 'below', 'few', 'and', 'r', 'doing', "hadn't", 'has', 'it
self', "isn't", 'through', 'their', "i'm", 'which', "i've", "i'll", 'befor
e', "how's", 'who']
```

Findings

We can clearly see that the word cloud has major chunk of positive reviews(roughly 80%) , some negative reviews (roughly 10%), with some neutral reviews(10%).

Let's dig into that and continue our analysis to back it up with statistical data.

Side Note

What was our goal for the EDA portion? To be able to take an initial look at our data and see if the results of some basic analysis made sense.

Guess what? Yes, now it does, for a first pass. There are definitely some things that could be better cleaned up, such as adding more stop words or including bi-grams. But we can save that for another day. The results, especially to our objective make general sense, so we're going to move on.

As a reminder, the data science process is an iterative one. It's better to see some non-perfect but acceptable results to help you quickly decide whether your project is inoperative or not.

Sentiment Analysis

Introduction

So far, all of the analysis we've done has been pretty generic - looking at counts, creating wordcloud plots, etc. These techniques could be applied to numeric data as well.

When it comes to text data, there are a few popular techniques that we may go through, starting with sentiment analysis. A few key points to remember with sentiment analysis.

1. **TextBlob Module:** Linguistic researchers have labeled the sentiment of words based on their domain expertise. Sentiment of words can vary based on where it is in a sentence. The TextBlob module allows us to take advantage of these labels.
2. **Sentiment Labels:** Each word in a corpus is labeled in terms of polarity and subjectivity (there are more labels as well, but we're going to ignore them for now). A corpus' sentiment is the average of these.
 - **Polarity:** How positive or negative a word is. -1 is very negative. +1 is very positive.
 - **Subjectivity:** How subjective, or opinionated a word is. 0 is fact. +1 is very much an opinion.

For more info on how TextBlob coded up its sentiment function. (https://planspace.org/20150607-textblob_sentiment/) (https://planspace.org/20150607-textblob_sentiment/)

Let's take a look at the sentiment of the various transcripts.

In [36]:

```
# Create quick lambda functions to find the polarity and subjectivity of each routine

from textblob import TextBlob

pol = lambda x: TextBlob(str(x)).sentiment.polarity
sub = lambda x: TextBlob(str(x)).sentiment.subjectivity

# Another way of writing the code , instead of using Lambda parameter above.
...
def get_Subjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def get_Polarity(text):
    return TextBlob(text).sentiment.polarity

...

data_clean_df['polarity'] = data_clean_df['transcript'].apply(pol)
data_clean_df['subjectivity'] = data_clean_df['transcript'].apply(sub)
data_clean_df
```

Out[36]:

	transcript	polarity	subjectivity
0	reviews of sacred games thriller	0.000000	0.00000
1	nan	0.000000	0.00000
2	nan	0.000000	0.00000
3	had read the sprawling world building novel by...	0.220000	0.36000
4	simply brilliant content to binge watch stella...	0.356795	0.52493
...
8443	good onebut was it necessary to use the dirty ...	0.033333	0.80000
8444	wating for seasion better then	0.500000	0.50000
8445	worth watching but they could have done better	0.400000	0.30000
8446	series is good but too many loop holes	0.600000	0.55000
8447	very intresting and full of twist show	0.275000	0.42500

8448 rows × 3 columns

In [37]:

```
data_clean_df.sample(5)
```

Out[37]:

	transcript	polarity	subjectivity
8295	this type of serials is a waste of time the st...	-0.200000	0.000000
4341	ab is too good superb acting	0.566667	0.533333
3982	it was not at all worth my time nothing made s...	0.328571	0.671429
4484	scary and over smart but maddy lived in his role	-0.142857	0.821429
7891	i didnt like itastrology has been accorded too...	0.306667	0.556667

In [38]:

```
#classifying sentiments based on the reviews'score
def get_analysis(score):
    if score > 0:
        return "positive"
    elif score < 0:
        return "negative"
    else:
        return 'neutral'
data_clean_df["Analysis"] = data_clean_df.polarity.apply(get_analysis)
data_clean_df
```

Out[38]:

	transcript	polarity	subjectivity	Analysis
0	reviews of sacred games thriller	0.000000	0.00000	neutral
1	nan	0.000000	0.00000	neutral
2	nan	0.000000	0.00000	neutral
3	had read the sprawling world building novel by...	0.220000	0.36000	positive
4	simply brilliant content to binge watch stella...	0.356795	0.52493	positive
...
8443	good onebut was it necessary to use the dirty ...	0.033333	0.80000	positive
8444	wating for seasion better then	0.500000	0.50000	positive
8445	worth watching but they could have done better	0.400000	0.30000	positive
8446	series is good but too many loop holes	0.600000	0.55000	positive
8447	very intresting and full of twist show	0.275000	0.42500	positive

8448 rows × 4 columns

In [39]:

```
data_clean_df.sample(10)
```

Out[39]:

	transcript	polarity	subjectivity	Analysis
48	congratulations to netflix and the team of sac...	0.103889	0.515556	positive
4793	interesting but a low budget series	0.250000	0.400000	positive
2850	realistic to an extent but the ease with polic...	-0.011111	0.177778	negative
3676	awsome but only for should watch this	0.000000	1.000000	neutral
4651	one of the best web series full of thriller	0.675000	0.425000	positive
5938	this show is overwhelming the unique cast make...	0.537771	0.568615	positive
457	the acting is simply awesome you should watch ...	0.000000	0.500000	neutral
6582	unbelievably real cant remember any better ind...	0.350000	0.400000	positive
187	really appreciate the series i used to watch o...	0.133333	0.316667	positive
1483	very sensitive and thriller	0.130000	1.000000	positive

In [40]:

```

j=0
k=0
for i in range(0,data_clean_df.shape[0]):
    if data_clean_df.Analysis[i]=='negative':

        j= j+1
    elif data_clean_df.Analysis[i]=='positive':
#The following code can be undocumented , if you're interested in reading that sentiments'
        #         print (k,data_clean_df.transcript[i])
        k+=1
neu= data_clean_df.shape[0]- (j+k)
print ('So,The following is our "Sentiment Analysis" for the Top 10 Indian Web Series(Thriller Genre)')
print ('\nNo of Negative Reviews from our Total DataSet(around 10k) ->',j)
print ('No of Positive Reviews from our Total DataSet(around 10k) ->',k)
print ('No of Neutral Reviews from our Total DataSet(around 10k) ->',neu)

neg_per= (j/data_clean_df.shape[0])*100
pos_per=(k/data_clean_df.shape[0])*100
neu_per=(neu/data_clean_df.shape[0])*100

print('\nPercentage of Negative Reviews -> '+ str(neg_per) + " %")
print('Percentage of Positive Reviews -> '+ str(pos_per) + ' %')
print('Percentage of Neutral Reviews -> '+ str(neu_per) + " %" )

```

So,The following is our "Sentiment Analysis" for the Top 10 Indian Web Series(Thriller Genre) :

No of Negative Reviews from our Total DataSet(around 10k) -> 1014
 No of Positive Reviews from our Total DataSet(around 10k) -> 6594
 No of Neutral Reviews from our Total DataSet(around 10k) -> 840

Percentage of Negative Reviews -> 12.002840909090908 %
 Percentage of Positive Reviews -> 78.05397727272727 %
 Percentage of Neutral Reviews -> 9.943181818181818 %

Sentiment Findings:

So,The following is our "Sentiment Analysis" for the Top 10 Indian Web Series(Thriller Genre) :

No of Negative Reviews from our Total DataSet(around 10k) -> 1014
 No of Positive Reviews from our Total DataSet(around 10k) -> 6594
 No of Neutral Reviews from our Total DataSet(around 10k) -> 840

Percentage of Negative Reviews -> 12.002840909090908 %
 Percentage of Positive Reviews -> 78.05397727272727 %
 Percentage of Neutral Reviews -> 9.943181818181818 %

This also confirms our vague analysis that we did using just the wordcloud sentiments.

Data Visualizations

Data Visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

The advantages and benefits of good data visualization

Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It's basically storytelling with a purpose.

Other benefits of data visualization include the following:

- **Confirms our results derived from numeric data analysis.**
- The ability to absorb information quickly, improve insights and make faster decisions;
- An increased understanding of the next steps that must be taken to improve the organization;
- An improved ability to maintain the audience's interest with information they can understand;
- An easy distribution of information that increases the opportunity to share insights with everyone involved;
- Eliminate the need for data scientists since data is more accessible and understandable; and
- An increased ability to act on findings quickly and, therefore, achieve success with greater speed and less mistakes.

In [41]:

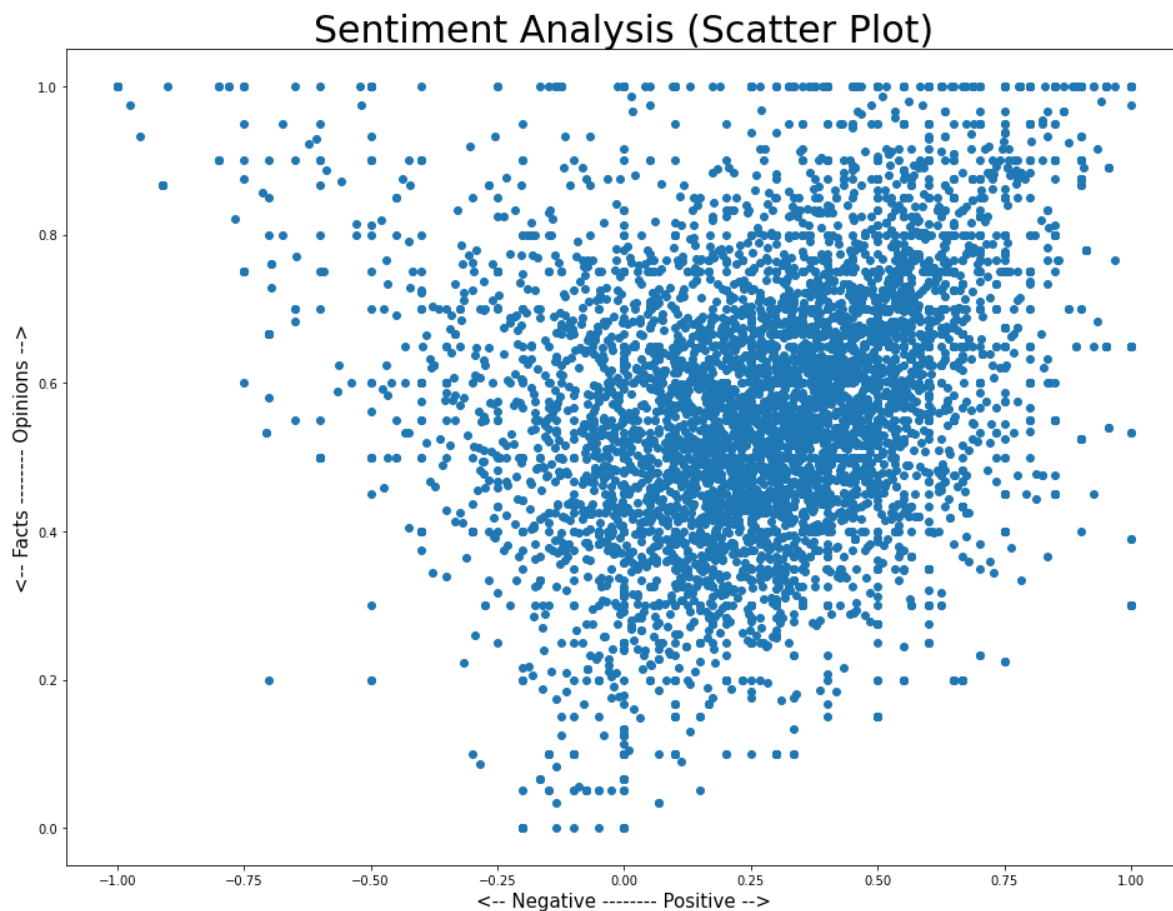
```
# Let's plot the results
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = [16, 12]

plt.scatter(data_clean_df['polarity'], data_clean_df['subjectivity'])

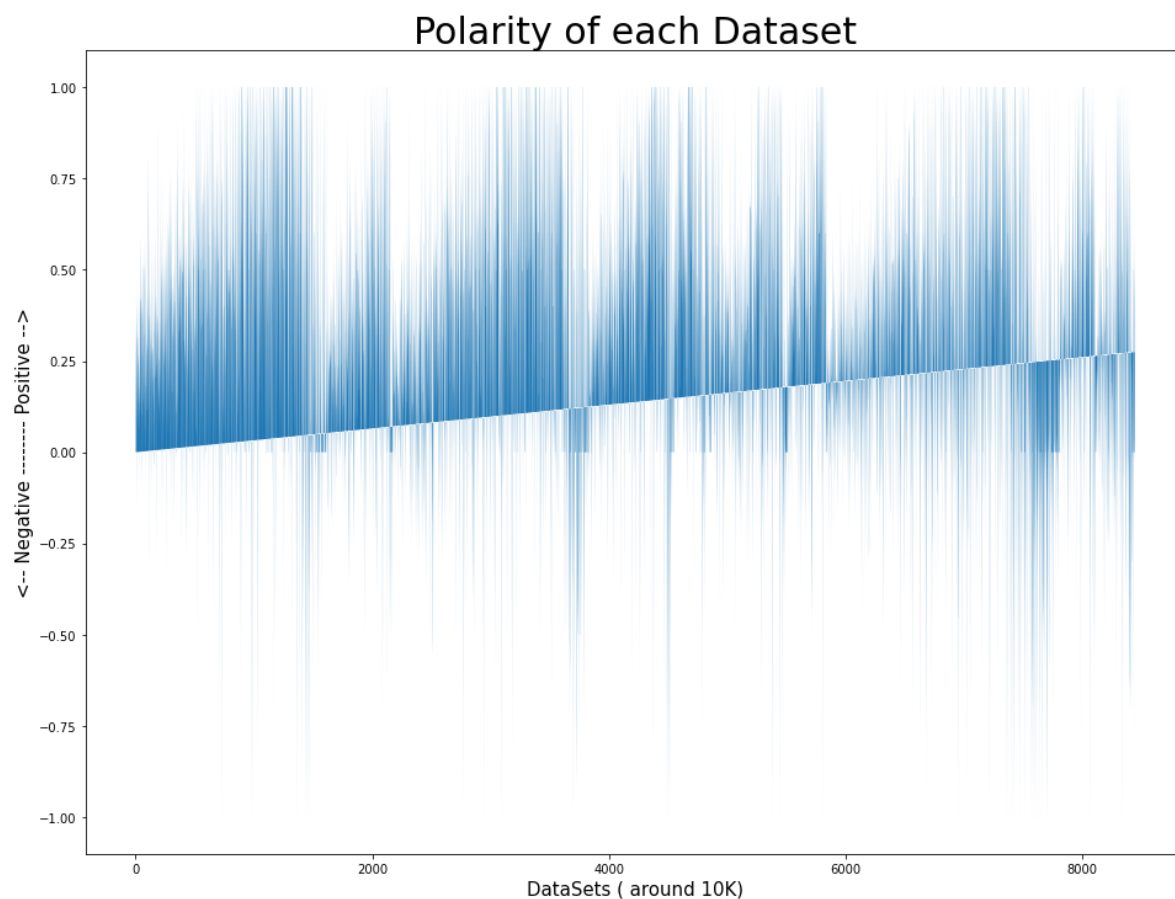
plt.title('Sentiment Analysis (Scatter Plot)', fontsize=30)
plt.xlabel('<-- Negative ----- Positive -->', fontsize=15)
plt.ylabel('<-- Facts ----- Opinions -->', fontsize=15)

plt.show()
```



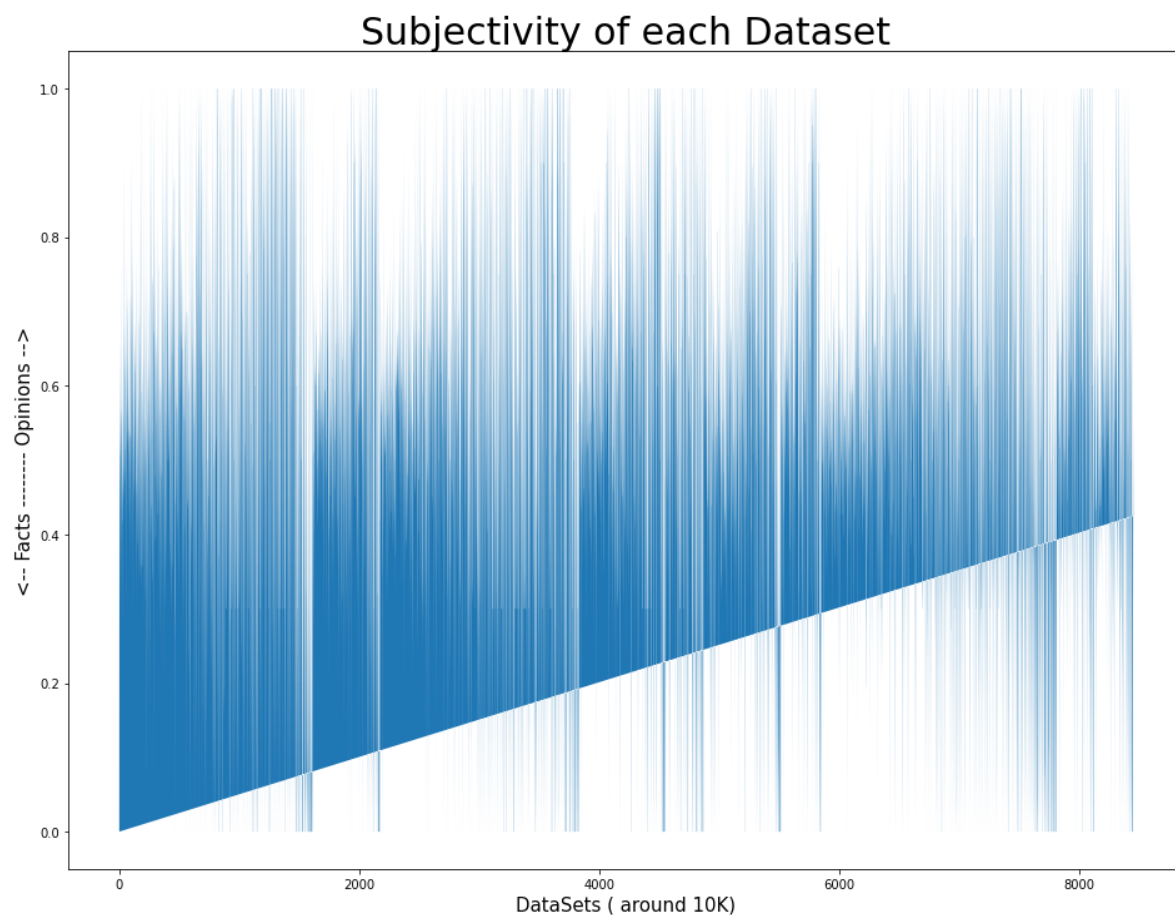
In [42]:

```
plt.fill(data_clean_df['polarity'])  
plt.title('Polarity of each Dataset', fontsize=30)  
plt.xlabel('DataSets ( around 10K)', fontsize=15)  
plt.ylabel('<-- Negative ----- Positive -->', fontsize=15)  
  
plt.show()
```



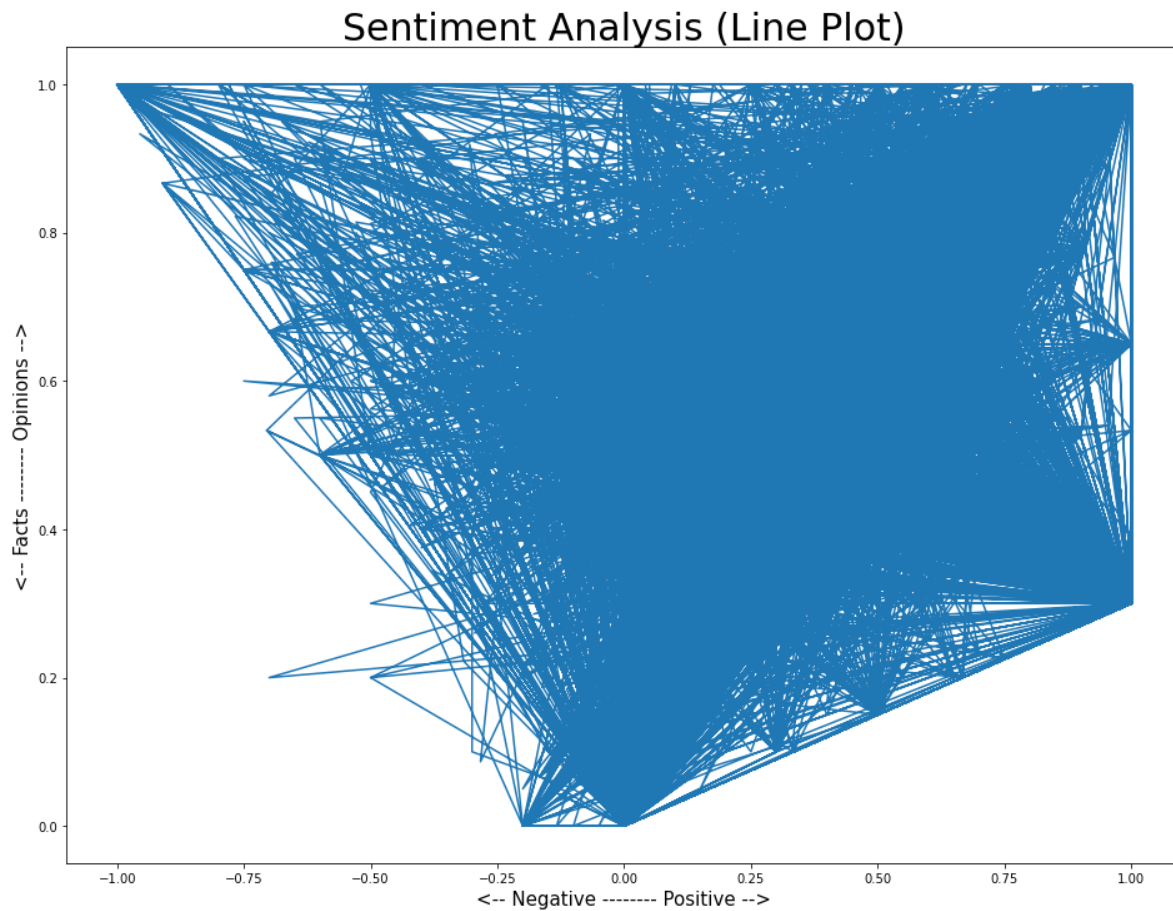
In [43]:

```
plt.fill(data_clean_df['subjectivity'])  
plt.title('Subjectivity of each Dataset', fontsize=30)  
plt.xlabel('DataSets ( around 10K)', fontsize=15)  
plt.ylabel('<-- Facts ----- Opinions -->', fontsize=15)  
  
plt.show()
```



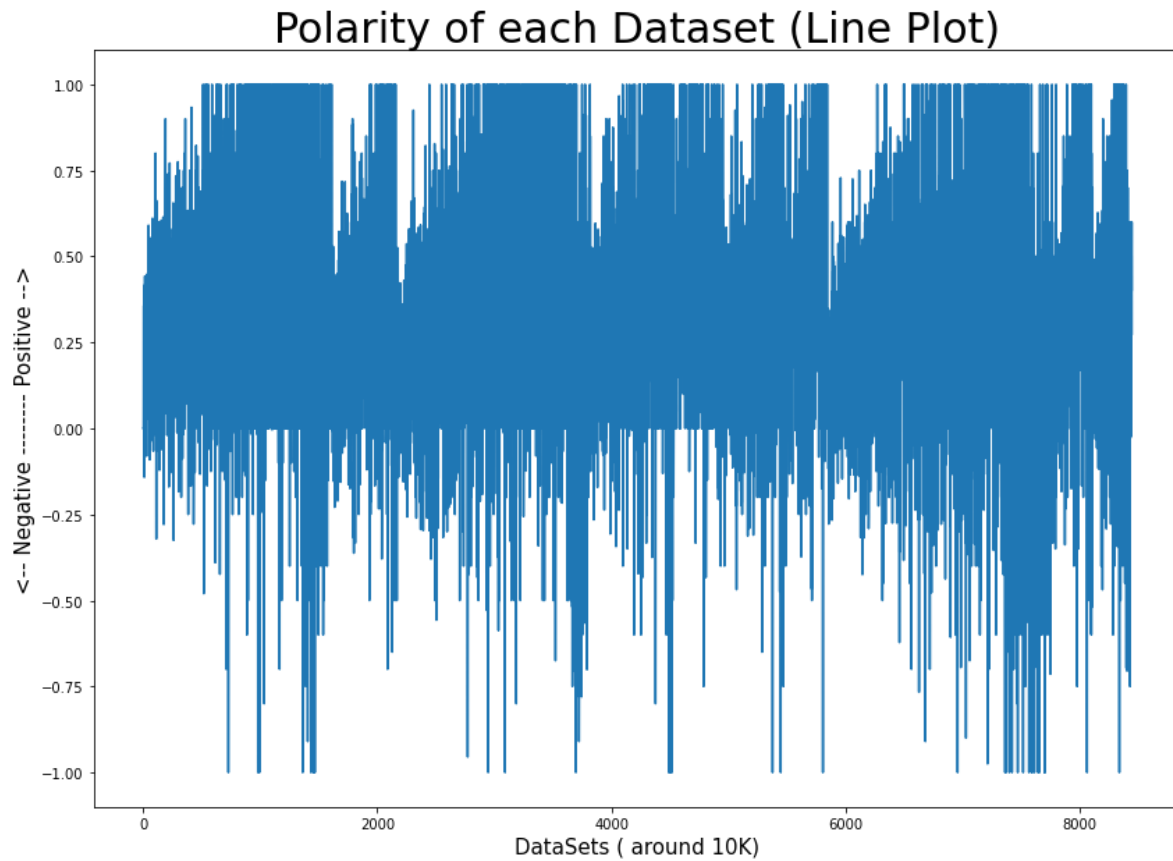
In [44]:

```
plt.plot(data_clean_df['polarity'],data_clean_df['subjectivity'])
plt.rcParams['figure.figsize'] = [14, 10]
plt.title('Sentiment Analysis (Line Plot)', fontsize=30)
plt.xlabel('<-- Negative ----- Positive -->', fontsize=15)
plt.ylabel('<-- Facts ----- Opinions -->', fontsize=15)
plt.show()
```



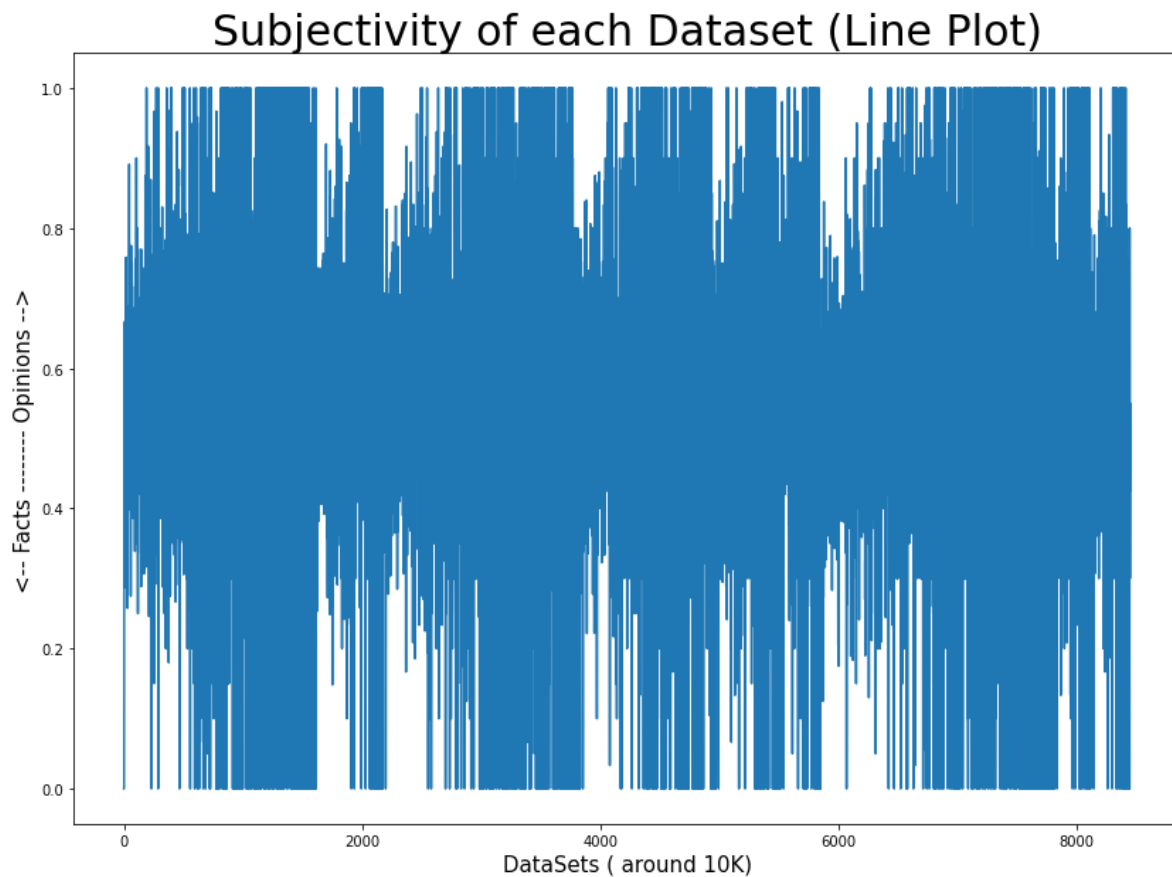
In [45]:

```
plt.plot(data_clean_df['polarity'])  
plt.title('Polarity of each Dataset (Line Plot)', fontsize=30)  
plt.xlabel('DataSets ( around 10K)', fontsize=15)  
plt.ylabel('<-- Negative ----- Positive -->', fontsize=15)  
  
plt.show()
```



In [46]:

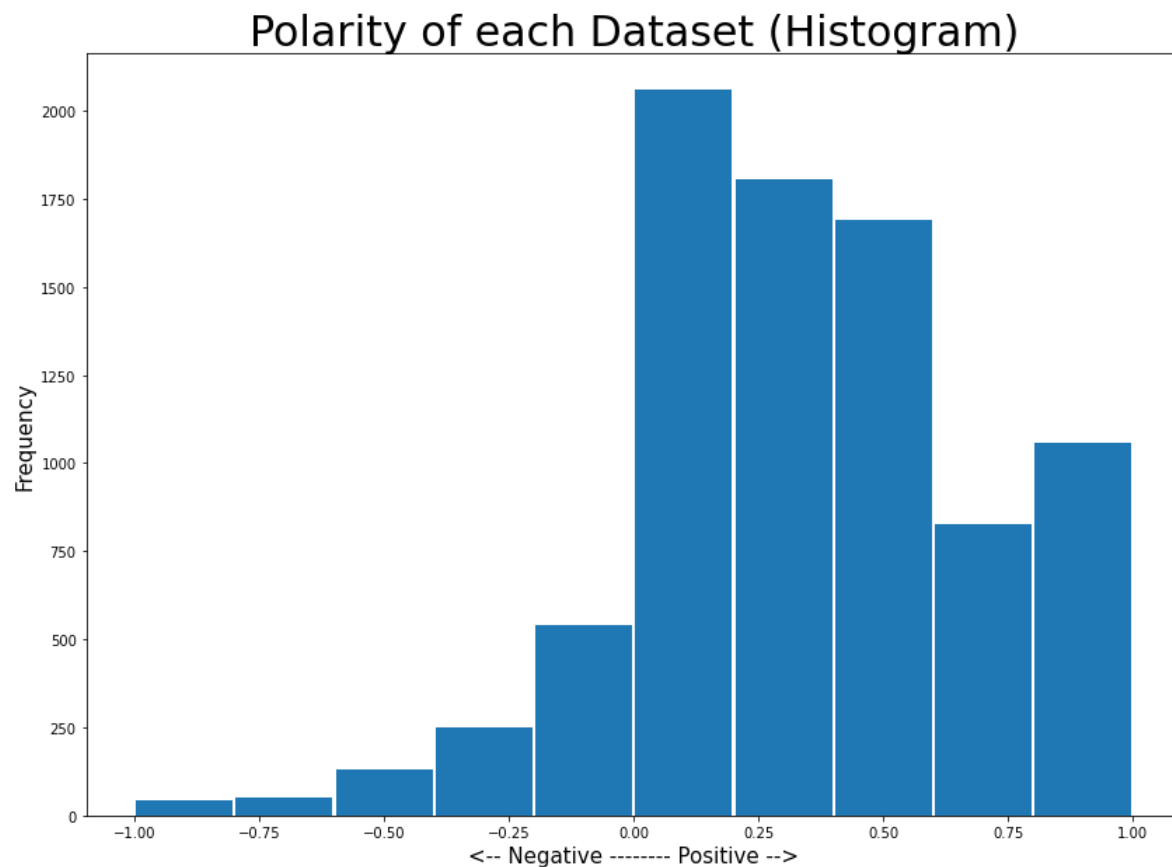
```
plt.plot(data_clean_df['subjectivity'])  
plt.title('Subjectivity of each Dataset (Line Plot)', fontsize=30)  
plt.xlabel('DataSets ( around 10K)', fontsize=15)  
plt.ylabel('<--- Facts ----- Opinions -->', fontsize=15)  
  
plt.show()
```



In [47]:

```
plt.hist(data_clean_df['polarity'], rwidth=.969)
plt.title('Polarity of each Dataset (Histogram)', fontsize=30)
plt.xlabel('<-- Negative ----- Positive -->', fontsize=15)
plt.ylabel('Frequency', fontsize=15)

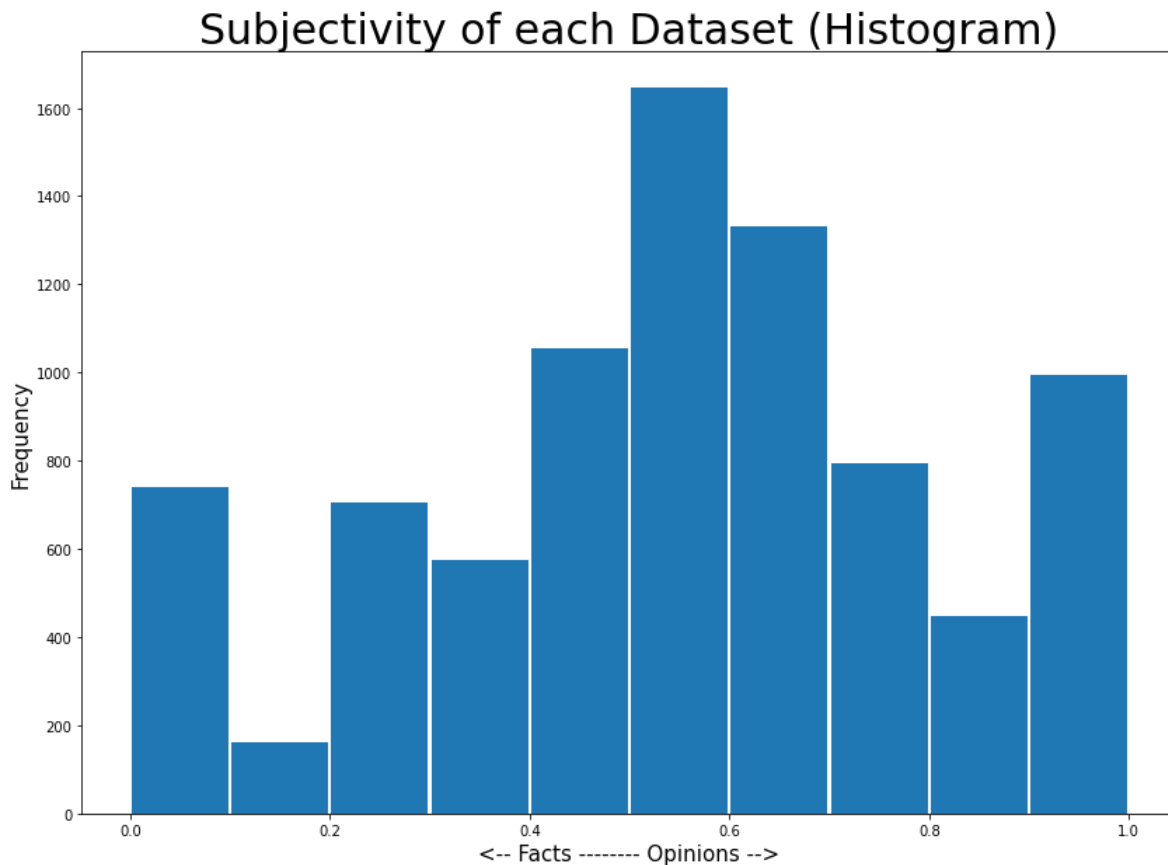
plt.show()
```



In [48]:

```
plt.hist(data_clean_df['subjectivity'], rwidth=.969)
plt.title('Subjectivity of each Dataset (Histogram)', fontsize=30)
plt.xlabel('<-- Facts ----- Opinions -->', fontsize=15)
plt.ylabel('Frequency', fontsize=15)

plt.show()
```



In [49]:

data_clean_df

Out[49]:

	transcript	polarity	subjectivity	Analysis
0	reviews of sacred games thriller	0.000000	0.00000	neutral
1	nan	0.000000	0.00000	neutral
2	nan	0.000000	0.00000	neutral
3	had read the sprawling world building novel by...	0.220000	0.36000	positive
4	simply brilliant content to binge watch stella...	0.356795	0.52493	positive
...
8443	good onebut was it necessary to use the dirty ...	0.033333	0.80000	positive
8444	wating for seasion better then	0.500000	0.50000	positive
8445	worth watching but they could have done better	0.400000	0.30000	positive
8446	series is good but too many loop holes	0.600000	0.55000	positive
8447	very intresting and full of twist show	0.275000	0.42500	positive

8448 rows × 4 columns

In [50]:

```
#Creating a new DataFrame with only Positive Reviews.
#We will later use this df to create a wordcloud having only positive sentiments.
positive_df=data_clean_df[data_clean_df['Analysis']=='positive']
```

In [51]:

positive_df

Out[51]:

	transcript	polarity	subjectivity	Analysis
3	had read the sprawling world building novel by...	0.220000	0.360000	positive
4	simply brilliant content to binge watch stella...	0.356795	0.524930	positive
5	it is a must watch series and for me it was th...	0.065385	0.428322	positive
6	its indian storytelling at its peak what i lov...	0.303175	0.666270	positive
7	sacred games explores a wide array of narrativ...	0.088889	0.511111	positive
...
8443	good onebut was it necessary to use the dirty ...	0.033333	0.800000	positive
8444	wating for seasion better then	0.500000	0.500000	positive
8445	worth watching but they could have done better	0.400000	0.300000	positive
8446	series is good but too many loop holes	0.600000	0.550000	positive
8447	very intresting and full of twist show	0.275000	0.425000	positive

6594 rows × 4 columns

In [52]:

```
# Python program to generate WordCloud for POSITIVE SENTIMENTS

# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

comment_words_pos = ''

# Add new stop words

selected_stop_words=['show','season','one','season','watch','story','web','episodes','bajpa
stopwords = list(additional_stop_words) + selected_stop_words + add_stop_words + list(STOPW

# iterate through the file
for val in positive_df.transcript:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()
    comment_words_pos += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words_pos)

# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title('WordCloud for POSITIVE SENTIMENTS\n',fontsize=30)

plt.show()
```

[illegible]

```
#Creating a new DataFrame with only Negative Reviews.
#We will later use this df to create a wordcloud having only negative sentiments.
negative_df=data_clean_df[data_clean_df['Analysis']=='negative']
```


In [54]:

negative_df

Out[54]:

	transcript	polarity	subjectivity	Analysis
9	sacred games is the premiere hindi netflix ser...	-0.141964	0.375446	negative
28	heard about this show and seen lot of marketin...	-0.080556	0.427991	negative
55	you need to have enough mind to understand the...	-0.090909	0.397727	negative
58	at the end of all the episodes which i watched...	-0.075000	0.275000	negative
71	the cast is amazing this is indianpakistanipun...	-0.035000	0.511167	negative
...
8424	loses its plot and the grip episode by episode	-0.300000	0.100000	negative
8425	waste of time no proper climax	-0.100000	0.050000	negative
8427	not at all interesting just boring	-0.250000	0.750000	negative
8431	never expect this from series disappointed	-0.750000	0.750000	negative
8439	in my opinion some episode was extra stretched	-0.025000	0.050000	negative

1014 rows × 4 columns

In [57]:

```
# Python program to generate WordCloud for NEGATIVE SENTIMENTS

# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

comment_words_neg = ''

# Add new stop words

selected_stop_words=['show', 'season', 'one', 'good', 'season', 'watch', 'story', 'bajpayee', 'bajp
stopwords = list(additional_stop_words) + selected_stop_words + add_stop_words + list(STOPW

# iterate through the file
for val in negative_df.transcript:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()
    comment_words_neg += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                      background_color = 'white',
                      stopwords = stopwords,
                      min_font_size = 10).generate(comment_words_neg)

# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title('WordCloud for NEGATIVE SENTIMENTS\n', fontsize=30)

plt.show()
```

WordCloud for NEGATIVE SENTIMENTS



In [58]:

```
#Creating a new DataFrame with only Neutral Reviews.  
#We will later use this df to create a wordcloud having only neutral sentiments.  
neutral_df = data_clean_df[data_clean_df['Analysis']=='neutral']
```

In [59]:

```
# Python program to generate WordCloud for NEUTRAL SENTIMENTS

# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

comment_words_neu = ''

# Add new stop words

selected_stop_words=['show','season','one','good','thriller','season','shame','watch','stor
stopwords = list(additional_stop_words) + selected_stop_words + add_stop_words + list(STOPW

# iterate through the file
for val in neutral_df.transcript:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()
    comment_words_neu += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                      background_color = 'white',
                      stopwords = stopwords,
                      min_font_size = 10).generate(comment_words_neu)

# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title('WordCloud for NEUTRAL SENTIMENTS\n',fontsize=30)

plt.show()
```

[illegible]

The most frequent words from POSITIVE , NEGATIVE and NEUTRAL REVIEWS' data set.

```
# Python program to find the most frequent words from POSITIVE REVIEWS' data set
from collections import Counter

# split() returns list of all the words in the string
split_it = comment_words_pos.split(" ")

# Pass the split_it list to instance of Counter class.
Counter = Counter(split_it)

# most_common() produces k frequently encountered
# input values and their respective counts.
most_occur = Counter.most_common()

print(most_occur)
```

[('the', 7606), ('and', 5026), ('of', 4059), ('is', 3547), ('series', 3486), ('a', 3441), ('to', 3058), ('it', 2561), ('i', 2417), ('in', 2220), ('this', 1983), ('for', 1570), ('best', 1511), ('watch', 1344), ('good', 1276), ('was', 1254), ('story', 1246), ('acting', 1138), ('all', 1111), ('very', 1087), ('one', 1084), ('but', 1078), ('web', 1071), ('you', 1056), ('its', 997), ('season', 997), ('with', 990), ('by', 972), ('are', 955), ('have', 938), ('show', 898), ('that', 887), ('great', 830), ('amazing', 750), ('as', 743), ('not', 742), ('just', 725), ('so', 683), ('on', 668), ('indian', 640), ('awesome', 618), ('must', 609), ('like', 597), ('well', 582), ('has', 578), ('from', 531), ('be', 513), ('an', 486), ('more', 465), ('really', 464), ('watching', 462), ('will', 448), ('actors', 432), ('what', 429), ('every', 412), ('thriller', 405), ('which', 399), ('ever', 387), ('loved', 381), ('watched', 379), ('superb', 379), ('excellent', 375), ('much', 373), ('my', 372), ('characters', 371), ('can', 370), ('crime', 364), ('brilliant', 363), ('at', 361), ('time', 357), ('love', 351), ('also', 343), ('direction', 341), ('his', 337), ('character', 333), ('india', 332), ('waiting', 331), ('they', 327), ('episode', 325), ('if', 320), ('performance', 317), ('nice', 315), ('too', 314), ('after', 313), ('some', 311), ('about', 310), ('there', 308), ('their', 303), ('cast', 301), ('the', 299), ('that', 298), ('it', 298), ('like', 298), ('the', 296)]

In [61]:

```
# Python program to find the most frequent words from NEGATIVE REVIEWS' data set
from collections import Counter

# split() returns list of all the words in the string
split_it = comment_words_neg.split(" ")

# Pass the split_it list to instance of Counter class.
Counter = Counter(split_it)

# most_common() produces k frequently encountered
# input values and their respective counts.
most_occur = Counter.most_common()

print(most_occur)
```

```
[('the', 1298), ('and', 820), ('of', 804), ('to', 670), ('is', 630), ('a', 620), ('it', 457), ('in', 443), ('series', 438), ('i', 418), ('this', 382), ('not', 269), ('for', 226), ('but', 221), ('you', 217), ('are', 214), ('story', 210), ('that', 208), ('with', 203), ('was', 202), ('very', 177), ('show', 172), ('watch', 170), ('have', 165), ('all', 163), ('on', 155), ('as', 150), ('its', 147), ('time', 142), ('season', 134), ('web', 134), ('like', 132), ('worst', 124), ('be', 123), ('one', 123), ('by', 114), ('just', 113), ('dont', 113), ('so', 110), ('no', 108), ('they', 107), ('waste', 104), ('watching', 99), ('bad', 99), ('can', 95), ('has', 94), ('there', 92), ('such', 91), ('will', 91), ('at', 90), ('which', 89), ('too', 86), ('or', 84), ('good', 83), ('what', 83), ('from', 81), ('only', 77), ('about', 76), ('how', 76), ('an', 75), ('episode', 75), ('if', 74), ('some', 73), ('people', 73), ('acting', 73), ('we', 71), ('much', 71), ('scenes', 68), ('who', 66), ('your', 62), ('episodes', 62), ('after', 61), ('any', 60), ('my', 59), ('up', 58), ('watched', 57), ('india', 56), ('hindu', 55), ('must', 54), ('ever', 53), ('boring', 53), ('well', 52), ('me', 52), ('make', 52), ('crime', 51), ('police', 50), ('really', 50), ('even', 50), ('their', 50), ('plot', 50), ('slow', 50), ('other', 49), ('nothing', 49), ('would', 48), ('also', 48), ('way', 48), ('characters', 47), ('see', 47), ...]
```

In [62]:

```
# Python program to find the most frequent words from NEGUTRAL REVIEWS' data set
from collections import Counter

# split() returns list of all the words in the string
split_it = comment_words_neu.split(" ")

# Pass the split_it list to instance of Counter class.
Counter = Counter(split_it)

# most_common() produces k frequently encountered
# input values and their respective counts.
most_occur = Counter.most_common()

print(most_occur)
```

```
[('the', 310), ('of', 235), ('series', 193), ('a', 186), ('and', 176), ('t
o', 158), ('is', 154), ('watch', 151), ('i', 142), ('for', 136), ('it', 13
0), ('this', 120), ('in', 104), ('season', 99), ('must', 92), ('its', 75),
('web', 71), ('one', 69), ('just', 66), ('like', 66), ('all', 58), ('waiti
ng', 58), ('show', 49), ('you', 47), ('not', 45), ('story', 42), ('was', 4
2), ('be', 41), ('acting', 41), ('by', 40), ('thriller', 38), ('indian', 3
8), ('have', 37), ('that', 37), ('mind', 35), ('no', 35), ('on', 35), ('wh
at', 33), ('with', 31), ('will', 31), ('as', 30), ('only', 29), ('mirzapu
r', 29), ('games', 28), ('should', 28), ('sacred', 27), ('are', 27), ('bu
t', 27), ('episode', 26), ('if', 25), ('next', 25), ('dont', 24), ('my', 2
4), ('from', 24), ('after', 24), ('well', 23), ('crime', 22), ('an', 22),
('can', 22), ('we', 21), ('they', 21), ('has', 20), ('netflix', 20), ('blo
wing', 20), ('at', 19), ('india', 19), ('see', 19), ('time', 19), ('up', 1
8), ('there', 18), ('about', 18), ('his', 17), ('such', 17), ('movie', 1
7), ('nan', 16), ('who', 16), ('am', 16), ('ever', 16), ('hindu', 15), ('s
o', 15), ('out', 15), ('level', 15), ('every', 15), ('word', 15), ('peopl
e', 15), ('masterpiece', 15), ('watching', 15), ('performance', 15), ('wor
ds', 15), ('suspense', 14), ('tv', 14), ('me', 14), ('made', 14), ('when',
14), ('eagerly', 14), ('life', 14), ('reality', 14), ('ultimate', 14), ('l
...
```

THANK YOU

- By Harsh Kumar (Delhi Technological University,DTU (formerly Delhi College of Engineering,DCE))

- Intern under Prof. Sasadhar Bera, Ph.D. (Indian Institute of Management ,Ranchi)