Opinion Mining + Sentiment Classification :

For the Top 10 Indian Web Series(Action 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 Action Genre, on which our further analysis are done.

In [1]:

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

In [2]:

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

In [3]:

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

Out[3]:

	Unnamed: 0
0	NaN
1	REVIEWS OF MAI HERO BOLL RAHA HU
2	NaN
3	Parth just nailed the role of nawab...as he re...
4	I love the series .it is amazing. Parth acting...
...	...
349	It will only satisfy debutants, starting to wa...
350	Bhai ? If the real Bhai watches this series he...
351	What a terribly done gangster series! The hero...
352	Copy of once upon a time in Mumbai don't watch...
353	Well, such shuru revolving about Gangsters hav...

354 rows × 1 columns

In [4]:

```
r2_df
```

Out[4]:

	Unnamed: 0
0	REVIEWS OF FAMILY MAN
1	NaN
2	NaN
3	stupid people dont watch their disclaimers and...
4	There is a considerably visible yet veiled lin...
...	...
2680	Awesome story must watch
2681	Awesome entertainment awesome acting
2682	i love thyis show
2683	must watch.very good story.fully excited.
2684	Excellent storyline..Excellent acting...now wa...

2685 rows × 1 columns

In [5]:

```
r3_df
```

Out[5]:

	Unnamed: 0
0	NaN
1	REVIEWS OF SPECIAL OPS
2	NaN
3	This is best web series in the world this king...
4	As you read on, I know this will sound quite a...
...	...
2428	Jbrdst 👌👌👌 Hollywood leval ki movie h .....
2429	Typical neeraj pandey thriller
2430	Any chances of season 2?? 😍😍
2431	Awesome!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2432	It's a Masterpiece web series!!!! 🔥🔥🔥

2433 rows × 1 columns

In [6]:

```
r4_df
```

Out[6]:

	Unnamed: 0
0	REVIEWS OF JEET KI ZID
1	A must watch series...for all the defence peop...
2	I really liked the acting of Amit Sadh..Very n...
3	Excellent!! Must watch series.. Complete packa...
4	Amazing never give up series. To know it's bas...
...	...
322	Rating for first 3-4 episodes only..
323	Not as expected
324	Ok what a joke!
325	Waste of good talent by poor direction
326	inspiring ... but you feel something is missin...

327 rows × 1 columns

In [7]:

```
r5_df
```

Out[7]:

	Unnamed: 0
0	REVIEWS OF THE TEST CASE
1	NaN
2	NaN
3	Superb serial on a lady who is going to prove ...
4	This is the best web series i have ever watche...
...	...
208	Really loved it....
209	Must watch.
210	Where is the second episode?
211	Worth watching. Must watch
212	Khara ...must watch

213 rows × 1 columns

In [8]:

```
r6_df
```

Out[8]:

	Unnamed: 0
0	REVIEWS OF THE FORGOTTEN ARMY
1	NaN
2	NaN
3	This is the series which has been able to show...
4	One of the best series of Indian history.Freed...
...	...
431	Please Make this kind of content more
432	Truth that we don't want to forget again.
433	Government should start pension for 'INA' sold...
434	It's awesome .....👍👍👍👍
435	Trying to push emotions to viewers

436 rows × 1 columns

In [9]:

```
r7_df
```

Out[9]:

	Unnamed: 0
0	NaN
1	REVIEWS OF RANGBAAZ
2	NaN
3	This series has written Tigmanshu dhulia all o...
4	This is one of those fabulous series which can...
...	...
343	Do not trust on the rating as each and every r...
344	Wow . Just wow . Excellent . Superb. Fantastic...
345	Sasta budget wali series isme sirf ek insan ki...
346	So many positive reviews??? concept is very ol...
347	Good story, good direction , acting and actors...

348 rows × 1 columns

In [10]:

```
r8_df
```

Out[10]:

	Unnamed: 0
0	NaN
1	REVIEWS OF ABHAY
2	NaN
3	I love Kunal Khemu. He has always been my favo...
4	Abhay is back. I'm already so in love with thi...
...	...
1481	I was expecting the same old 30 minutes crime ...
1482	Watchable, if you have nothing left in your bu...
1483	Kunal Khemu disappoints in an author backed ro...
1484	This is very well done. Watch it atleast one t...
1485	Entire team did a good job, mostly the antagon...

1486 rows × 1 columns

In [11]:

```
r9_df
```

Out[11]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2
0	NaN	NaN	NaN
1	REVIEWS OF JAMTARA-SABKA NUMBER AYEGA	NaN	NaN
2	NaN	NaN	NaN
3	Most of us have heard stories about Jamtara[a ...	NaN	NaN
4	I loved watching Jamtara as 10 episodes don't ...	NaN	NaN
...	...	...	...
418	Starcast is simple but most of them have justi...	NaN	NaN
419	The story has so many different angles. Amazin...	NaN	NaN
420	The subject is interesting.\nThere are gluing ...	NaN	NaN
421	I'm very satisfied with the series.It keeps yo...	NaN	NaN
422	One of the best TV series on Netflix right now...	NaN	NaN

423 rows × 3 columns

In [12]:

```
r10_df
```

Out[12]:

	Unnamed: 0
0	REVIEWS OF BARD OF BLOOD
1	NaN
2	NaN
3	9/10\n\nI don't know why Netflix is dilevering...
4	As this is released back to back with Amazon S...
...	...
1191	Not seen yet
1192	entertainment is just for entertainment.
1193	AVERAGE .....
1194	Time pass series .
1195	Please avoid this one.

1196 rows × 1 columns

In [13]:

```
#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 [14]:

```
combined_df
```

Out[14]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2
0	NaN	NaN	NaN
1	REVIEWS OF MAI HERO BOLL RAHA HU	NaN	NaN
2	NaN	NaN	NaN
3	Parth just nailed the role of nawab...as he re...	NaN	NaN
4	I love the series .it is amazing. Parth acting...	NaN	NaN
...	...	...	...
9896	Not seen yet	NaN	NaN
9897	entertainment is just for entertainment.	NaN	NaN
9898	AVERAGE	NaN	NaN
9899	Time pass series .	NaN	NaN
9900	Please avoid this one.	NaN	NaN

9901 rows × 3 columns

In [15]:

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

In [16]:

```
#deleting the additional empty columns
combined_df.pop('a')
combined_df.pop('b')
```

Out[16]:

```
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
       ...
9896    NaN
9897    NaN
9898    NaN
9899    NaN
9900    NaN
Name: b, Length: 9901, dtype: object
```

In [17]:

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

Out[17]:

	transcript
0	NaN
1	REVIEWS OF MAI HERO BOLL RAHA HU
2	NaN
3	Parth just nailed the role of nawab...as he re...
4	I love the series .it is amazing. Parth acting...
...	...
9896	Not seen yet
9897	entertainment is just for entertainment.
9898	AVERAGE .....
9899	Time pass series .
9900	Please avoid this one.

9901 rows × 1 columns

In [18]:

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

Out[18]:

	transcript
2599	We liked the show but end is depressing....alr...
2796	Tagda show h. Having lots of fun snd information.
4668	Super script, best direction and awesome perfo...
9243	The web series bard of blood is good but the s...
9300	best thriller web series i have ever watched t...
7330	Also, I got goosebumps on in the end. Share it...
4398	Starcast : Just Superb\nEnd Of Episode : Oyy B...
2895	awesome web series by amazon prime video.\n\nm...
4583	Extremely well directed.....One of the best sh...
8264	Such a great acting by kunal khemu and other a...

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 [20]:

```
# Let's take a look at the updated text
data_clean_df = pd.DataFrame(combined_df.transcript.apply(clean_text_round1))
data_clean_df
```

Out[20]:

	transcript
0	nan
1	reviews of mai hero boll raha hu
2	nan
3	parth just nailed the role of nawabas he recei...
4	i love the series it is amazing parth acting i...
...	...
9896	not seen yet
9897	entertainment is just for entertainment
9898	average
9899	time pass series
9900	please avoid this one

9901 rows × 1 columns

In [21]:

```
# Apply a second round of cleaning
def clean_text_round2(text):
    '''Get rid of some additional punctuation and non-sensical text that was missed the fir
    text = str(text)
    text = re.sub('[‘’“”…]', '', text)
    text = re.sub('\n', '', 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	nan
1	reviews of mai hero boll raha hu
2	nan
3	parth just nailed the role of nawabas he recei...
4	i love the series it is amazing parth acting i...
...	...
9896	not seen yet
9897	entertainment is just for entertainment
9898	average
9899	time pass series
9900	please avoid this one

9901 rows × 1 columns

In [23]:

```
randomcheck=data_clean_df.loc[2694]
randomcheck
# emoji still present.
```

Out[23]:

```
transcript    awesome seriescant wait to see next season ❤👌
Name: 2694, dtype: object
```

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))
data_clean_df
```

Out[25]:

	transcript
0	nan
1	reviews of mai hero boll raha hu
2	nan
3	parth just nailed the role of nawabas he recei...
4	i love the series it is amazing parth acting i...
...	...
9896	not seen yet
9897	entertainment is just for entertainment
9898	average
9899	time pass series
9900	please avoid this one

9901 rows × 1 columns

In [26]:

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

Out[26]:

	transcript
6696	rangbaaz phirse is story of a gangster from ra...
2244	outstanding writing brilliant performance over...
5165	unthinkable twists and incredible ending
9290	whatever red chillies touchesturns to zero now...
6924	the concept and the plot of the show may be co...
3346	wat a brilliant performance by all the cast sp...

In [27]:

```
randomcheck=data_clean_df.loc[2694]
randomcheck
#we see that the emoji has been removed too , along with other text cleaning.
```

Out[27]:

```
transcript    awesome seriescant wait to see next season
Name: 2694, dtype: object
```

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...

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.

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

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 create 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 [30]:

```
print(stopwords)
```

```
{'if', 'else', 'at', 'their', 'below', 'but', 'all', "they're", "that's", 'm
ore', 'hence', 'be', 'is', 'some', "he'd", 'after', "hadn't", 'into', "we'r
e", "how's", "you'll", 'my', 'so', 'does', 'r', 'other', 'you', 'above', 'fe
w', 'any', 'such', 'both', 'this', 'where', 'as', "they'll", 'under', 'has',
'theirs', 'your', 'ever', 'shall', 'only', 'www', 'had', 'no', 'on', 'by',
'until', 'http', "here's", "couldn't", "they'd", 'whom', 'off', "who's", 'u
p', 'our', "where's", "she's", 'those', 'there', 'to', 'we', 'were', "has
n't", 'that', 'however', "shouldn't", 'ought', 'i', 'out', "he's", "i'd", 'h
er', 'being', 'here', 'who', 'most', "didn't", 'too', 'can', "she'd", 'bee
n', 'themselves', 'also', 'each', "doesn't", 'him', 'am', 'why', "there's",
'like', "i'm", 'hers', 'an', 'when', "won't", 'how', 'herself', 'doing', "ar
en't", "when's", 'cannot', 'me', 'own', 'would', "you've", 'k', 'com', 'th
e', 'further', 'myself', 'in', 'over', 'yourselves', "we'll", "isn't", 'an
d', 'having', 'a', 'for', 'very', 'during', "it's", 'was', "what's", 'nor',
'it', 'did', "shan't", 'itself', 'ours', 'about', 'should', 'down', 'otherwi
se', 'not', 'she', 'same', 'these', "we'd", 'once', 'are', 'its', "haven't",
'yours', 'or', 'them', "weren't", 'ourselves', 'could', 'do', 'between', "ca
n't", 'from', "i've", 'again', "don't", 'therefore', "he'll", 'his', 'than',
'then', "why's", "mustn't", 'just', "let's", "wasn't", 'have', 'what', 'whic
h', 'himself', 'while', 'get', "we've", 'they', 'with', "you'd", 'because',
"i'll", 'through', 'yourself', 'against', 'since', 'he', "she'll", 'before',
"wouldn't", 'of', "you're", "they've"}
```

In [31]:

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

Out[31]:

```
'nan reviews of mai hero boll raha hu nan parth just nailed the role of na
wabas he received in compliment that he is is not even one percent to what
nawab isits truejust loved the dialogues n action though this is not my ge
nreloved laila lala manaswi n mutazzgood show to watch in lockdown i love
the series it is amazing parth acting is superb i just want to ask parthho
w so perfectparth and intense acting goes hand in handso powerpacktop notc
h performancei couldnt take my eyes off not even for a second terrific per
formancea must watch show amazing show awesome story loved the characters
everyone did really amazing and nawab was like fire his aura confidence ac
ting skills were just mindblowing parth did really really well he never st
op surprising ushis eyes were so passionate so expression as always our pa
rth was superb mindblowing i was just shocked to see this avtar of himcant
just recognise it was our parth loved the bonding of nawab and mumtaz the
friendship was really amazing nawab and jagan loved them and also nawab nd
lailas bond was firee also loved how manasvi as a true partner all the act
ors did really amazing work loved each and everything about the show every
thingalso want season of mhbrhoverall the show is bomb its a freaking mast
erpiece very much worth our waiting wonderful series with power cast huge
```

In [32]:

```
# 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', 11059), ('and', 7161), ('of', 5785), ('is', 5575), ('a', 5116),
('series', 4914), ('to', 4860), ('i', 3661), ('it', 3572), ('this', 3255),
('in', 3041), ('for', 2394), ('watch', 2319), ('best', 1819), ('one', 176
9), ('all', 1700), ('good', 1641), ('show', 1638), ('was', 1622), ('you',
1605), ('with', 1570), ('its', 1462), ('very', 1453), ('web', 1433), ('sto
ry', 1424), ('are', 1387), ('acting', 1376), ('season', 1344), ('have', 13
30), ('on', 1257), ('as', 1256), ('by', 1246), ('that', 1243), ('but', 116
9), ('must', 1125), ('amazing', 1122), ('just', 1110), ('not', 1059), ('gr
eat', 1027), ('has', 1008), ('so', 976), ('like', 856), ('his', 855), ('fa
mily', 835), ('manoj', 818), ('awesome', 794), ('be', 793), ('really', 75
2), ('watching', 741), ('well', 727), ('watched', 725), ('indian', 725),
('an', 724), ('will', 697), ('thriller', 692), ('man', 689), ('from', 68
1), ('they', 669), ('more', 667), ('kay', 653), ('every', 652), ('loved',
647), ('what', 629), ('he', 621), ('time', 595), ('actors', 581), ('my', 5
67), ('which', 557), ('their', 547), ('ever', 541), ('love', 528), ('als
o', 528), ('at', 526), ('episode', 522), ('menon', 512), ('cast', 505),
('work', 502), ('waiting', 501), ('can', 496), ('such', 494), ('performanc
e', 480), ('no', 477), ('who', 472), ('direction', 467), ('some', 464),
('superb', 460), ('after', 446), ('episodes', 443), ('about', 442), ('don
```

In [33]:

```
# 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 > 2400]
add_stop_words
```

Out[33]:

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

In [34]:

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

```
frozenset({'eg', 'go', 'after', 'front', 'my', 'perhaps', 'few', 'therein',
'un', 'find', 'afterwards', 'full', 'your', 'ever', 'only', 'no', 'on', 'b
y', 'one', 'take', 'every', 'there', 'we', 'much', 'least', 'hasnt', 'togeth
er', 'most', 'thereby', 'last', 'also', 'each', 'am', 'twelve', 'whoever',
'though', 'seeming', 'herself', 'cannot', 'fifty', 'via', 'meanwhile', 'myse
lf', 'twenty', 'in', 'yourselves', 'and', 'whenever', 'seemed', 'very', 'd
e', 'was', 'become', 'ours', 'others', 'thin', 'same', 'seems', 'wherever',
'thru', 'are', 'made', 'yours', 'them', 'will', 'do', 'from', 'see', 'nothin
g', 'without', 'therefore', 'although', 'first', 'still', 'get', 'they', 'si
ncere', 'yourself', 'indeed', 'third', 'he', 'move', 'detail', 'if', 'else',
'at', 'fill', 'their', 'yet', 'hence', 'whole', 'anyone', 'into', 'everythin
g', 'alone', 'whether', 'next', 'such', 'this', 'many', 'con', 'nobody', 'no
ne', 'thence', 'thereupon', 'had', 'often', 'everyone', 'off', 'now', 'our',
'those', 'were', 'herein', 'among', 'anyhow', 'amoungst', 'here', 'who', 'to
o', 'can', 'former', 'namely', 'except', 'describe', 'two', 'an', 'hers', 'h
ow', 'me', 'formerly', 'around', 'own', 'whereafter', 'ten', 'neither', 'to
p', 'behind', 'interest', 'nine', 'nor', 'hundred', 'about', 'should', 'sh
e', 'mostly', 'once', 'forty', 'became', 'than', 'six', 'hereby', 'while',
'empty', 'besides', 'found', 'below', 'but', 'all', 'within', 'more', 'be',
'another', 'five', 'bottom', 'so', 'you', 'above', 'thereafter', 'four', 'bo
th', 'beside', 'where', 'as', 'under', 'has', 'always', 'until', 'whom', 'et
c', 'anyway', 'three', 'may', 'however', 'system', 'enough', 'latterly', 'le
ss', 'being', 'back', 'done', 'nevertheless', 'been', 'themselves', 'amoun
t', 'bill', 'across', 'him', 'ie', 'elsewhere', 'why', 'thick', 'whence', 'w
hereupon', 'someone', 'latter', 'would', 'sometimes', 'further', 'mill', 'ov
er', 'fire', 'might', 'due', 'thus', 'for', 'during', 'side', 'it', 'otherwi
se', 'couldnt', 'its', 'somewhere', 'or', 'mine', 'ourselves', 'could', 'bey
ond', 'must', 'becomes', 'wherein', 'fifteen', 'his', 'then', 'have', 'wha
t', 'well', 'himself', 'cant', 'with', 'because', 'co', 'even', 'call', 'now
here', 'show', 'give', 'whatever', 'against', 'since', 'before', 'toward',
'us', 'keep', 'somehow', 'upon', 'is', 'eleven', 'some', 'along', 'beforehan
d', 'other', 'any', 'whose', 'several', 'ltd', 'something', 'everywhere', 'a
lmost', 'inc', 'up', 'either', 'to', 'that', 'of', 'towards', 'i', 'out', 'e
ight', 'her', 'amongst', 'noone', 'anything', 'whereby', 'seem', 'becoming',
'whither', 'when', 'sixty', 'rather', 'anywhere', 'please', 'hereupon', 'som
etime', 'a', 're', 'throughout', 'itself', 'down', 'not', 'part', 'these',
'between', 'put', 'again', 'whereas', 'moreover', 'hereafter', 'cry', 'whic
h', 'already', 'through', 'name', 'never', 'per', 'onto', 'serious', 'the'})
```

In [38]:

```
# 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()
```

Sentiment WorldCloud

In [40]:

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

```
['eg', 'go', 'after', 'front', 'my', 'perhaps', 'few', 'therein', 'un', 'fin
d', 'afterwards', 'full', 'your', 'ever', 'only', 'no', 'on', 'by', 'one',
'take', 'every', 'there', 'we', 'much', 'least', 'hasnt', 'together', 'mos
t', 'thereby', 'last', 'also', 'each', 'am', 'twelve', 'whoever', 'though',
'seeming', 'herself', 'cannot', 'fifty', 'via', 'meanwhile', 'myself', 'twen
ty', 'in', 'yourselves', 'and', 'whenever', 'seemed', 'very', 'de', 'was',
'become', 'ours', 'others', 'thin', 'same', 'seems', 'wherever', 'thru', 'ar
e', 'made', 'yours', 'them', 'will', 'do', 'from', 'see', 'nothing', 'withou
t', 'therefore', 'although', 'first', 'still', 'get', 'they', 'sincere', 'yo
urself', 'indeed', 'third', 'he', 'move', 'detail', 'if', 'else', 'at', 'fil
l', 'their', 'yet', 'hence', 'whole', 'anyone', 'into', 'everything', 'alon
e', 'whether', 'next', 'such', 'this', 'many', 'con', 'nobody', 'none', 'the
nce', 'thereupon', 'had', 'often', 'everyone', 'off', 'now', 'our', 'those',
'were', 'herein', 'among', 'anyhow', 'amoungst', 'here', 'who', 'too', 'ca
n', 'former', 'namely', 'except', 'describe', 'two', 'an', 'hers', 'how', 'm
e', 'formerly', 'around', 'own', 'whereafter', 'ten', 'neither', 'top', 'beh
ind', 'interest', 'nine', 'nor', 'hundred', 'about', 'should', 'she', 'mostl
y', 'once', 'forty', 'became', 'than', 'six', 'hereby', 'while', 'empty', 'b
esides', 'found', 'below', 'but', 'all', 'within', 'more', 'be', 'another',
'five', 'bottom', 'so', 'you', 'above', 'thereafter', 'four', 'both', 'besid
e', 'where', 'as', 'under', 'has', 'always', 'until', 'whom', 'etc', 'anywa
y', 'three', 'may', 'however', 'system', 'enough', 'latterly', 'less', 'bein
g', 'back', 'done', 'nevertheless', 'been', 'themselves', 'amount', 'bill',
'across', 'him', 'ie', 'elsewhere', 'why', 'thick', 'whence', 'whereupon',
'someone', 'latter', 'would', 'sometimes', 'further', 'mill', 'over', 'fir
e', 'might', 'due', 'thus', 'for', 'during', 'side', 'it', 'otherwise', 'cou
ldnt', 'its', 'somewhere', 'or', 'mine', 'ourselves', 'could', 'beyond', 'mu
st', 'becomes', 'wherein', 'fifteen', 'his', 'then', 'have', 'what', 'well',
'himself', 'cant', 'with', 'because', 'co', 'even', 'call', 'nowhere', 'sho
w', 'give', 'whatever', 'against', 'since', 'before', 'toward', 'us', 'kee
p', 'somehow', 'upon', 'is', 'eleven', 'some', 'along', 'beforehand', 'othe
r', 'any', 'whose', 'several', 'ltd', 'something', 'everywhere', 'almost',
'inc', 'up', 'either', 'to', 'that', 'of', 'towards', 'i', 'out', 'eight',
'her', 'amongst', 'noone', 'anything', 'whereby', 'seem', 'becoming', 'whith
er', 'when', 'sixty', 'rather', 'anywhere', 'please', 'hereupon', 'sometim
e', 'a', 're', 'throughout', 'itself', 'down', 'not', 'part', 'these', 'betw
een', 'put', 'again', 'whereas', 'moreover', 'hereafter', 'cry', 'which', 'a
lready', 'through', 'name', 'never', 'per', 'onto', 'serious', 'the', 'sho
w', 'season', 'one', 'season', 'watch', 'story', 'bajpayee', 'webserie', 'we
bseries', 'bajpai', 'manoj', 'episode', 'review', 'actor', 'actors', 'the',
'and', 'of', 'is', 'a', 'series', 'to', 'i', 'it', 'this', 'in', 'if', 'els
e', 'at', 'their', 'below', 'but', 'all', "they're", "that's", 'more', 'henc
e', 'be', 'is', 'some', "he'd", 'after', "hadn't", 'into', "we're", "how's",
"you'll", 'my', 'so', 'does', 'r', 'other', 'you', 'above', 'few', 'any', 's
uch', 'both', 'this', 'where', 'as', "they'll", 'under', 'has', 'theirs', 'y
our', 'ever', 'shall', 'only', 'www', 'had', 'no', 'on', 'by', 'until', 'htt
p', "here's", "couldn't", "they'd", 'whom', 'off', "who's", 'up', 'our', "wh
ere's", "she's", 'those', 'there', 'to', 'we', 'were', "hasn't", 'that', 'ho
wever', "shouldn't", 'ought', 'i', 'out', "he's", "i'd", 'her', 'being', 'he
re', 'who', 'most', "didn't", 'too', 'can', "she'd", 'been', 'themselves',
'also', 'each', "doesn't", 'him', 'am', 'why', "there's", 'like', "i'm", 'he
rs', 'an', 'when', "won't", 'how', 'herself', 'doing', "aren't", "when's",
'cannot', 'me', 'own', 'would', "you've", 'k', 'com', 'the', 'further', 'mys
elf', 'in', 'over', 'yourselves', "we'll", "isn't", 'and', 'having', 'a', 'f
or', 'very', 'during', "it's", 'was', "what's", 'nor', 'it', 'did', "sha
n't", 'itself', 'ours', 'about', 'should', 'down', 'otherwise', 'not', 'sh
```

```
e', 'same', 'these', "we'd", 'once', 'are', 'its', "haven't", 'yours', 'or',
'them', "weren't", 'ourselves', 'could', 'do', 'between', "can't", 'from',
"i've", 'again', "don't", 'therefore', "he'll", 'his', 'than', 'then', "wh
y's", "mustn't", 'just', "let's", "wasn't", 'have', 'what', 'which', 'himsel
f', 'while', 'get', "we've", 'they', 'with', "you'd", 'because', "i'll", 'th
rough', 'yourself', 'against', 'since', 'he', "she'll", 'before', "would
n't", 'of', "you're", "they've"]
```

Findings

We can clearly see that the word cloud has major chunk of positve reviews(roughly 75%) , some negative reviews (roughly 15%), 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 interative 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 [41]:

```
# 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[41]:

	transcript	polarity	subjectivity
0	nan	0.000	0.000
1	reviews of mai hero boll raha hu	0.000	0.000
2	nan	0.000	0.000
3	parth just nailed the role of nawabas he recei...	0.400	0.450
4	i love the series it is amazing parth acting i...	0.525	0.625
...	...	...	...
9896	not seen yet	0.000	0.000
9897	entertainment is just for entertainment	0.000	0.000
9898	average	-0.150	0.400
9899	time pass series	0.000	0.000
9900	please avoid this one	0.000	0.000

9901 rows × 3 columns

In [42]:

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

Out[42]:

	transcript	polarity	subjectivity
2206	i watched it in one go was very interesting an...	0.575000	0.825000
2240	although manoj bajpayee has played his talent ...	0.511111	0.288889
2588	series is so good but end is so disappointing ...	0.050000	0.650000
2241	no words to describe the series amazing	0.600000	0.900000
2388	amazing serieskoodos to manoj sir and to the crew	0.600000	0.900000

In [43]:

```
#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[43]:

	transcript	polarity	subjectivity	Analysis
0	nan	0.000	0.000	neutral
1	reviews of mai hero boll raha hu	0.000	0.000	neutral
2	nan	0.000	0.000	neutral
3	parth just nailed the role of nawabas he recei...	0.400	0.450	positive
4	i love the series it is amazing parth acting i...	0.525	0.625	positive
...	...	...	...	...
9896	not seen yet	0.000	0.000	neutral
9897	entertainment is just for entertainment	0.000	0.000	neutral
9898	average	-0.150	0.400	negative
9899	time pass series	0.000	0.000	neutral
9900	please avoid this one	0.000	0.000	neutral

9901 rows × 4 columns

In [45]:

```
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 folloswing 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(Actio
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 Serie
s(Action Genre) :

No of Negative Reviews from our Total DataSet(around 10k) -> 831
No of Positive Reviews from our Total DataSet(around 10k) -> 8163
No of  Neutral Reviews from our Total DataSet(around 10k) -> 907

Percentage of Negative Reviews -> 8.393091606908392 %
Percentage of Positive Reviews -> 82.44621755378245 %
Percentage of Neutral  Reviews -> 9.16069083930916  %
```

Sentiment Findings:

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

```
- No of Negative Reviews from our Total DataSet(around 10k) -> 831
- No of Positive Reviews from our Total DataSet(around 10k) -> 8163
- No of Neutral  Reviews from our Total DataSet(around 10k) -> 907


- Percentage of Negative Reviews -> 8.393091606908392 %
- Percentage of Positive Reviews -> 82.44621755378245 %
- Percentage of Neutral  Reviews -> 9.16069083930916  %


This also confirms our vague analysis that we did using just the wordcloud sentime
nts.
```

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

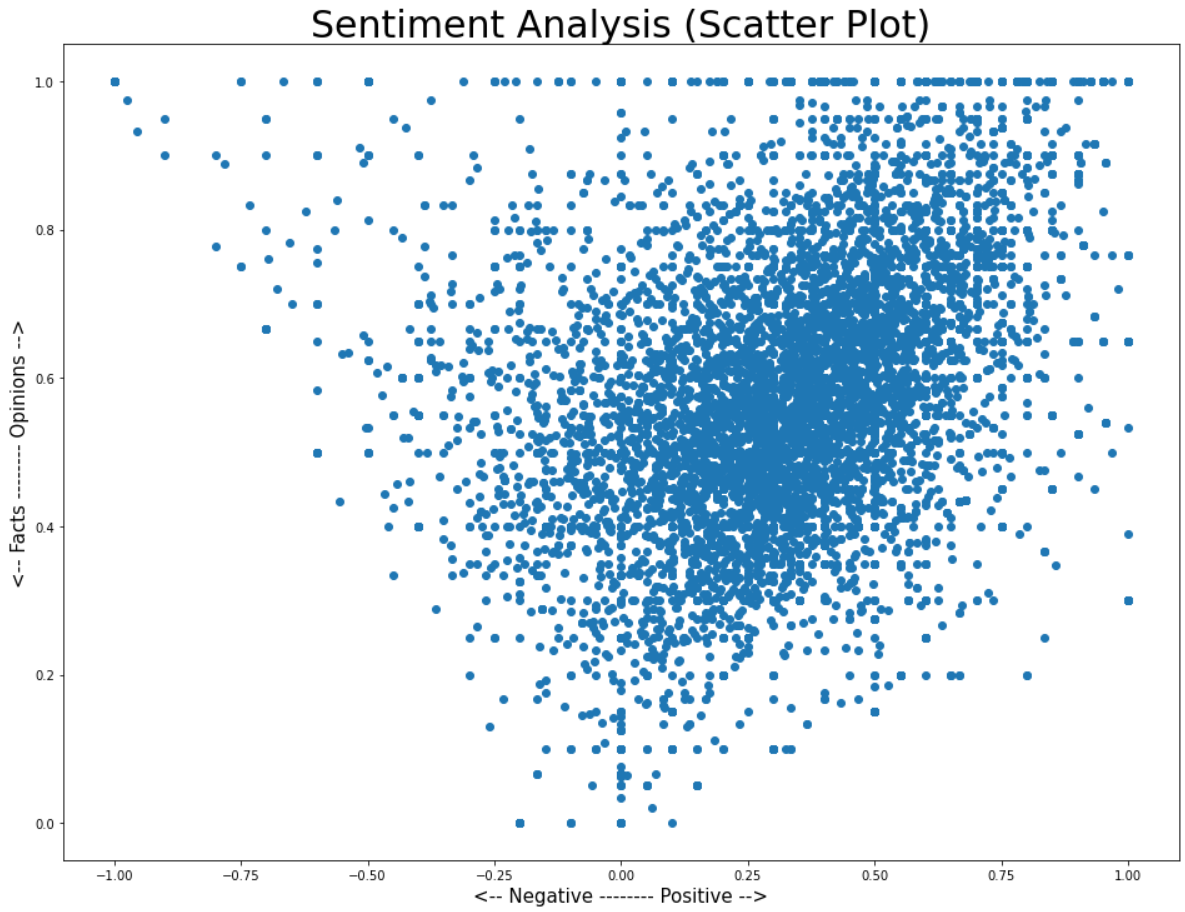
Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When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

Other benefits of data visualization include the following:

- 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.

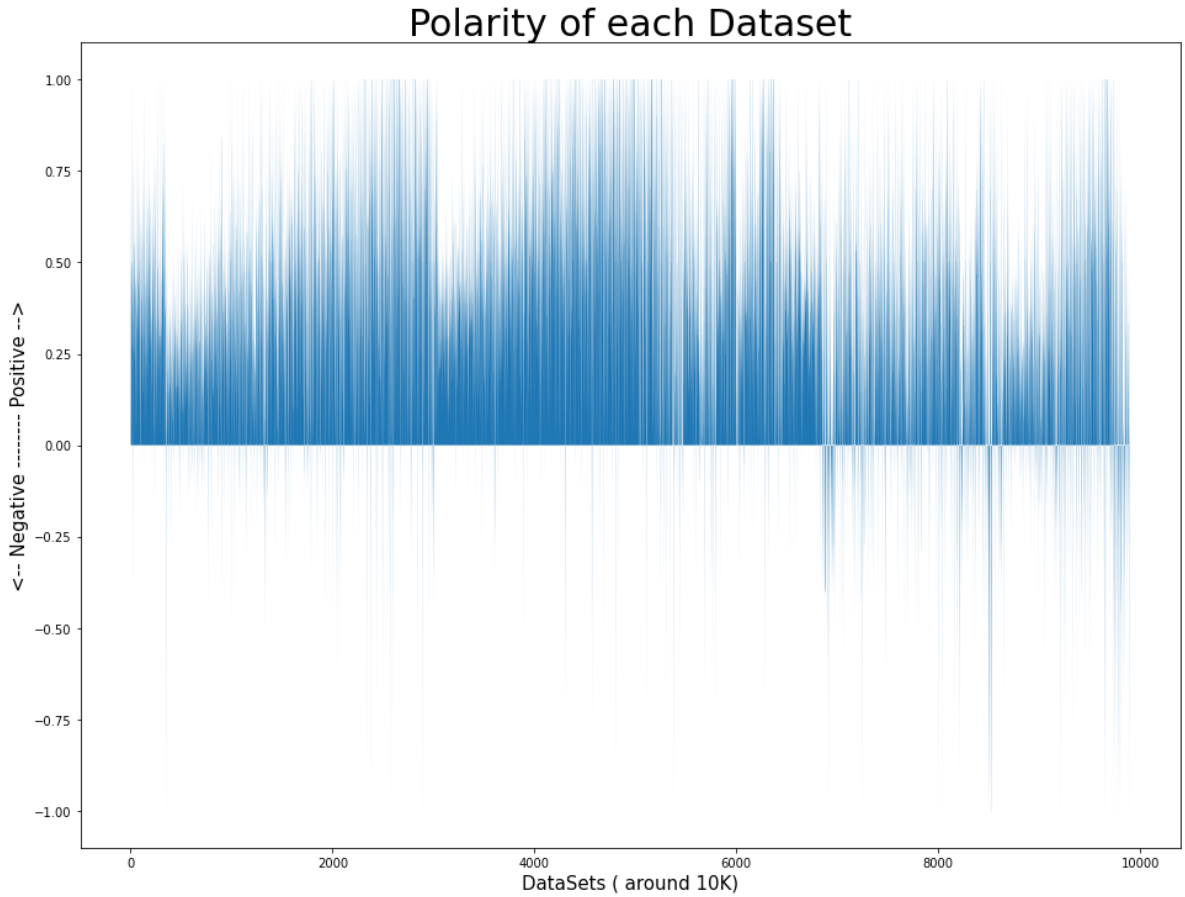
In [46]:

```
# 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()
```



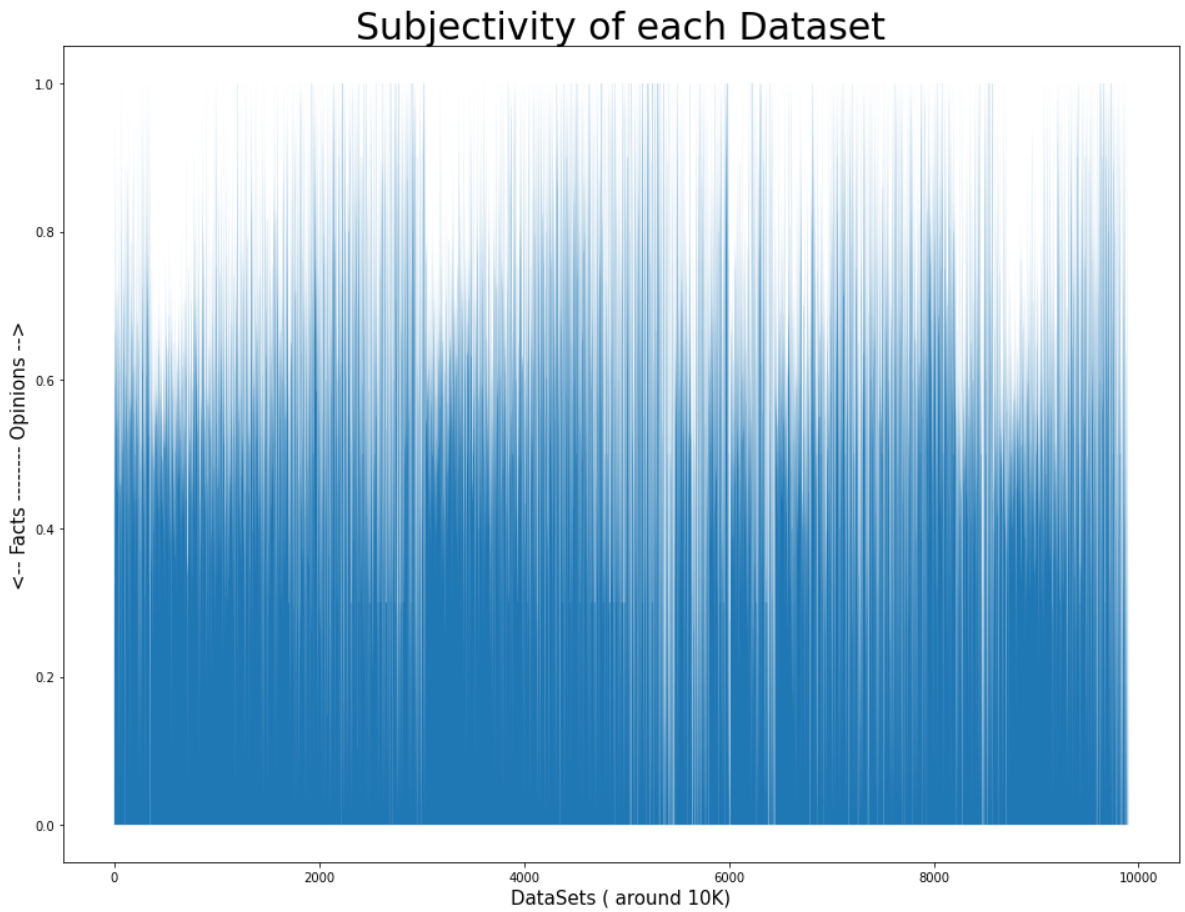
In [47]:

```
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()
```



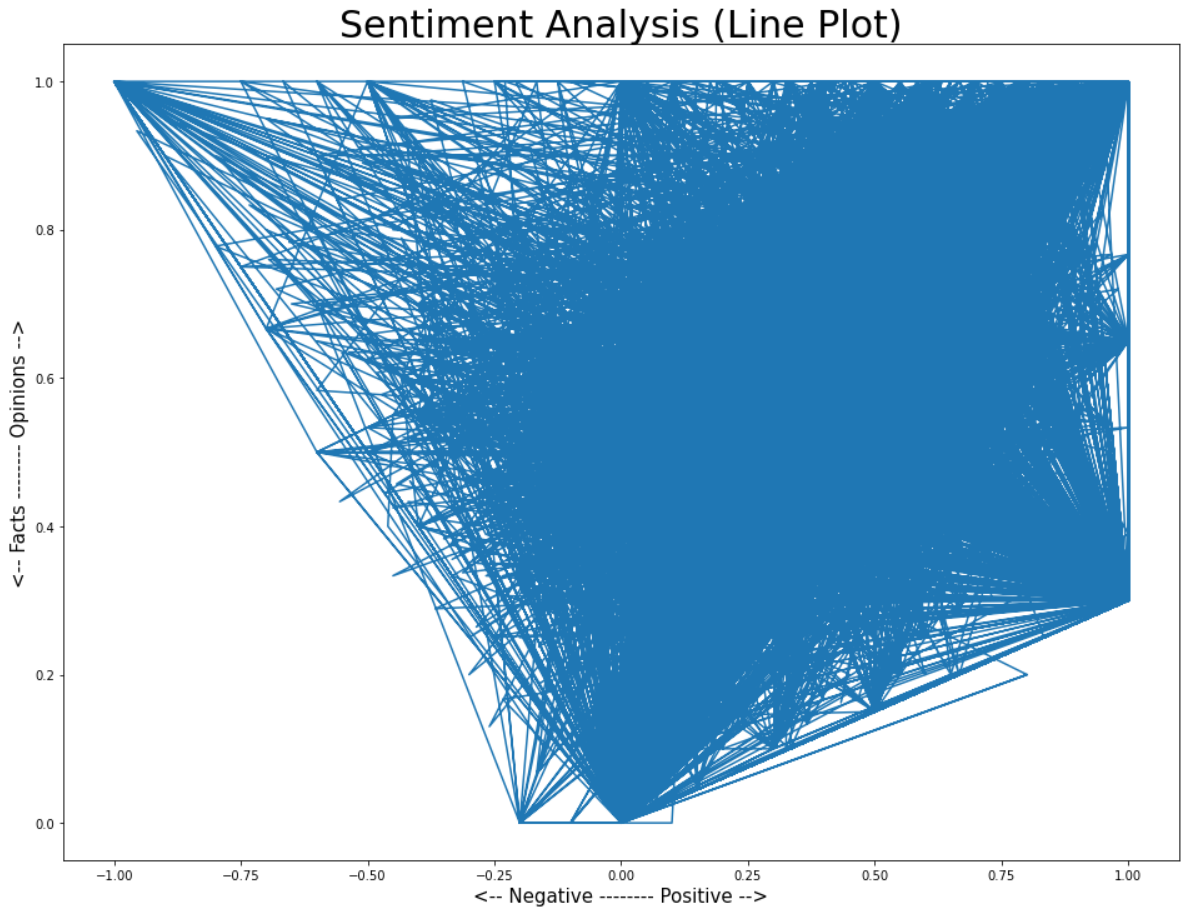
In [48]:

```
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()
```



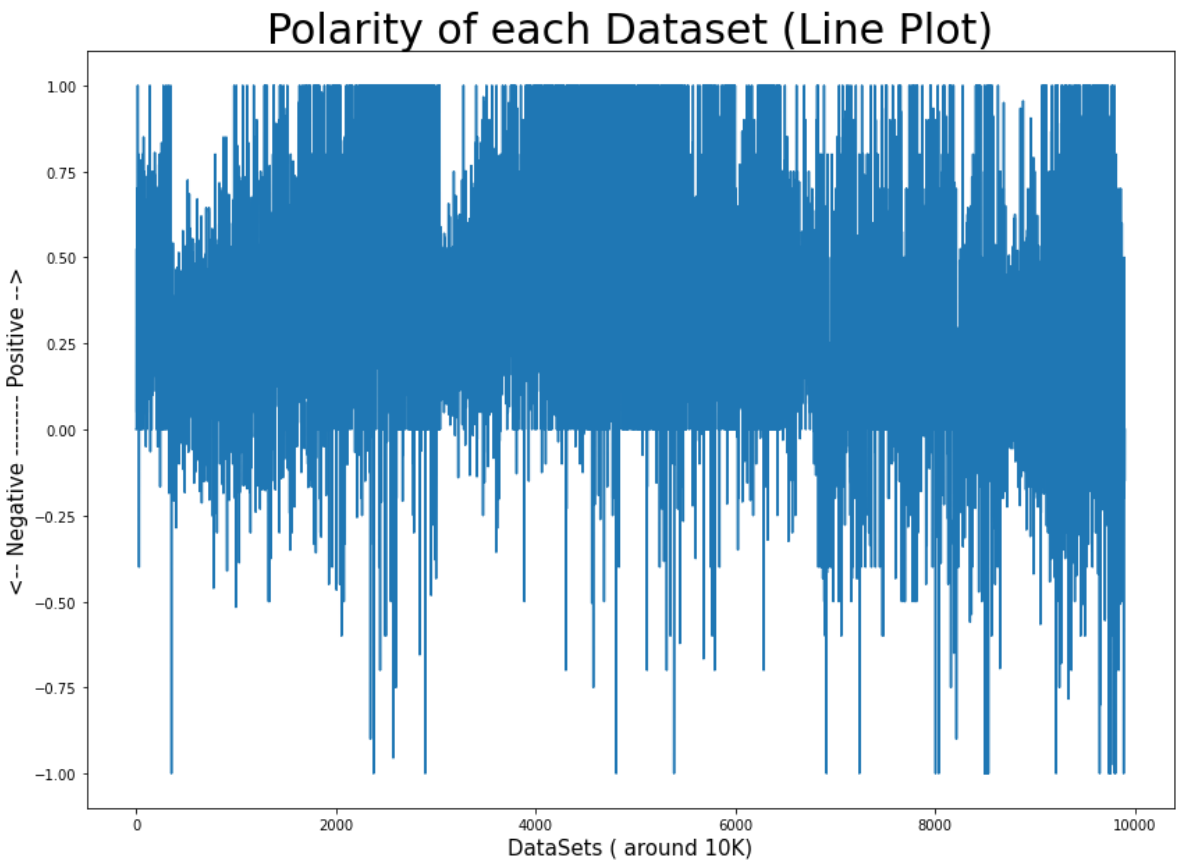
In [49]:

```
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()
```



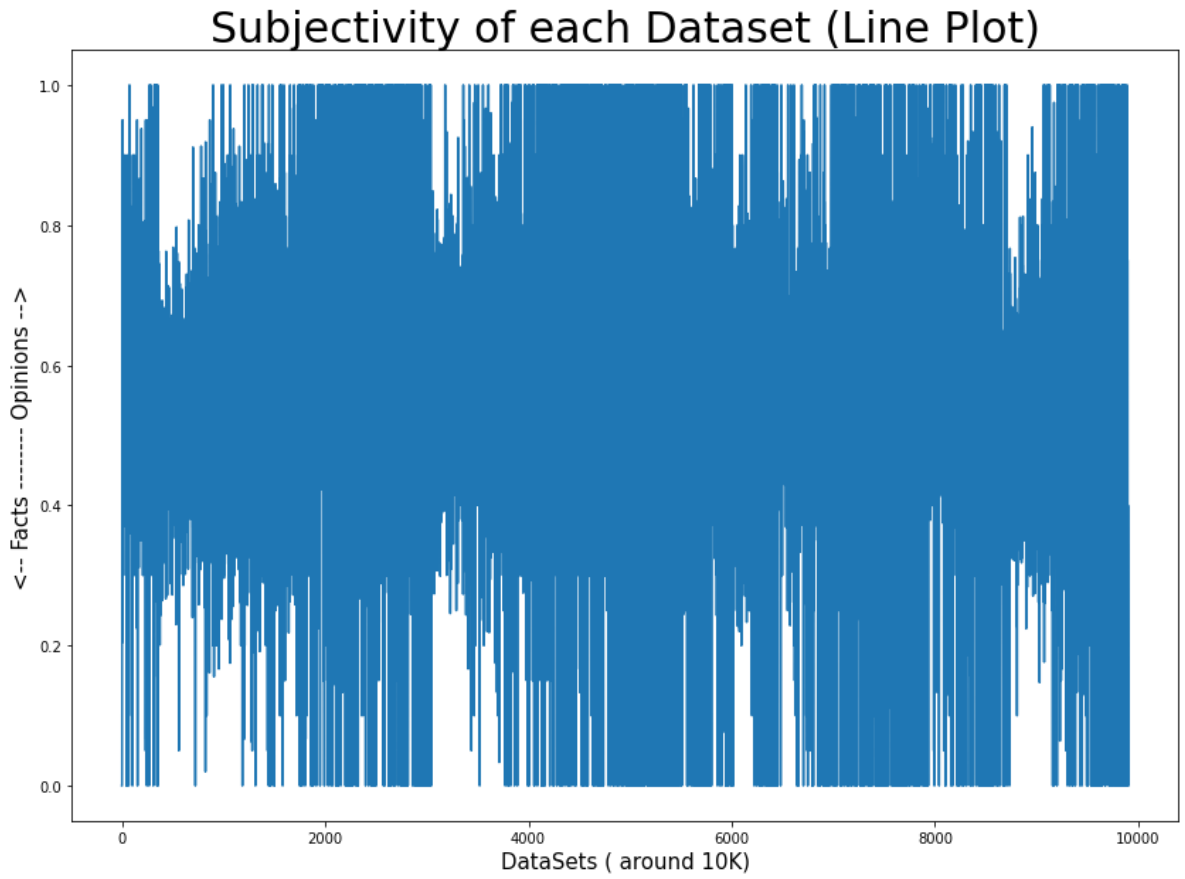
In [50]:

```
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()
```



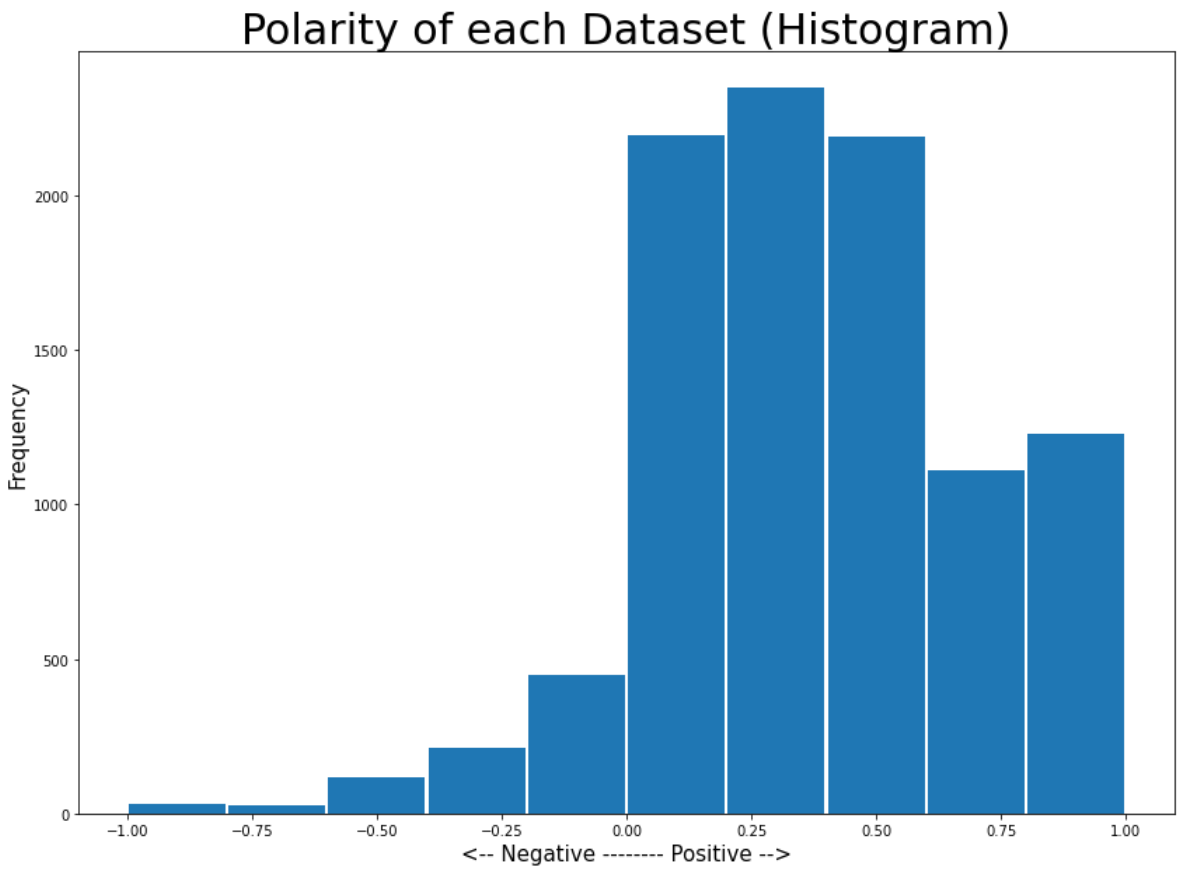
In [51]:

```
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()
```



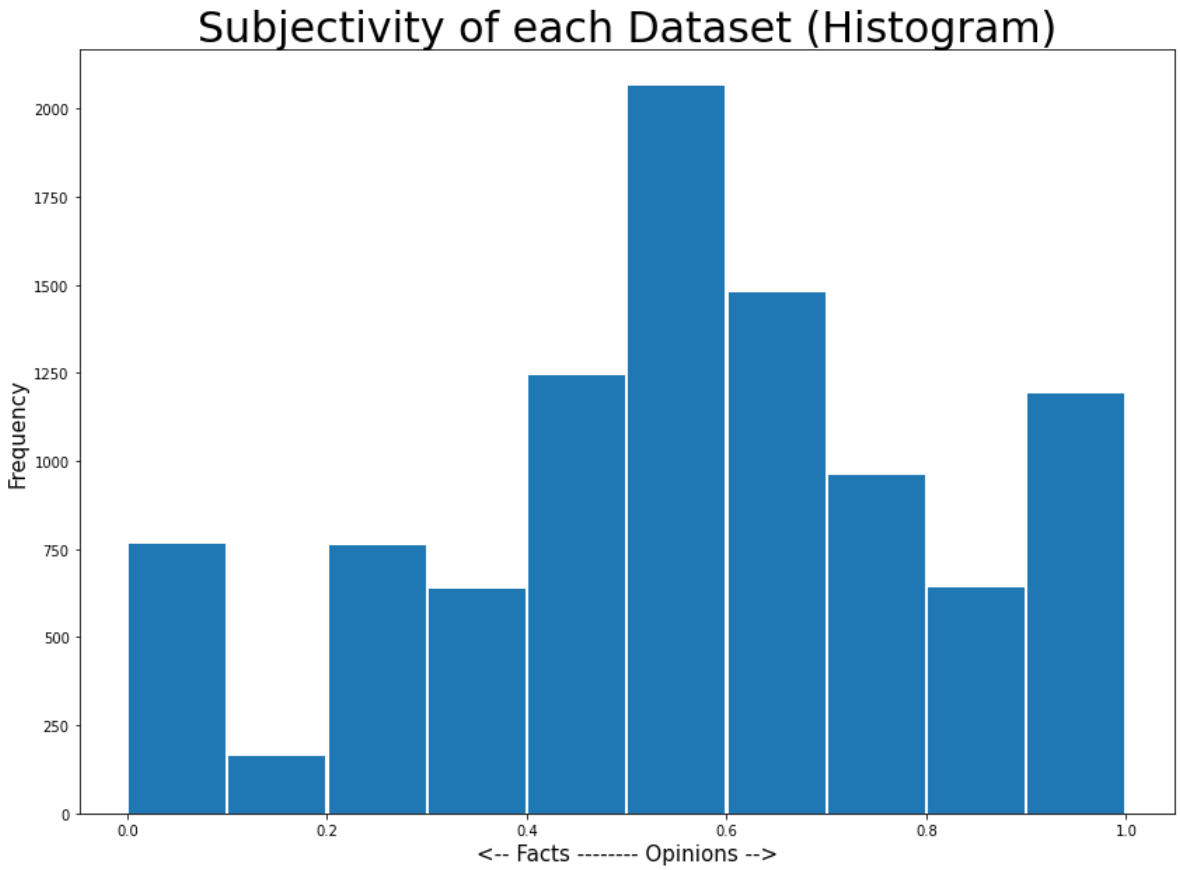
In [52]:

```
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 [53]:

```
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 [54]:

```
data_clean_df
```

Out[54]:

	transcript	polarity	subjectivity	Analysis
0	nan	0.000	0.000	neutral
1	reviews of mai hero boll raha hu	0.000	0.000	neutral
2	nan	0.000	0.000	neutral
3	parth just nailed the role of nawabas he recei...	0.400	0.450	positive
4	i love the series it is amazing parth acting i...	0.525	0.625	positive
...	...	...	...	...
9896	not seen yet	0.000	0.000	neutral
9897	entertainment is just for entertainment	0.000	0.000	neutral
9898	average	-0.150	0.400	negative
9899	time pass series	0.000	0.000	neutral
9900	please avoid this one	0.000	0.000	neutral

9901 rows × 4 columns

In [55]:

```
#Creating a new DataFrame with only Positve 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 [56]:

```
positive_df
```

Out[56]:

	transcript	polarity	subjectivity	Analysis
3	parth just nailed the role of nawabas he recei...	0.400000	0.450000	positive
4	i love the series it is amazing parth acting i...	0.525000	0.625000	positive
5	i just want to ask parthhow so perfectparth an...	0.050000	0.500000	positive
6	amazing show awesome story loved the character...	0.483333	0.705556	positive
7	wonderful series with power cast huge applauds...	0.700000	0.950000	positive
...	...	...	...	...
9881	not bad	0.350000	0.666667	positive
9883	got to be more realistic	0.333333	0.416667	positive
9884	bard of blood is bold	0.333333	0.666667	positive
9893	same old bollywood wine	0.050000	0.162500	positive
9894	need to make more gripping	0.500000	0.750000	positive

8163 rows × 4 columns

In [57]:

```
# Python program to generate WordCloud for POSITVE 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
                     'episode','review','actor','actors']
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 POSITVE SENTIMENTS\n',fontsize=30)

plt.show()
```

In [58]:

```
#Creating a new DataFrame with only Negatve 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 [59]:

```
negative_df
```

Out[59]:

	transcript	polarity	subjectivity	Analysis
28	i am just literally shocked after sewing parth...	-0.400000	0.900	negative
97	a bang on action thrillerparth is literally th...	-0.050000	0.100	negative
141	who said it is a showseries it is a freaking b...	-0.064583	0.525	negative
214	a power packed series each and everyone justif...	-0.050000	0.690	negative
232	i love you parth and your acting skills i wish...	-0.033333	0.500	negative
...	...	...	...	...
9879	average after watching sacred games	-0.150000	0.400	negative
9880	late latif	-0.300000	0.600	negative
9886	waste ot time	-0.200000	0.000	negative
9888	terrible show	-1.000000	1.000	negative
9898	average	-0.150000	0.400	negative

831 rows × 4 columns

In [60]:

```
# Python program to generate WordCloud for NEGATVE 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',
                     'bajpai','manoj','episode','review','actor','actors']
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 NEGATVE SENTIMENTS\n',fontsize=30)

plt.show()
```

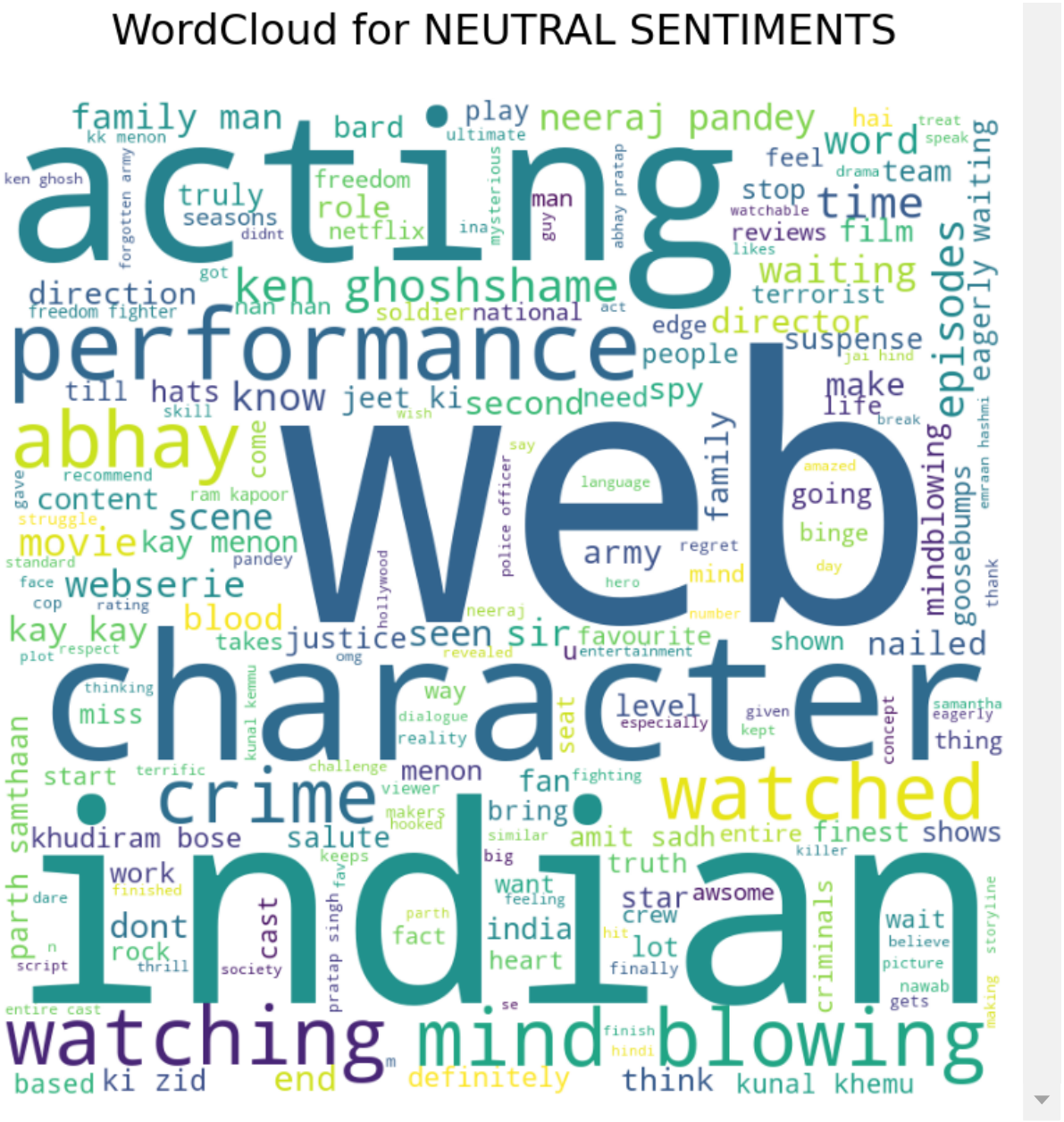
WordCloud for NEGATVE SENTIMENTS

In [61]:

```
#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 [62]:

```
# 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',
                     'story','masterpiece','bajpayee','bajpai','manoj','episode','review','
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()
```



Additonal Information

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

In [66]:

```
# 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', 9531), ('and', 6252), ('of', 4875), ('is', 4795), ('series', 442
5), ('a', 4300), ('to', 4029), ('i', 3215), ('it', 3156), ('this', 2768),
('in', 2613), ('for', 2051), ('watch', 1988), ('best', 1813), ('one', 162
0), ('good', 1589), ('all', 1501), ('was', 1441), ('show', 1392), ('with',
1343), ('web', 1328), ('very', 1311), ('you', 1297), ('its', 1282), ('acti
ng', 1253), ('story', 1230), ('season', 1188), ('are', 1160), ('have', 115
9), ('by', 1121), ('amazing', 1117), ('as', 1055), ('that', 1040), ('grea
t', 1009), ('on', 999), ('but', 974), ('must', 956), ('just', 949), ('ha
s', 873), ('so', 861), ('not', 819), ('awesome', 794), ('manoj', 750), ('h
is', 739), ('really', 708), ('family', 704), ('like', 701), ('watched', 65
5), ('be', 653), ('well', 649), ('loved', 646), ('watching', 645), ('mor
e', 637), ('thriller', 636), ('an', 634), ('indian', 616), ('kay', 608),
('every', 597), ('will', 596), ('man', 584), ('from', 571), ('actors', 55
2), ('what', 530), ('he', 519), ('love', 519), ('they', 515), ('ever', 50
1), ('work', 480), ('their', 478), ('menon', 477), ('my', 476), ('cast', 4
76), ('which', 471), ('also', 467), ('episode', 462), ('superb', 459), ('w
aiting', 457), ('time', 449), ('performance', 446), ('at', 440), ('such',
436), ('done', 403), ('can', 401), ('action', 399), ('direction', 399),
('excellent', 385), ('episodes', 379), ('who', 375), ('much', 374), ('som
```

In [67]:

```
# 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', 1077), ('and', 662), ('of', 661), ('to', 620), ('is', 585), ('a',
579), ('in', 331), ('this', 321), ('series', 303), ('i', 302), ('it', 29
6), ('for', 217), ('you', 206), ('not', 193), ('are', 184), ('with', 173),
('but', 171), ('that', 164), ('show', 164), ('was', 150), ('as', 149), ('o
n', 148), ('story', 145), ('watch', 144), ('very', 137), ('all', 132), ('h
ave', 129), ('they', 126), ('its', 125), ('time', 125), ('be', 111), ('lik
e', 109), ('family', 102), ('so', 100), ('bad', 96), ('just', 94), ('don
t', 94), ('has', 91), ('no', 90), ('waste', 89), ('acting', 87), ('from',
87), ('one', 85), ('his', 82), ('man', 82), ('who', 76), ('how', 75), ('b
y', 74), ('season', 73), ('or', 73), ('indian', 73), ('can', 73), ('he', 7
2), ('some', 71), ('will', 69), ('which', 69), ('watching', 68), ('at', 6
7), ('after', 63), ('worst', 63), ('an', 62), ('criminal', 62), ('poor', 6
1), ('if', 61), ('web', 59), ('there', 59), ('too', 56), ('people', 56),
('only', 55), ('well', 55), ('direction', 55), ('about', 54), ('your', 5
3), ('what', 52), ('good', 52), ('should', 51), ('up', 50), ('know', 49),
('their', 49), ('do', 47), ('we', 47), ('other', 46), ('boring', 46), ('ev
en', 45), ('my', 45), ('netflix', 45), ('really', 44), ('watched', 44),
('khudiram', 44), ('must', 43), ('any', 42), ('out', 42), ('episodes', 4
2), ('something', 41), ('them', 41), ('manoj', 40), ('episode', 39), ('bee
```

In [69]:

```
# 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', 451), ('of', 249), ('and', 247), ('a', 237), ('to', 211), ('is',
195), ('watch', 187), ('series', 186), ('this', 166), ('i', 144), ('for',
126), ('must', 126), ('it', 120), ('on', 110), ('you', 102), ('in', 97),
('season', 83), ('show', 82), ('just', 67), ('all', 67), ('one', 64), ('it
s', 55), ('with', 54), ('as', 52), ('by', 51), ('story', 49), ('what', 4
7), ('not', 47), ('like', 46), ('my', 46), ('web', 46), ('has', 44), ('ar
e', 43), ('have', 42), ('that', 39), ('thriller', 38), ('acting', 36), ('i
ndian', 36), ('mind', 35), ('his', 34), ('shame', 33), ('will', 32), ('blo
wing', 31), ('was', 31), ('kay', 31), ('abhay', 31), ('ken', 31), ('he', 3
0), ('masterpiece', 30), ('be', 29), ('waiting', 29), ('family', 29), ('a
n', 28), ('they', 28), ('manoj', 28), ('watching', 28), ('no', 27), ('suc
h', 27), ('watched', 26), ('menon', 26), ('never', 25), ('cant', 25), ('gh
oshshame', 25), ('always', 24), ('but', 24), ('next', 24), ('up', 24), ('c
haracter', 23), ('me', 23), ('man', 23), ('should', 23), ('also', 23), ('a
bout', 23), ('from', 23), ('after', 23), ('well', 23), ('neeraj', 23), ('w
e', 23), ('pandey', 23), ('crime', 23), ('can', 22), ('every', 22), ('epis
odes', 22), ('performance', 21), ('again', 21), ('episode', 21), ('time',
21), ('who', 21), ('nan', 20), ('their', 20), ('us', 19), ('at', 19), ('so
me', 19), ('army', 19), ('sir', 18), ('there', 18), ('which', 17), ('kno
```

THANK YOU

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

- Under Prof. Sasadhar Bera, Ph.D. (Indian Institute of Mamagement ,Ranchi)