Opinion Mining + Sentiment Classification :

For the Top 10 Indian Web Series(Comedy 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 Comedy 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 comedy genre.
r1_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r2_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r3_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r4_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r5_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r6_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r7_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r8_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r9_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVIE
r10_df=pd.read_excel(r"C:\Users\Asus\Desktop\Intern Work-IIM Ranchi\ALL REVIEWS\COMEDY REVI
```

In [3]:

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

Out[3]:

	Unnamed: 0
0	Nostalgic
1	TVF has done it again, Brilliant series.
2	Please dont stop. Continue journey with Season...
3	Story which will seriously tickle it's audienc...
4	Must watch web series
...	...
687	This show has everything right except the ster...
688	I watched the series very recently and enjoyed...
689	Typical TVF look a like show, with good hardwo...
690	I would have given this season a rating of 9, ...
691	Centred around the middle class Mishra family ...

692 rows × 1 columns

In [4]:

```
r2_df
```

Out[4]:

	Unnamed: 0
0	I occasionally review any shows or web series ...
1	I never write any review before but this time ...
2	Serials, movies and web series all have a fix ...
3	Showcased the life of a young IT employee and ...
4	The best web series in the world ...covered al...
...	...
633	I was so excited for the Pichers Finale to be ...
634	First of all I would like to congratulate the ...
635	Amazing series for those who dream and work ha...
636	"There's nothing wrong with staying small. You...
637	Cant wait to see what happens next. Keep it up...

638 rows × 1 columns

In [5]:

```
r3_df
```

Out[5]:

	Unnamed: 0
0	I am speechless guys really... Melted by the u...
1	TVF has hit a goal so hard that any 80's or 90...
2	I have never watched anything more real in my ...
3	Yeh Meri Family\n\nNostalgia, nostalgia and mo...
4	This is so refreshing. Being in my teens in th...
...	...
1209	Good script...good direction... awesome perfor...
1210	Love this show so much that I already have wat...
1211	After a long time i watched a show of 90s best...
1212	A very good series!!!!Hell yeah!!!!I ate omele...
1213	This is the most anticipated series as it take...

1214 rows × 1 columns

In [6]:

```
r4_df
```

Out[6]:

	Unnamed: 0
0	A must watch with your Yaar and Pyaar
1	Best webseries
2	Bagga rocks
3	Bang on show
4	Hats off to The Timeliners
...	...
2662	Why is this rated 9?\nNot a single thing in th...
2663	I have watched it 3 times. I want more season ...
2664	Thissss was brilliant frm timeliners. Lob u ba...
2665	You will surely miss your friends and if you w...
2666	College Romance and Flames are the best series...

2667 rows × 1 columns

In [7]:

```
r5_df
```

Out[7]:

	Unnamed: 0
0	TVF is known to produce content which is roote...
1	As a 25 year old engineer, I could absolutely ...
2	Hello everyone, \nPanchayat !!\nA deja vu to S...
3	One of the simplest, finest and a heart warmin...
4	Amid lockdown it's been bliss watching Panchay...
...	...
1663	Worth watch.....
1664	When Panchayat 2 is released????
1665	.....AWESOME.....
1666	super !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!...
1667	This is good but not the great.

1668 rows × 1 columns

In [8]:

```
r6_df
```

Out[8]:

	Unnamed: 0
0	NaN
1	REVIEWS OF HOSTEL DAZE
2	NaN
3	TVF proves it yet again, why their series are ...
4	An amazing series to watch. All episodes are f...
...	...
199	gaand faad
200	This was just plain beautiful to watch!
201	story
202	Good show reminds one of their own hostel stor...
203	Very relatable and well directed

204 rows × 1 columns

In [9]:

```
r7_df
```

Out[9]:

	Unnamed: 0
0	Zakir khan take a bow. A class act from start ...
1	Firstly i thought as Zakir khan being comedian...
2	Well this show is a combo of comedy and emotio...
3	I only like to watch witty sitcoms and this is...
4	Season 1 is full with comedy, emotion , drama....
...	...
269	Enjoyed throughly.
270	Waste of time
271	Watched two episodes so far.Positives : Fresh,...
272	Don't hesitate to see this season, you will de...
273	Waste of time, painful to watch.

274 rows × 1 columns

In [10]:

```
r8_df
```

Out[10]:

	Unnamed: 0
0	Started as a light heart comedy which is slowl...
1	.. and funny.. \nIt has been long since a show...
2	Amazing show. Brilliantly written, conceptuali...
3	Good effort and has us laughing hard at places...
4	Kept me hooked throughout and I can't wait to ...
...	...
234	Loved it!
235	Super dooper mad and naturally funny!!!
236	Amazing..must watch
237	An amazing
238	Enjoyed everything but the storyline

239 rows × 1 columns

In [11]:

```
r9_df
```

Out[11]:

	Unnamed: 0
0	I have been following comic scene in India sin...
1	I think this show has genuinely actually found...
2	I think the negativity for this show makes no ...
3	What I like about this show is it's raw and im...
4	I think people are missing the point of the sh...
...	...
239	its ok
240	=(
241	There is nothing good about this except the mu...
242	NaN
243	Ohh. Pehla review. Umm... cool. Waiting for th...

244 rows × 1 columns

In [12]:

```
r10_df
```

Out[12]:

	Unnamed: 0
0	REFRESHING SHOW .\nStarring cyrus sahukar and ...
1	Oh so going against the general trend, let me ...
2	Malhotras aren't a couple who are that crazy a...
3	I would give the show 1/5. I loved the trailor...
4	This is a mind refreshing comedy show, which c...
...	...
166	2.5/5\n\nWatch for light hearted, rib tickling...
167	time wasting series
168	worst.....waste of time
169	forced to act crazyy
170	watch sumit sambhal lega instead

171 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
0	Nostalgic
1	TVF has done it again, Brilliant series.
2	Please dont stop. Continue journey with Season...
3	Story which will seriously tickle it's audienc...
4	Must watch web series
...	...
8006	2.5/5\n\nWatch for light hearted, rib tickling...
8007	time wasting series
8008	worst.....waste of time
8009	forced to act crazyy
8010	watch sumit sambhal lega instead

8011 rows × 1 columns

In [15]:

```
#naming the column(s)
combined_df.columns=['transcript']
```

In [16]:

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

Out[16]:

	transcript
0	Nostalgic
1	TVF has done it again, Brilliant series.
2	Please dont stop. Continue journey with Season...
3	Story which will seriously tickle it's audienc...
4	Must watch web series
...	...
8006	2.5/5\n\nWatch for light hearted, rib tickling...
8007	time wasting series
8008	worst.....waste of time
8009	forced to act crazyy
8010	watch sumit sambhal lega instead

8011 rows × 1 columns

In [17]:

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

Out[17]:

	transcript
803	Really good I like it. waiting for next episode
3826	Out of all the web series i find this one as o...
2057	A not to be missed gem
431	The best series based on middle class family. ...
1995	Best TVF show ever
3127	Bombasting.....
3731	If u want to really enjoy ur time u should wat...
7934	super funny! really hope theres a season 2
6493	hilarious simplicity and comedy.... reminds m...
3575	A series you must watch

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 [18]:

```
# 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 [19]:

```
# 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[19]:

	transcript
0	nostalgic
1	tvf has done it again brilliant series
2	please dont stop continue journey with season ...
3	story which will seriously tickle its audience...
4	must watch web series
...	...
8006	\n\nwatch for light hearted rib tickling humor
8007	time wasting series
8008	worstwaste of time
8009	forced to act crazyy
8010	watch sumit sambhal lega instead

8011 rows × 1 columns

In [20]:

```
# 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 [21]:

```
# 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[21]:

	transcript
0	nostalgic
1	tvf has done it again brilliant series
2	please dont stop continue journey with season ...
3	story which will seriously tickle its audience...
4	must watch web series
...	...
8006	watch for light hearted rib tickling humor
8007	time wasting series
8008	worstwaste of time
8009	forced to act crazyy
8010	watch sumit sambhal lega instead

8011 rows × 1 columns

In [22]:

```
# 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 [23]:

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

In [24]:

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

Out[24]:

	transcript
0	nostalgic
1	tvf has done it again brilliant series
2	please dont stop continue journey with season ...
3	story which will seriously tickle its audience...
4	must watch web series
...	...
8006	watch for light hearted rib tickling humor
8007	time wasting series
8008	worstwaste of time
8009	forced to act crazyy
8010	watch sumit sambhal lega instead

8011 rows × 1 columns

In [25]:

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

Out[25]:

	transcript
5191	if you want to know how to ruin sequel of a go...
7950	not goodnot relatableas indian we are more rel...
7242	refreshingly accurate depiction of a typical s...
3992	an awesome series with cool cast specially i e...
7040	nice good
1880	so glad this was made

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 [26]:

```
# 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 [27]:

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

```
{"we'll", "we'd", 'http', "it's", 'about', 'out', 'yours', 'shall', 'under',
"they'll", 'it', 'this', "won't", 'own', "they've", 'do', 'during', 'had',
'could', 'if', 'should', 'theirs', 'yourself', 'has', 'they', "didn't", 'hen
ce', 'by', "mustn't", 'you', 'nor', 'its', 'with', "you'd", "why's", "he'l
l", "wasn't", 'up', "we're", 'again', "she'll", 'am', 'from', 'not', 'thes
e', 'all', 'when', 'your', 'was', 'of', 'same', 'themselves', 'because', 'ge
t', 'than', 'each', "shan't", "you've", 'as', "i'd", 'i', 'are', 'their', 't
o', 'www', 'no', 'so', "how's", 'on', 'been', 'over', 'is', 'above', 'hersel
f', 'else', 'does', 'she', "let's", 'into', 'some', 'have', 'where', 'himsel
f', 'other', "hadn't", 'an', 'both', "hasn't", 'only', 'few', 'those', 'bu
t', 'in', 'myself', 'too', 'he', "can't", 'who', 'the', 'through', "would
n't", 'and', 'him', "haven't", "we've", "don't", "doesn't", 'which', 'woul
d', "they're", 'ought', 'here', 'like', 'doing', "i'll", "isn't", 'at', 'hav
ing', 'itself', 'them', 'any', 'also', 'since', 'therefore', "she's", 'or',
"he's", 'r', 'ever', 'that', 'while', 'for', 'very', "he'd", 'were', 'we',
"who's", 'her', 'before', 'below', 'my', "when's", 'between', 'whom', "sh
e'd", "i'm", "couldn't", "aren't", 'did', "where's", 'hers', 'yourselves',
'com', 'off', "that's", 'can', 'his', 'why', 'further', 'our', 'otherwise',
"they'd", 'ourselves', 'me', 'most', 'just', 'such', 'being', 'k', "should
n't", 'how', "what's", 'ours', 'cannot', "there's", "i've", 'down', 'more',
"you'll", 'there', 'a', 'be', 'after', 'once', 'until', "you're", "here's",
'however', "weren't", 'then', 'what', 'against'}
```

In [28]:

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

Out[28]:

```
'nostalgic tvf has done it again brilliant series please dont stop continu
e journey with season and so on story which will seriously tickle its audi
ence for sure must watch web series need season tvf impress everytime best
web series of the year takes you back in time very well directed best tv s
eries one of the best from tvf loved it one the best web series must watch
for every household amazing show close to my heart its beautiful superb ca
st and beautiful episodes tvf has done again a fantastic job none less tha
n season simplicity at its finest i cried watching this ye zindagi yaado k
i gullak si tvf means happiness bestest series for this year tvf rules aga
in one for ages out of this world i wish i could gove more than simplicity
at its best must watch mind blowing truely awesome reminds our childhood d
ays simply hilarious such a well made show fabulous show nostalgic just be
autiful real excellent seriesis there a second season coming tvf nailed it
laugh riot that is so so pure and comes from heart cute oh my hod what a s
eries close to reality full of emotions wow just wow excellent storyline o
f a middleclass challenges related connection to the viewers at its best l
oved it based on true situations of a middle class family best of the best
what a chemistry between family members typical indian family best series
```

In [29]:

```
# 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', 8412), ('and', 5733), ('of', 4669), ('a', 3974), ('series', 380
7), ('to', 3806), ('is', 3693), ('i', 3215), ('it', 3117), ('this', 2879),
('in', 2435), ('for', 2134), ('best', 1908), ('you', 1785), ('watch', 168
0), ('with', 1534), ('all', 1478), ('show', 1464), ('web', 1259), ('seaso
n', 1242), ('was', 1210), ('are', 1179), ('that', 1143), ('one', 1089),
('have', 1081), ('so', 1043), ('its', 1030), ('very', 1012), ('love', 95
1), ('but', 918), ('my', 895), ('just', 890), ('good', 889), ('story', 83
1), ('on', 818), ('must', 793), ('like', 780), ('will', 754), ('amazing',
734), ('not', 722), ('family', 720), ('as', 719), ('awesome', 714), ('ever
y', 710), ('tvf', 698), ('acting', 693), ('by', 679), ('life', 667), ('gre
at', 658), ('be', 644), ('ever', 626), ('loved', 619), ('more', 617), ('co
llege', 582), ('really', 580), ('can', 559), ('from', 558), ('characters',
543), ('waiting', 528), ('has', 523), ('your', 514), ('me', 508), ('whic
h', 495), ('they', 493), ('time', 487), ('watching', 487), ('an', 486),
('episode', 484), ('bagga', 481), ('their', 475), ('watched', 470), ('char
acter', 461), ('indian', 440), ('episodes', 436), ('well', 431), ('up', 43
1), ('comedy', 406), ('we', 400), ('such', 388), ('cast', 383), ('make', 3
83), ('no', 375), ('much', 372), ('at', 367), ('if', 357), ('real', 354),
('or', 353), ('who', 350), ('what', 349), ('about', 346), ('actors', 344),
```

In [30]:

```
# 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 > 2135]
add_stop_words
```

Out[30]:

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

In [31]:

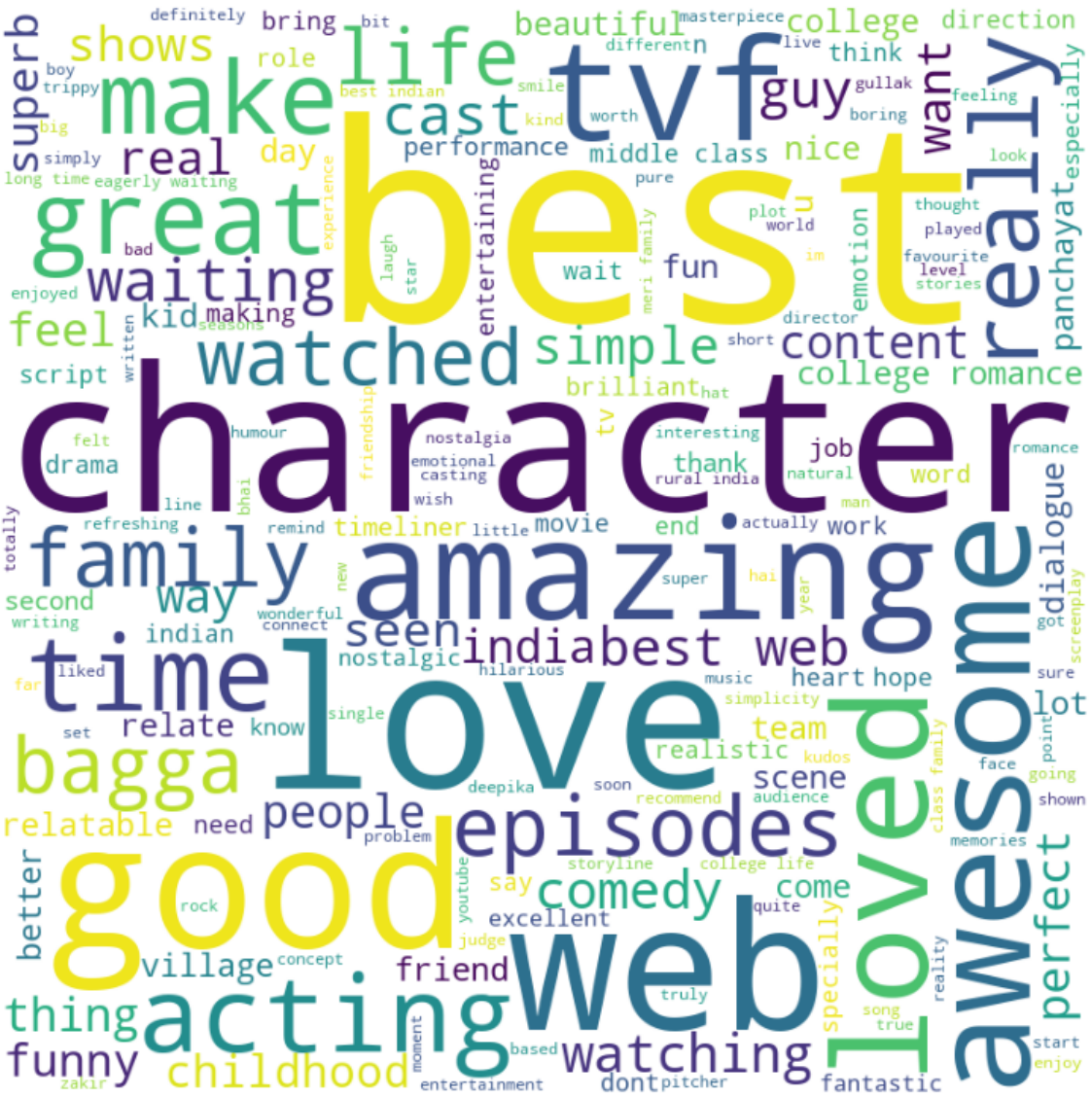
```
#adding more stopwords for better analysis
from sklearn.feature_extraction import text
additional_stop_words = text.ENGLISH_STOP_WORDS
print (additional_stop_words)
```

```
frozenset({'three', 'about', 'out', 'around', 'six', 'found', 'along', 'thi
s', 'rather', 'alone', 'without', 'others', 'could', 'has', 'hence', 'someti
mes', 'up', 'thereby', 'whose', 'fill', 'your', 'same', 'than', 'bottom', 'e
nough', 'sincere', 'amount', 'third', 'together', 'to', 'no', 'something',
'therein', 'been', 'over', 'else', 'whereupon', 'afterwards', 'she', 'into',
'other', 'show', 'too', 'few', 'he', 'through', 'anyway', 'him', 'us', 'ye
t', 'everyone', 'would', 'here', 'now', 'nowhere', 'either', 'beforehand',
'beside', 'very', 'next', 'someone', 'even', 'hereafter', 'before', 'below',
'fifty', 'whom', 'whence', 'forty', 'within', 'always', 'our', 'more', 'te
n', 'then', 'eleven', 'thereupon', 'thereafter', 'becomes', 'on', 'herein',
'it', 'wherein', 'mill', 'never', 'do', 'had', 'well', 'top', 'anyhow', 'sys
tem', 'with', 'again', 'twenty', 'seemed', 'not', 'whither', 'when', 'empt
y', 'whether', 'because', 'everywhere', 'formerly', 'get', 'each', 'eg', 'pa
rt', 'thin', 'are', 'their', 'whole', 'due', 'per', 'above', 'herself', 'las
t', 'move', 'fire', 'back', 'have', 'where', 'both', 'only', 'but', 'cant',
'who', 'couldnt', 'became', 'nobody', 'detail', 'them', 'two', 'mostly', 'si
xty', 'seems', 'therefore', 'made', 'hereupon', 'or', 'sometime', 'thus', 'w
hile', 'whereby', 'were', 'her', 'hasnt', 'former', 'hers', 'off', 'furthe
r', 'otherwise', 'toward', 'most', 'five', 're', 'ours', 'every', 'call', 't
here', 'be', 'after', 'once', 'whereafter', 'become', 'thru', 'first', 'ofte
n', 'yours', 'under', 'upon', 'neither', 'yourself', 'during', 'should', 'i
f', 'elsewhere', 'they', 'may', 'its', 'one', 'except', 'am', 'from', 'fou
r', 'behind', 'nine', 'these', 'all', 'side', 'of', 'themselves', 'across',
'nothing', 'must', 'as', 'eight', 'anything', 'among', 'serious', 'is', 'alm
ost', 'some', 'besides', 'himself', 'seeming', 'meanwhile', 'already', 'an
d', 'everything', 'although', 'mine', 'un', 'ie', 'inc', 'any', 'since', 'an
ywhere', 'onto', 'ever', 'that', 'go', 'least', 'we', 'between', 'take', 'tw
elve', 'towards', 'might', 'yourselves', 'can', 'done', 'con', 'ourselves',
'please', 'interest', 'will', 'latter', 'latterly', 'still', 'amoungst', 'ca
nnot', 'down', 'however', 'becoming', 'see', 'whatever', 'against', 'full',
'throughout', 'find', 'own', 'front', 'by', 'you', 'nor', 'etc', 'wherever',
'perhaps', 'hundred', 'was', 'many', 'indeed', 'nevertheless', 'ltd', 'i',
'name', 'describe', 'give', 'so', 'none', 'whoever', 'via', 'another', 'muc
h', 'somehow', 'an', 'bill', 'those', 'in', 'myself', 'the', 'fifteen', 'noo
ne', 'which', 'de', 'hereby', 'thence', 'namely', 'at', 'itself', 'also', 's
omewhere', 'keep', 'for', 'cry', 'less', 'moreover', 'though', 'anyone', 'wh
enever', 'my', 'whereas', 'several', 'put', 'thick', 'amongst', 'his', 'wh
y', 'co', 'seem', 'me', 'such', 'being', 'how', 'beyond', 'a', 'until', 'wha
t'})
```

In [32]:

```
# 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 [33]:

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

```
['three', 'about', 'out', 'around', 'six', 'found', 'along', 'this', 'rathe
r', 'alone', 'without', 'others', 'could', 'has', 'hence', 'sometimes', 'u
p', 'thereby', 'whose', 'fill', 'your', 'same', 'than', 'bottom', 'enough',
'sincere', 'amount', 'third', 'together', 'to', 'no', 'something', 'therei
n', 'been', 'over', 'else', 'whereupon', 'afterwards', 'she', 'into', 'othe
r', 'show', 'too', 'few', 'he', 'through', 'anyway', 'him', 'us', 'yet', 'ev
eryone', 'would', 'here', 'now', 'nowhere', 'either', 'beforehand', 'besid
e', 'very', 'next', 'someone', 'even', 'hereafter', 'before', 'below', 'fift
y', 'whom', 'whence', 'forty', 'within', 'always', 'our', 'more', 'ten', 'th
en', 'eleven', 'thereupon', 'thereafter', 'becomes', 'on', 'herein', 'it',
'wherein', 'mill', 'never', 'do', 'had', 'well', 'top', 'anyhow', 'system',
'with', 'again', 'twenty', 'seemed', 'not', 'whither', 'when', 'empty', 'whe
ther', 'because', 'everywhere', 'formerly', 'get', 'each', 'eg', 'part', 'th
in', 'are', 'their', 'whole', 'due', 'per', 'above', 'herself', 'last', 'mov
e', 'fire', 'back', 'have', 'where', 'both', 'only', 'but', 'cant', 'who',
'couldnt', 'became', 'nobody', 'detail', 'them', 'two', 'mostly', 'sixty',
'seems', 'therefore', 'made', 'hereupon', 'or', 'sometime', 'thus', 'while',
'whereby', 'were', 'her', 'hasnt', 'former', 'hers', 'off', 'further', 'othe
rwise', 'toward', 'most', 'five', 're', 'ours', 'every', 'call', 'there', 'b
e', 'after', 'once', 'whereafter', 'become', 'thru', 'first', 'often', 'your
s', 'under', 'upon', 'neither', 'yourself', 'during', 'should', 'if', 'elsew
here', 'they', 'may', 'its', 'one', 'except', 'am', 'from', 'four', 'behin
d', 'nine', 'these', 'all', 'side', 'of', 'themselves', 'across', 'nothing',
'must', 'as', 'eight', 'anything', 'among', 'serious', 'is', 'almost', 'som
e', 'besides', 'himself', 'seeming', 'meanwhile', 'already', 'and', 'everyth
ing', 'although', 'mine', 'un', 'ie', 'inc', 'any', 'since', 'anywhere', 'on
to', 'ever', 'that', 'go', 'least', 'we', 'between', 'take', 'twelve', 'towa
rds', 'might', 'yourselves', 'can', 'done', 'con', 'ourselves', 'please', 'i
nterest', 'will', 'latter', 'latterly', 'still', 'amoungst', 'cannot', 'dow
n', 'however', 'becoming', 'see', 'whatever', 'against', 'full', 'throughou
t', 'find', 'own', 'front', 'by', 'you', 'nor', 'etc', 'wherever', 'perhap
s', 'hundred', 'was', 'many', 'indeed', 'nevertheless', 'ltd', 'i', 'name',
'describe', 'give', 'so', 'none', 'whoever', 'via', 'another', 'much', 'some
how', 'an', 'bill', 'those', 'in', 'myself', 'the', 'fifteen', 'noone', 'whi
ch', 'de', 'hereby', 'thence', 'namely', 'at', 'itself', 'also', 'somewher
e', 'keep', 'for', 'cry', 'less', 'moreover', 'though', 'anyone', 'wheneve
r', 'my', 'whereas', 'several', 'put', 'thick', 'amongst', 'his', 'why', 'c
o', 'seem', 'me', 'such', 'being', 'how', 'beyond', 'a', 'until', 'what', 's
how', 'season', 'one', 'season', 'watch', 'story', 'bajpayee', 'webserie',
'webseries', 'bajpai', 'manoj', 'episode', 'review', 'actor', 'actors', 'th
e', 'and', 'of', 'a', 'series', 'to', 'is', 'i', 'it', 'this', 'in', "we'l
l", "we'd", 'http', "it's", 'about', 'out', 'yours', 'shall', 'under', "the
y'll", 'it', 'this', "won't", 'own', "they've", 'do', 'during', 'had', 'coul
d', 'if', 'should', 'theirs', 'yourself', 'has', 'they', "didn't", 'hence',
'by', "mustn't", 'you', 'nor', 'its', 'with', "you'd", "why's", "he'll", "wa
sn't", 'up', "we're", 'again', "she'll", 'am', 'from', 'not', 'these', 'al
l', 'when', 'your', 'was', 'of', 'same', 'themselves', 'because', 'get', 'th
an', 'each', "shan't", "you've", 'as', "i'd", 'i', 'are', 'their', 'to', 'ww
w', 'no', 'so', "how's", 'on', 'been', 'over', 'is', 'above', 'herself', 'el
se', 'does', 'she', "let's", 'into', 'some', 'have', 'where', 'himself', 'ot
her', "hadn't", 'an', 'both', "hasn't", 'only', 'few', 'those', 'but', 'in',
'myself', 'too', 'he', "can't", 'who', 'the', 'through', "wouldn't", 'and',
'him', "haven't", "we've", "don't", "doesn't", 'which', 'would', "they're",
'ought', 'here', 'like', 'doing', "i'll", "isn't", 'at', 'having', 'itself',
'them', 'any', 'also', 'since', 'therefore', "she's", 'or', "he's", 'r', 'ev
er', 'that', 'while', 'for', 'very', "he'd", 'were', 'we', "who's", 'her',
```

```
'before', 'below', 'my', "when's", 'between', 'whom', "she'd", "i'm", "could
n't", "aren't", 'did', "where's", 'hers', 'yourselves', 'com', 'off', "tha
t's", 'can', 'his', 'why', 'further', 'our', 'otherwise', "they'd", 'ourselv
es', 'me', 'most', 'just', 'such', 'being', 'k', "shouldn't", 'how', "wha
t's", 'ours', 'cannot', "there's", "i've", 'down', 'more', "you'll", 'ther
e', 'a', 'be', 'after', 'once', 'until', "you're", "here's", 'however', "wer
en't", 'then', 'what', 'against']
```

Findings

We can clearly see that the word cloud has major chunk of positve reviews(roughly 75%) , some negative reviews (roughly 5%), with some neutral reviews(20%).

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 [34]:

```
# 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[34]:

	transcript	polarity	subjectivity
0	nostalgic	-0.500000	1.000000
1	tvf has done it again brilliant series	0.900000	1.000000
2	please dont stop continue journey with season ...	0.000000	0.000000
3	story which will seriously tickle its audience...	0.083333	0.777778
4	must watch web series	0.000000	0.000000
...	...	...	...
8006	watch for light hearted rib tickling humor	0.400000	0.700000
8007	time wasting series	0.000000	0.000000
8008	worstwaste of time	0.000000	0.000000
8009	forced to act crazyy	-0.300000	0.200000
8010	watch sumit sambhal lega instead	0.000000	0.000000

8011 rows × 3 columns

In [35]:

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

Out[35]:

	transcript	polarity	subjectivity
196	benchmark is set for ages	0.000000	0.000000
7941	it was a joy ridemini cyrus are superb	0.900000	0.600000
6046	i saw panchayat web series really i enjoyed to...	0.300000	0.400000
4962	baggaaaaaaaa i loved the series please make so...	0.566667	0.633333
1039	amazing storyline amazing acting overall super...	0.440000	0.560000

In [36]:

```
#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[36]:

	transcript	polarity	subjectivity	Analysis
0	nostalgic	-0.500000	1.000000	negative
1	tvf has done it again brilliant series	0.900000	1.000000	positive
2	please dont stop continue journey with season ...	0.000000	0.000000	neutral
3	story which will seriously tickle its audience...	0.083333	0.777778	positive
4	must watch web series	0.000000	0.000000	neutral
...	...	...	...	...
8006	watch for light hearted rib tickling humor	0.400000	0.700000	positive
8007	time wasting series	0.000000	0.000000	neutral
8008	worstwaste of time	0.000000	0.000000	neutral
8009	forced to act crazyy	-0.300000	0.200000	negative
8010	watch sumit sambhal lega instead	0.000000	0.000000	neutral

8011 rows × 4 columns

In [37]:

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

Out[37]:

	transcript	polarity	subjectivity	Analysis
5699	congratulations tvf for making such an awesome...	0.500000	0.750000	positive
5136	best series for youngstera and college student...	0.833333	0.366667	positive
3461	siraa	0.000000	0.000000	neutral
2117	brings on waves of nostalgia	0.000000	0.000000	neutral
5180	have a great time watching this and had a fun ...	0.650000	0.437500	positive
5641	its simplicity is so alluring you will connect...	0.250000	0.200000	positive
7763	a very good entertaining smart and funny india...	0.468571	0.780714	positive
5397	nice light comedy and awesome surroundings of ...	0.518750	0.700000	positive
418	beautiful creation by tvf and the direction te...	0.625000	1.000000	positive
7766	very funny look forward to it each week indian...	0.275000	0.866667	positive

In [38]:

```
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(Comed
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(Comedy Genre) :

No of Negative Reviews from our Total DataSet(around 10k) -> 419
No of Positive Reviews from our Total DataSet(around 10k) -> 6364
No of  Neutral Reviews from our Total DataSet(around 10k) -> 1228

Percentage of Negative Reviews -> 5.230308326051679 %
Percentage of Positive Reviews -> 79.44076894270378 %
Percentage of Neutral  Reviews -> 15.328922731244537  %
```

Sentiment Findings:

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

```
No of Negative Reviews from our Total DataSet(around 10k) -> 419
No of Positive Reviews from our Total DataSet(around 10k) -> 6364
No of  Neutral Reviews from our Total DataSet(around 10k) -> 1228

Percentage of Negative Reviews -> 5.230308326051679 %
Percentage of Positive Reviews -> 79.44076894270378 %
Percentage of Neutral  Reviews -> 15.328922731244537 %
```

```
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

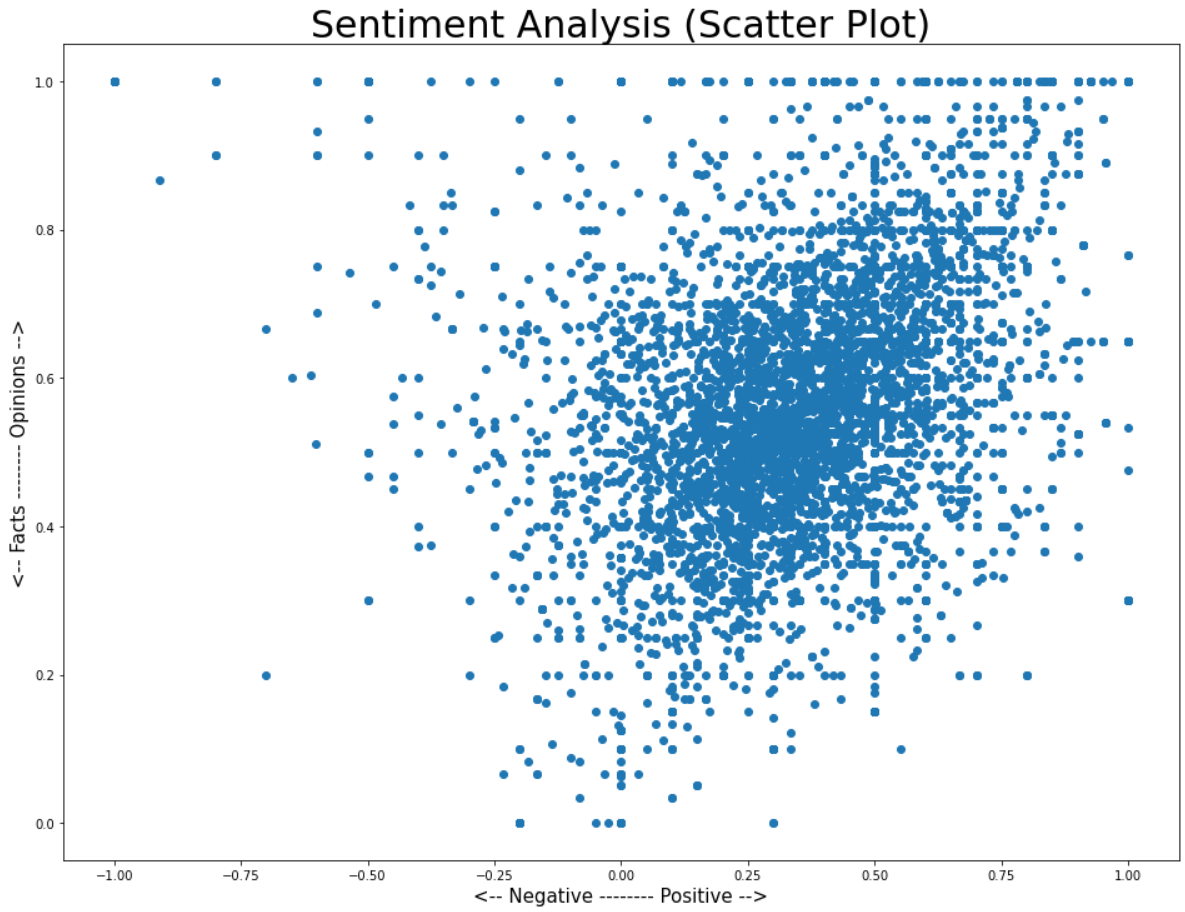
Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When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

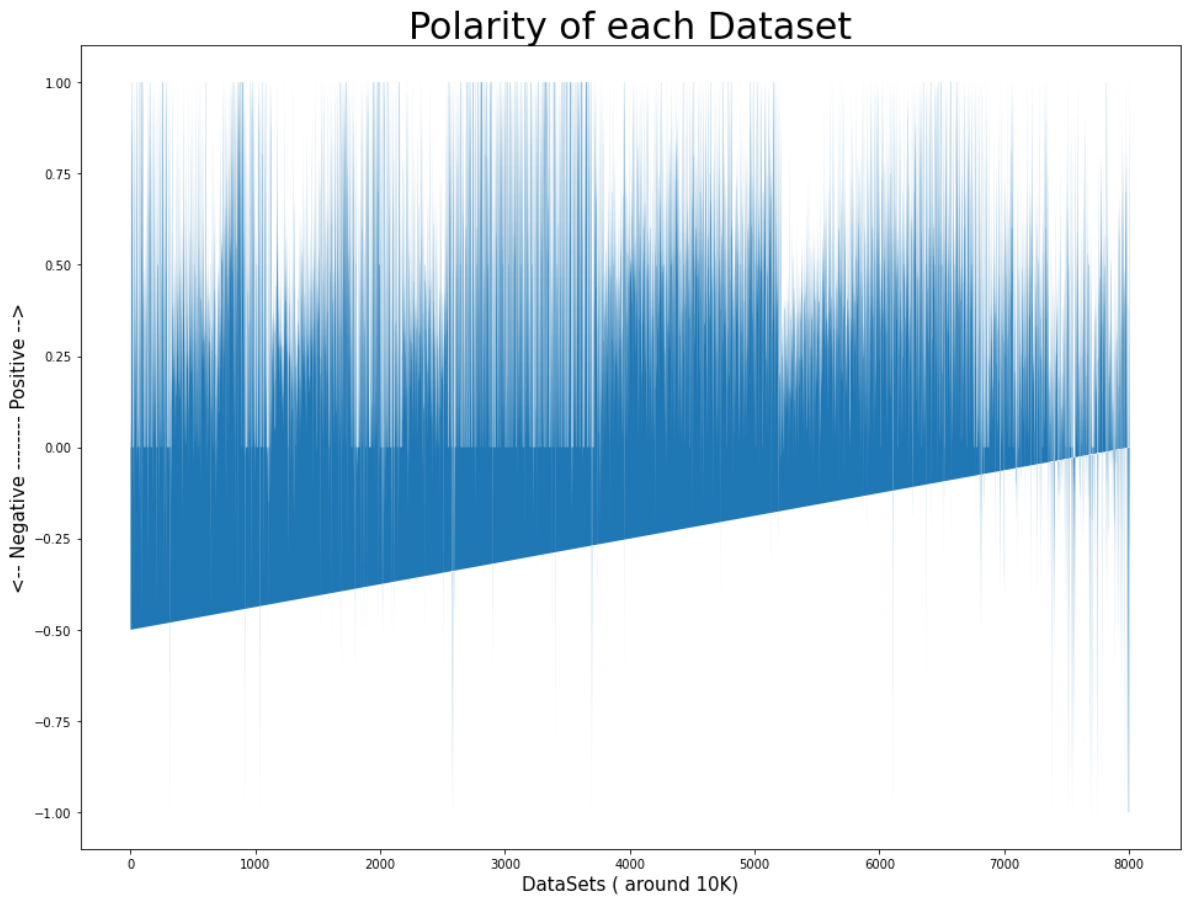
In [39]:

```
# 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()
```



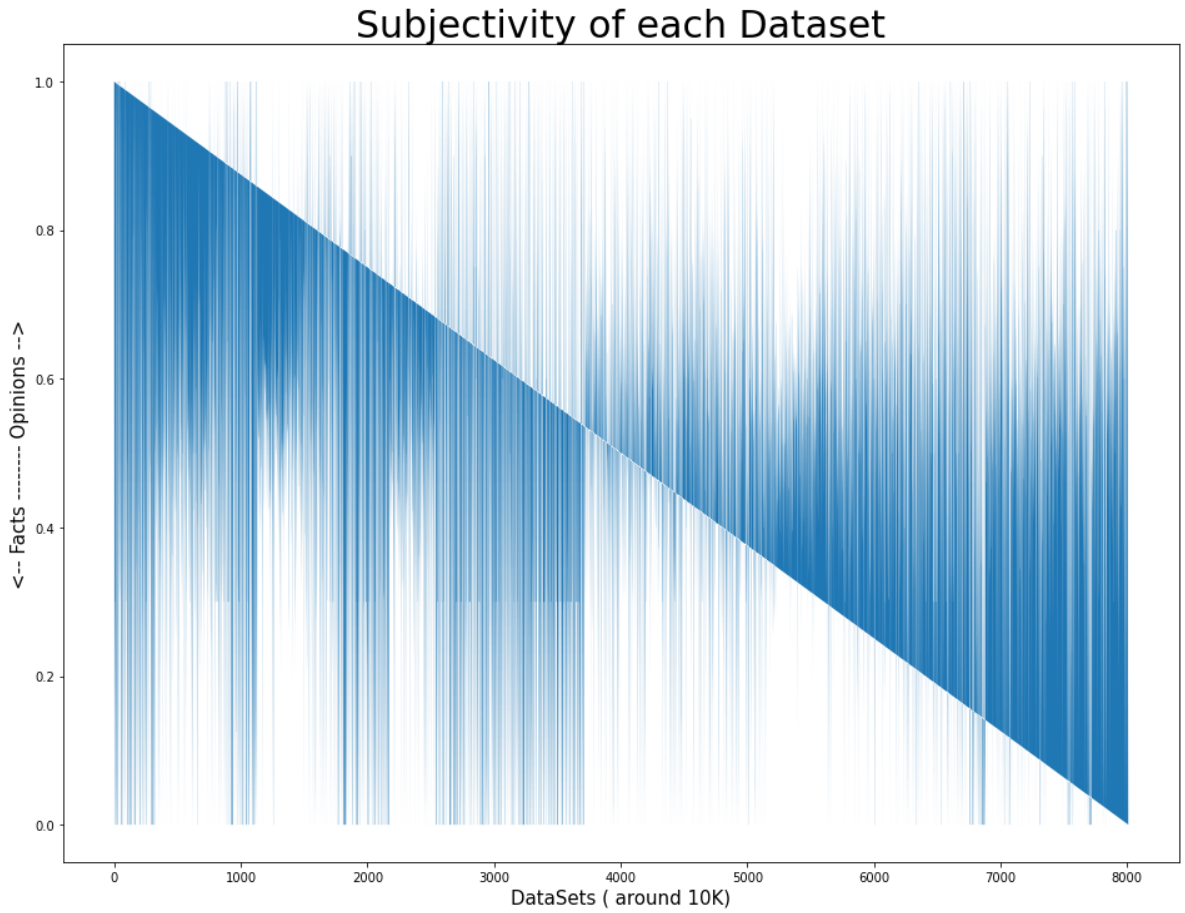
In [40]:

```
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()
```



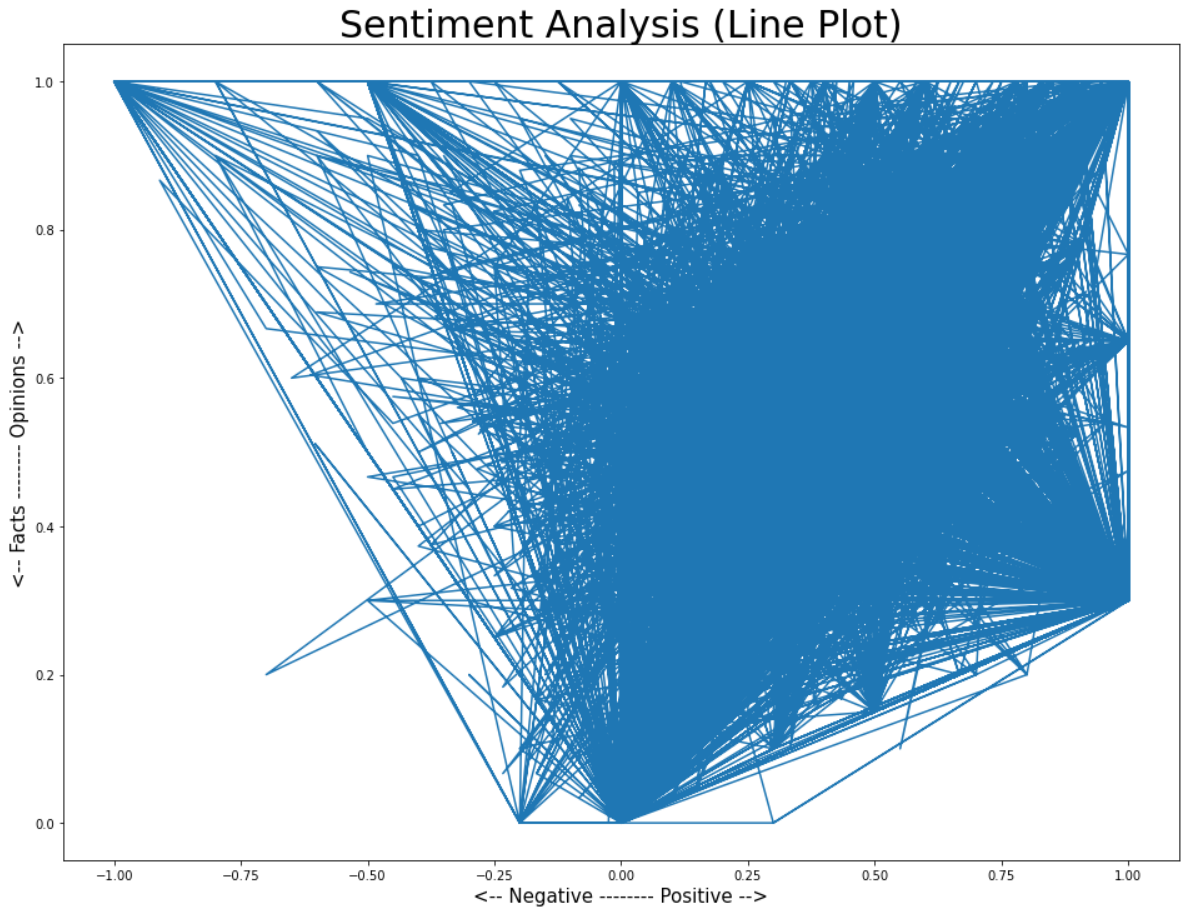
In [41]:

```
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()
```



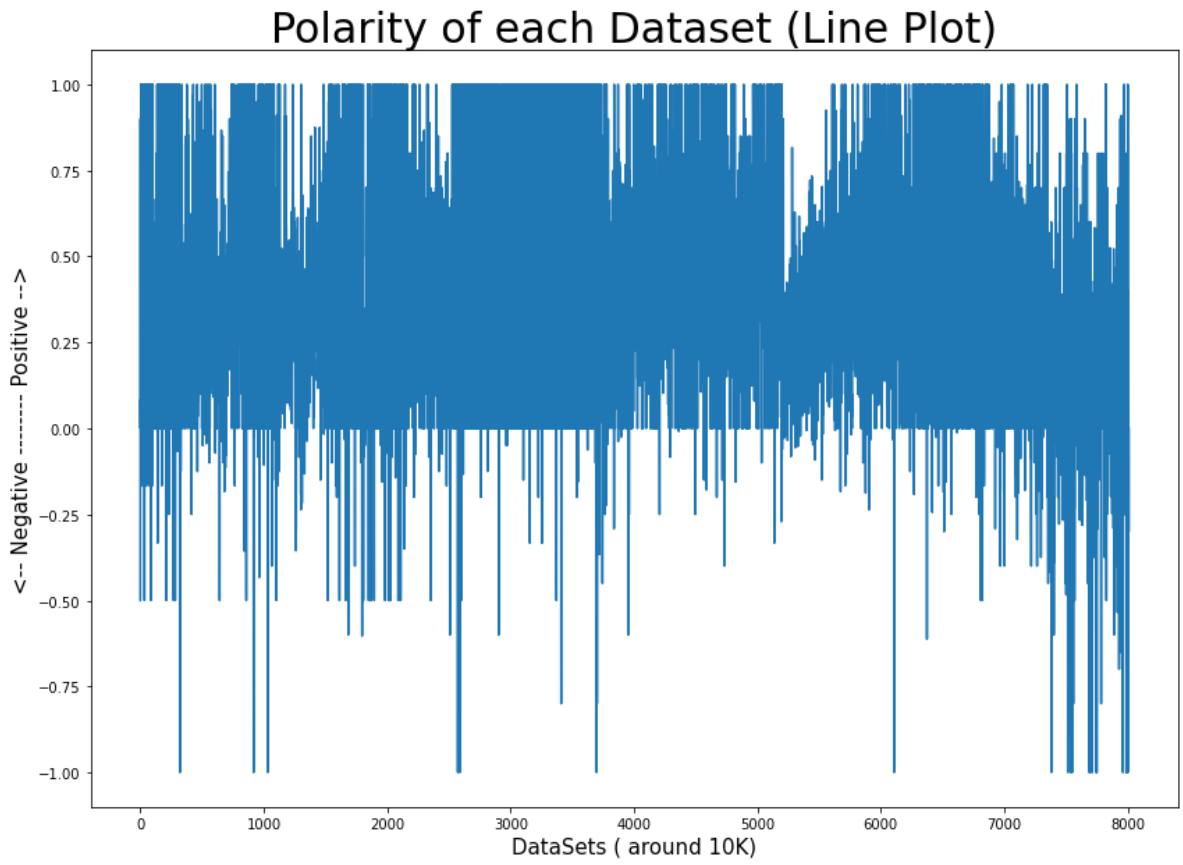
In [42]:

```
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()
```



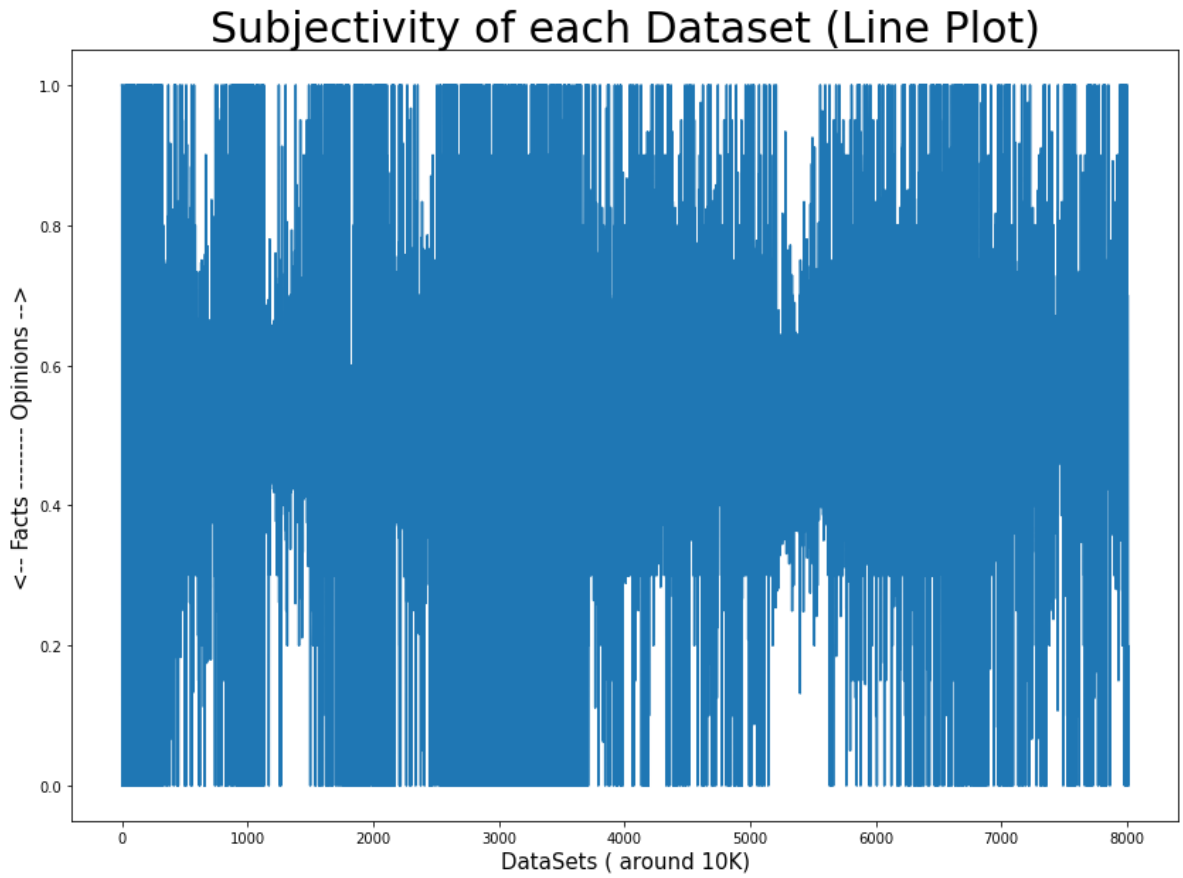
In [43]:

```
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()
```



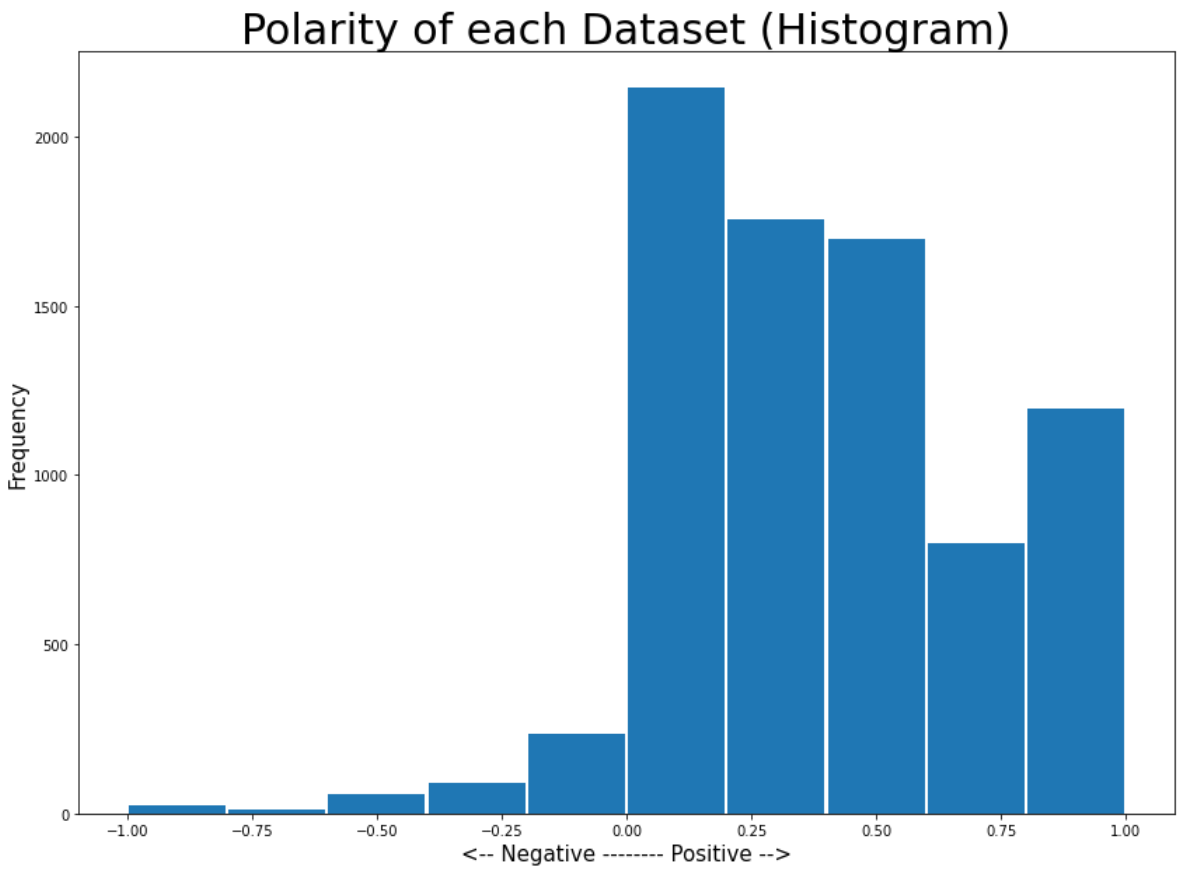
In [44]:

```
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()
```



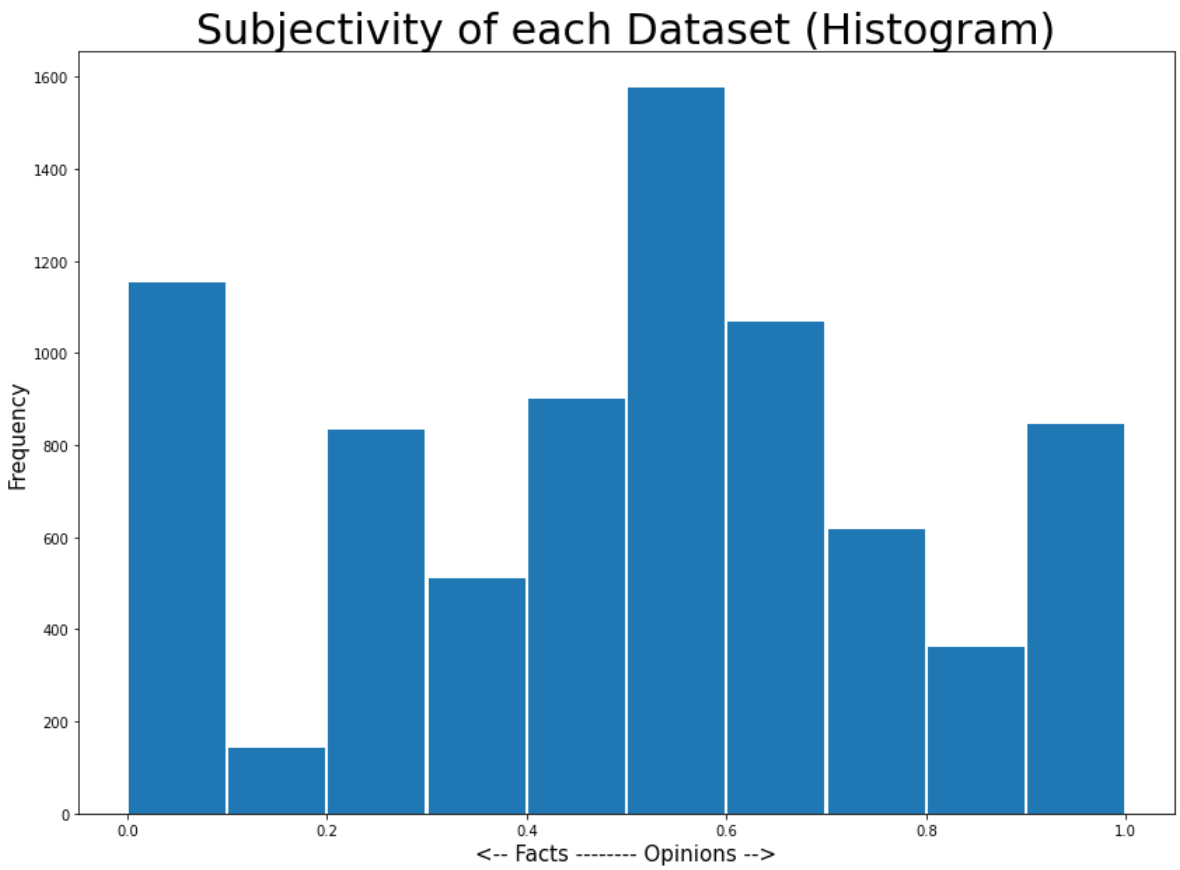
In [45]:

```
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 [46]:

```
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 [47]:

```
data_clean_df
```

Out[47]:

	transcript	polarity	subjectivity	Analysis
0	nostalgic	-0.500000	1.000000	negative
1	tvf has done it again brilliant series	0.900000	1.000000	positive
2	please dont stop continue journey with season ...	0.000000	0.000000	neutral
3	story which will seriously tickle its audience...	0.083333	0.777778	positive
4	must watch web series	0.000000	0.000000	neutral
...	...	...	...	...
8006	watch for light hearted rib tickling humor	0.400000	0.700000	positive
8007	time wasting series	0.000000	0.000000	neutral
8008	worstwaste of time	0.000000	0.000000	neutral
8009	forced to act crazyy	-0.300000	0.200000	negative
8010	watch sumit sambhal lega instead	0.000000	0.000000	neutral

8011 rows × 4 columns

In [48]:

```
#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 [49]:

```
positive_df
```

Out[49]:

	transcript	polarity	subjectivity	Analysis
1	tvf has done it again brilliant series	0.900000	1.000000	positive
3	story which will seriously tickle its audience...	0.083333	0.777778	positive
7	best web series of the year	1.000000	0.300000	positive
8	takes you back in time very well directed	0.100000	0.150000	positive
9	best tv series	1.000000	0.300000	positive
...	...	...	...	...
7995	light hearted show	0.400000	0.700000	positive
7996	its okay show but funny in bits and pieces	0.375000	0.750000	positive
8000	i liked the show a lot	0.600000	0.800000	positive
8001	superb series	1.000000	1.000000	positive
8006	watch for light hearted rib tickling humor	0.400000	0.700000	positive

6364 rows × 4 columns

In [50]:

```
# 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','character','epis
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()
```

WordCloud for POSITVE SENTIMENTS

In [51]:

```
#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 [52]:

```
negative_df
```

Out[52]:

	transcript	polarity	subjectivity	Analysis
0	nostalgic	-0.500000	1.000000	negative
18	none less than season	-0.166667	0.066667	negative
34	nostalgic	-0.500000	1.000000	negative
50	typical indian family	-0.166667	0.500000	negative
53	show to remember for long	-0.050000	0.400000	negative
...	...	...	...	...
7999	terribly bad no fun lame jokes	-0.450000	0.538889	negative
8002	boring irritated web series	-1.000000	1.000000	negative
8003	worst show	-1.000000	1.000000	negative
8004	its super boring nahh	-0.333333	0.833333	negative
8009	forced to act crazyy	-0.300000	0.200000	negative

419 rows × 4 columns

In [63]:

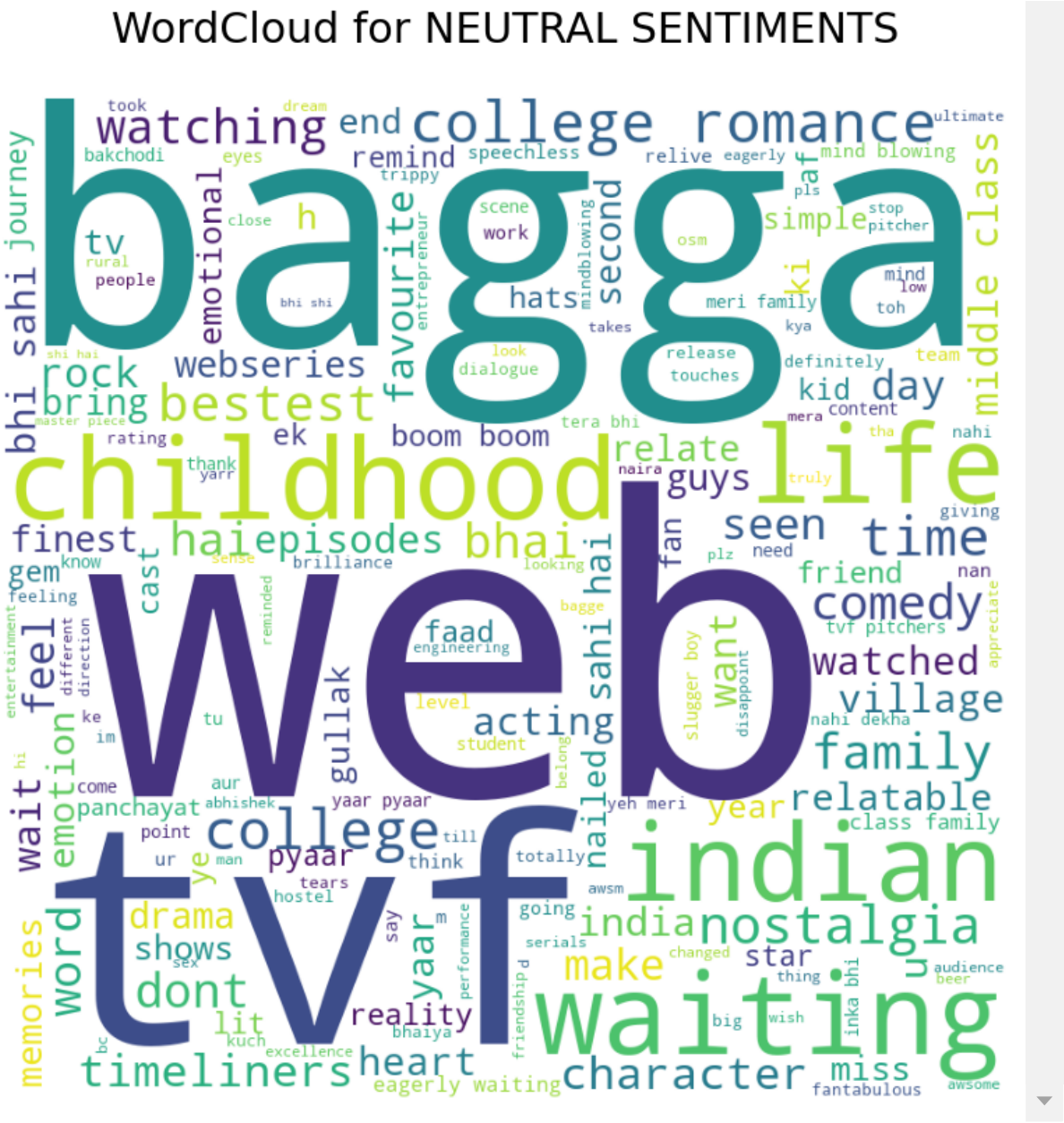
```
# Python program to generate WordCloud for NEGATVE SENTIMENTS

# 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','funny','season','watch','character','cha
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()
```

In [54]:

```
#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 [55]:

```
# 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()
```



Additonal Information

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

In [56]:

```
# 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', 7771), ('and', 5349), ('of', 4225), ('a', 3552), ('series', 351
7), ('to', 3444), ('is', 3354), ('i', 2927), ('it', 2829), ('this', 2586),
('in', 2226), ('for', 1905), ('best', 1904), ('you', 1623), ('watch', 141
5), ('with', 1402), ('all', 1376), ('show', 1294), ('web', 1171), ('was',
1100), ('season', 1075), ('are', 1063), ('that', 1036), ('one', 1018), ('h
ave', 981), ('very', 951), ('so', 945), ('love', 943), ('its', 941), ('goo
d', 858), ('but', 793), ('just', 784), ('my', 770), ('story', 756), ('amaz
ing', 730), ('on', 730), ('awesome', 714), ('like', 703), ('will', 695),
('family', 658), ('great', 658), ('as', 658), ('acting', 648), ('must', 64
6), ('by', 643), ('every', 641), ('tvf', 625), ('loved', 615), ('life', 61
0), ('not', 597), ('more', 596), ('ever', 572), ('be', 572), ('really', 55
6), ('characters', 511), ('college', 510), ('from', 507), ('can', 500),
('has', 483), ('waiting', 480), ('which', 461), ('me', 460), ('an', 455),
('their', 453), ('your', 451), ('they', 442), ('episode', 438), ('watche
d', 435), ('character', 430), ('watching', 427), ('bagga', 423), ('time',
404), ('well', 402), ('episodes', 401), ('indian', 397), ('up', 389), ('w
e', 377), ('cast', 363), ('such', 358), ('comedy', 351), ('much', 349),
('real', 345), ('make', 345), ('at', 332), ('actors', 323), ('also', 322),
('guys', 320), ('about', 317), ('his', 314), ('if', 310), ('what', 309),
```

In [57]:

```
# 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', 442), ('and', 285), ('of', 275), ('a', 254), ('is', 235), ('to',
229), ('i', 191), ('it', 161), ('this', 157), ('in', 140), ('show', 116),
('but', 116), ('series', 111), ('not', 101), ('for', 97), ('are', 92), ('w
as', 85), ('with', 84), ('that', 81), ('watch', 79), ('so', 71), ('you', 7
0), ('have', 66), ('all', 63), ('time', 60), ('very', 59), ('on', 54), ('s
eason', 53), ('boring', 52), ('its', 50), ('just', 50), ('no', 49), ('as',
48), ('story', 47), ('like', 46), ('my', 43), ('they', 43), ('be', 43),
('dont', 42), ('watching', 39), ('nostalgic', 38), ('waste', 37), ('who',
35), ('he', 34), ('comedy', 33), ('or', 33), ('bad', 32), ('good', 31),
('acting', 31), ('your', 31), ('episode', 30), ('has', 30), ('worst', 30),
('up', 29), ('one', 29), ('too', 29), ('can', 28), ('at', 28), ('only', 2
8), ('will', 27), ('from', 27), ('which', 27), ('family', 26), ('every', 2
6), ('web', 26), ('how', 26), ('if', 26), ('funny', 26), ('life', 25), ('a
n', 24), ('really', 24), ('even', 24), ('other', 23), ('such', 23), ('char
acters', 23), ('make', 23), ('by', 22), ('much', 22), ('than', 21), ('watc
hed', 21), ('character', 21), ('episodes', 21), ('what', 21), ('me', 20),
('could', 20), ('after', 20), ('about', 20), ('some', 20), ('zakir', 20),
('any', 19), ('panchayat', 19), ('would', 19), ('more', 19), ('do', 19),
('there', 19), ('his', 19), ('must', 18), ('out', 18), ('think', 18), ('av
```

In [58]:

```
# 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', 199), ('watch', 186), ('series', 179), ('of', 169), ('a', 168),
('this', 136), ('to', 133), ('for', 132), ('must', 129), ('it', 127), ('se
ason', 114), ('is', 104), ('and', 99), ('i', 97), ('you', 92), ('my', 82),
('in', 69), ('web', 62), ('college', 59), ('tvf', 58), ('just', 56), ('bag
ga', 56), ('show', 54), ('with', 48), ('every', 43), ('waiting', 43), ('on
e', 42), ('its', 39), ('all', 39), ('ever', 37), ('family', 36), ('childho
od', 36), ('hai', 36), ('on', 34), ('have', 34), ('romance', 33), ('life',
32), ('will', 32), ('your', 32), ('can', 31), ('like', 31), ('indian', 3
0), ('be', 29), ('story', 28), ('me', 28), ('so', 27), ('that', 26), ('ple
ase', 25), ('nostalgia', 25), ('was', 25), ('bestest', 24), ('back', 24),
('masterpiece', 24), ('not', 24), ('are', 24), ('from', 24), ('should', 2
4), ('time', 23), ('next', 22), ('no', 22), ('comedy', 22), ('dont', 21),
('watching', 21), ('if', 21), ('bhai', 21), ('sahi', 21), ('after', 20),
('boom', 20), ('what', 19), ('cant', 19), ('', 19), ('timeliners', 19),
('bhi', 19), ('seen', 18), ('yaar', 18), ('class', 16), ('episode', 16),
('again', 15), ('middle', 15), ('favourite', 15), ('make', 15), ('india',
15), ('wait', 15), ('made', 14), ('heart', 14), ('never', 14), ('feel', 1
4), ('episodes', 14), ('watched', 14), ('only', 14), ('acting', 14), ('u',
14), ('by', 14), ('pyaar', 14), ('want', 14), ('out', 13), ('up', 13), ('r
```

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)