Opinion Mining + Sentiment Classification :

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

In [3]:

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

Out[3]:

	Unnamed: 0
0	REVIEWS OF ASPIRANTS
1	This series is really inspiring, good message,...
2	Just amazing. this series is just real it coul...
3	One of the best series I ever seen. I like the...
4	So involving and absorbing. Amazing job ... Al...
...	...
2805	Sandip Bhiyya is an emotion. The way he descri...
2806	Aspirants is not a series,it's a reflection of...
2807	Mature actors , no stars but pure talent for c...
2808	Go and watch this another masterpiece from tvf...
2809	What a Series , Actors and Direction. Unfortun...

2810 rows × 1 columns

In [4]:

```
r2_df
```

Out[4]:

	Unnamed: 0
0	NaN
1	REVIEWS OF KOTA FACTORY
2	NaN
3	KOTA FACTORY\nI have always felt the story of ...
4	I was in the middle of binge-watching all the ...
...	...
586	masterpiece ....
587	When episode 6 will come ?
588	A web series for students
589	Superbbbbbbb......no words left
590	Should more louder on conclusion

591 rows × 1 columns

In [5]:

```
r3_df
```

Out[5]:

```
                                          Unnamed: 0
  0                                              NaN
  1                                REVIEWS OF KAAFIR
  2                                              NaN
  3     Purity, Goodness, love touches you, gets insid...
  4     Kaafir is an intense and a series based on hum...
...                                              ...
444                   I wish Season two will be better ♡
445   Zaberdast drama Zaberdast story Zaberdast dial...
446           It will make you cry specially the 8th episode.
447                        Don't waste time just do watch it
448             Itz get me inside the story.....i felt all fee...

449 rows × 1 columns
```

In [6]:

```
r4_df
```

Out[6]:

```
                                           Unnamed: 0
  0                                               NaN
  1                       REVIEWS OF MADE IN HEAVEN
  2                                               NaN
  3          Cant thank enough to Zoya for this beautiful p...
  4     The web series, Made In heaven was a conglomer...
...                                               ...
573                 SENSATIONAL AND ENGAGING DRAMA
574                              Can't get enough of it.
575                                 To many emotions...
576                                      FULL BAKWAS
577                                This is pathbreakinh

578 rows × 1 columns
```

In [7]:

```
r5_df
```

Out[7]:

```
                                          Unnamed: 0
   0                                             NaN
   1                       REVIEWS OF LITTLE THINGS
   2                                             NaN
   3   This has gotta be the BEST Indian show on a st...
   4    Since the covid started me and my girlfriend a...
 ...                                             ...
 302              Favoriteeeeeeeeeee ♥♥♡♥♡♥
 303                                     It's boring
 304                           Season 3 is the best...
 305                bratay because i got a chrush on her
 306                         I love the show♥♥♥♥♥♥

307 rows × 1 columns
```

In [8]:

```
r6_df
```

Out[8]:

```
                                          Unnamed: 0
   0                                             NaN
   1                        REVIEWS OF LAAKHON MAI EK
   2                                             NaN
   3   Lakhon Mein Ek kind of ShowKnown for his subtl...
   4      Sweata Tripathi is just magnificent. She has s...
 ...                                             ...
 255                                      must watch
 256                       Excellent script and acting
 257                                     Must watch!
 258    Mind numbingly boring first season with no con...
 259    don't know why it has such low rating. must wa...

260 rows × 1 columns
```

In [9]:

```
r7_df
```

Out[9]:

	Unnamed: 0
0	NaN
1	REVIEWS OF AFSOS
2	NaN
3	Were you able to find the immortal man/woman ?...
4	Indian audience deserves better on OTT. This s...
...	...
215	Kashyap Legend Grows!!!
216	Afsos......u know wot...evan cun weyt. ...Nope...
217	It's underrated
218	Unwanted smile on ur face
219	Waiting for season 2

220 rows × 1 columns

In [10]:

```
r8_df
```

Out[10]:

	Unnamed: 0
0	NaN
1	REVIEWS OF SELECTION DAY
2	NaN
3	Selection Day Season 1 was good for the most p...
4	Selection day isn't perfect but quite great co...
...	...
232	It's interesting
233	waiting for the 2nd season!
234	hi a am sidhart g
235	Not up to the mark.
236	more episodes please!!

237 rows × 1 columns

In [11]:

```
r9_df
```

Out[11]:

	Unnamed: 0
0	REVIEWS OF MASABA MASABA
1	NaN
2	I could have liked it little more If Masaba wo...
3	Simplicity, honesty and courage. I think the t...
4	Nice and entertaining, its a potpourri materia...
...	...
446	Great direction ...never going overboard...exc...
447	I wish I could watch more episodes.. Masaba is...
448	The series looks is very freshmasaba looks goo...
449	Guys please fo watch this. You will never bore...
450	#sanjuzzreviews\nBased on the life of one of I...

451 rows × 1 columns

In [12]:

```
r10_df
```

Out[12]:

	Unnamed: 0
0	NaN
1	REVIEWS OF FOUR MORE SHOTS PLEASE!
2	NaN
3	Season one is miles ahead of 2. The worst aspe...
4	Four More Shots Please (season 2)\n\nFour guys...
...	...
997	Where's 2nd part.....?
998	Waiting for more shots.. season 3....?
999	I want season 2!!
1000	One word "Worst!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!...
1001	Story pattern is same as season 1.

1002 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	REVIEWS OF ASPIRANTS
1	This series is really inspiring, good message,...
2	Just amazing. this series is just real it coul...
3	One of the best series I ever seen. I like the...
4	So involving and absorbing. Amazing job ... Al...
...	...
6900	Where's 2nd part.....?
6901	Waiting for more shots.. season 3....?
6902	I want season 2!!
6903	One word "Worst!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!...
6904	Story pattern is same as season 1.

6905 rows × 1 columns

In [15]:

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

In [16]:

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

Out[16]:

	transcript
0	REVIEWS OF ASPIRANTS
1	This series is really inspiring, good message,...
2	Just amazing. this series is just real it coul...
3	One of the best series I ever seen. I like the...
4	So involving and absorbing. Amazing job ... Al...
...	...
6900	Where's 2nd part.....?
6901	Waiting for more shots.. season 3....?
6902	I want season 2!!
6903	One word "Worst!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!...
6904	Story pattern is same as season 1.

6905 rows × 1 columns

In [25]:

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

Out[25]:

	transcript
1066	Great show!
4690	It's very commonly known thing...... Very rela...
6754	Season 2 is a disaster , Nothing relatable , U...
2263	Series shows the truth about what the UPSC Asp...
5701	Many of us cannot keeping up with Kardashian. ...
2735	Tvf always make unique content.i love tvf .all...
396	Hats off to the makers.
6725	Nice wonderful series 🔥🔥😎 😍 🤩
5914	To be honest, when i first saw season 1, it wa...
310	Masterpiece, Must Watch, Very Inspirational..

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 [26]:

```
# 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 [27]:

```
# 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[27]:

```
                                         transcript
   0                            reviews of aspirants
   1     this series is really inspiring good message l...
   2        just amazing this series is just real it could...
   3       one of the best series i ever seen i like the ...
   4   so involving and absorbing amazing job all ac...
 ...                                             ...
6900                                     wheres part
6901                    waiting for more shots season
6902                                   i want season
6903                                  one word worst
6904                  story pattern is same as season
```

6905 rows × 1 columns

In [28]:

```
# 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 [29]:

```
# 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[29]:

	transcript
0	reviews of aspirants
1	this series is really inspiring good message l...
2	just amazing this series is just real it could...
3	one of the best series i ever seen i like the ...
4	so involving and absorbing amazing job all ac...
...	...
6900	wheres part
6901	waiting for more shots season
6902	i want season
6903	one word worst
6904	story pattern is same as season

6905 rows × 1 columns

In [30]:

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

Out[30]:

```
transcript    good script and good direction tvf always give...
Name: 2694, dtype: object
```

In [31]:

```
# 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 [32]:

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

In [33]:

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

Out[33]:

	transcript
0	reviews of aspirants
1	this series is really inspiring good message l...
2	just amazing this series is just real it could...
3	one of the best series i ever seen i like the ...
4	so involving and absorbing amazing job all act...
...	...
6900	wheres part
6901	waiting for more shots season
6902	i want season
6903	one word worst
6904	story pattern is same as season

6905 rows × 1 columns

In [34]:

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

Out[34]:

	transcript
5813	good series
1328	everything good in this web seriescast perfect...
652	motivational series
1810	very much clean easy to digest no over acting ...
3567	its a must and should watch web series loved i...
1708	its a good series talking about upsc aspirants...

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:

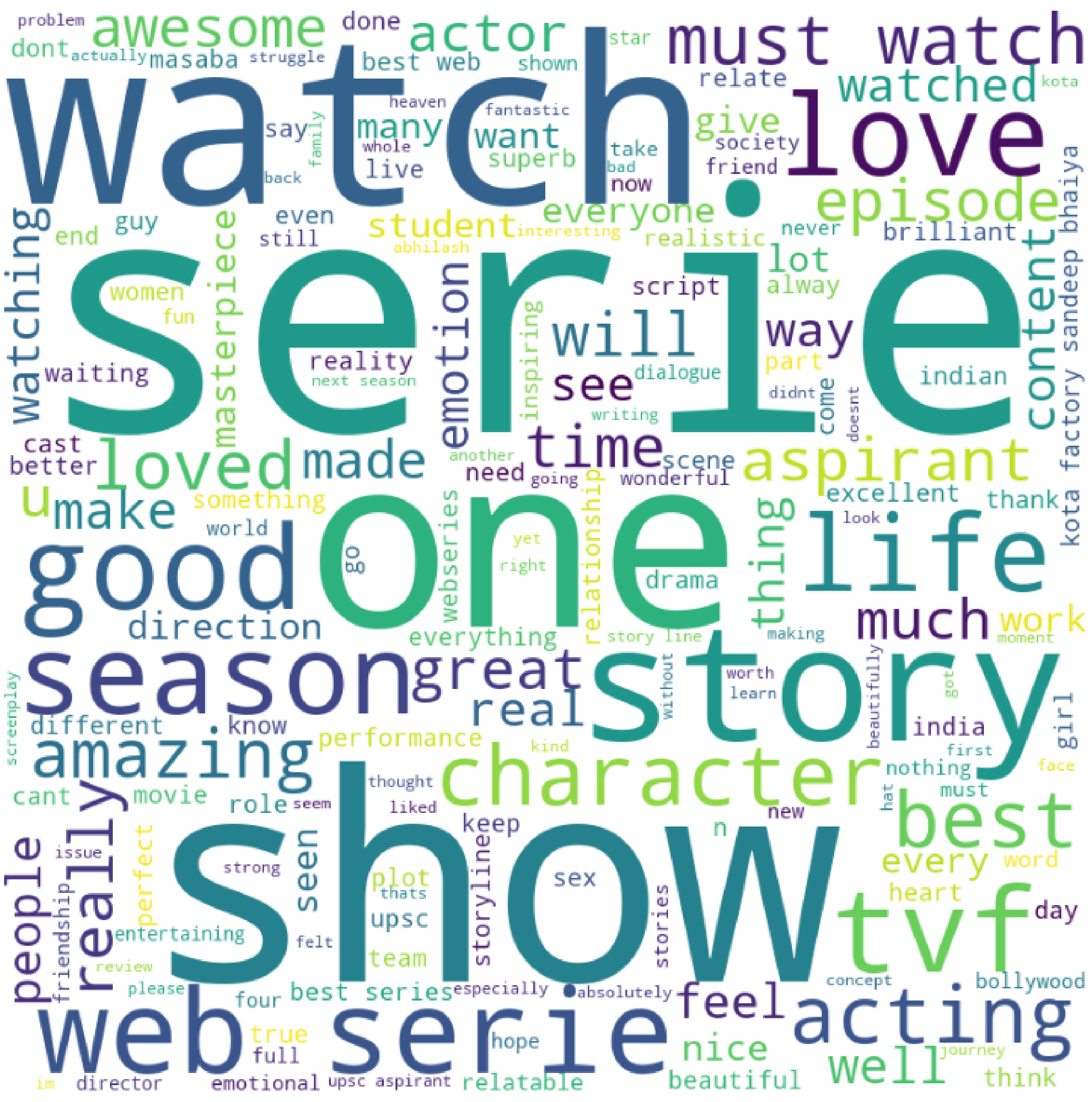
1. Corpus - a collection of text
2. Document-Term Matrix - word counts in matrix format

Corpus

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

In [35]:

```
# 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 [36]:

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

```
{"we're", 'these', 'since', 'out', 'k', 'http', 'not', "they're", 'to', 'dow
n', 'your', 'under', 'yours', 'such', 'also', 'nor', 'ours', 'whom', 'furthe
r', 'his', 'however', "where's", 'they', 'does', "doesn't", 'having', "i'l
l", "she'd", "what's", "she's", 'so', 'after', 'yourself', 'has', 'otherwis
e', 'because', "aren't", "they'd", 'once', 'where', "when's", 'com', 'than',
'again', 'both', 'you', 'as', 'any', 'been', "haven't", 'below', "you're",
'most', 'www', 'about', "won't", "he'll", 'between', 'yourselves', 'off', "s
houldn't", 'doing', "she'll", "i've", 'ought', 'and', 'ourselves', 'too', 'h
e', "they've", 'own', "can't", "here's", 'had', 'for', 'if', 'did', "we've",
'being', 'ever', 'i', "we'd", "hadn't", "wasn't", "he'd", 'during', 'an', 'h
erself', "i'd", 'be', 'we', "wouldn't", 'himself', 'are', 'theirs', "could
n't", 'itself', 'few', 'all', "mustn't", 'only', "we'll", 'cannot', 'else',
'their', 'just', "hasn't", 'r', 'hers', 'like', "you'd", 'more', 'do', 'ho
w', 'here', "weren't", 'it', "let's", 'until', 'was', 'in', 'very', 'throug
h', 'what', "don't", 'but', 'get', 'then', 'before', 'therefore', "he's", 'i
nto', 'other', 'themselves', "isn't", "you've", "didn't", 'were', 'shall',
'when', 'against', "it's", "there's", 'over', 'each', "i'm", 'up', 'who', "t
hat's", 'from', 'this', 'would', 'of', "shan't", 'is', 'could', 'myself', 'a
m', "you'll", "who's", 'some', 'have', 'my', 'by', "they'll", 'which', 'a',
'should', 'him', 'why', 'same', 'our', 'that', 'me', 'at', 'with', 'those',
'while', "why's", 'she', 'them', 'the', 'above', 'her', 'no', "how's", 'ther
e', 'hence', 'can', 'or', 'on', 'its'}
```

In [37]:

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

Out[37]:

```
'reviews of aspirants this series is really inspiring good message light h
earted and sandeep bhaiya by sunny hinduja really he is getting appreciati
on what he deserves sandeep bhaiya ko dekhke sach me rona aajata hai where
performance by naveen kasturia as abhilash abhilash thayipal as sk and shi
vankit singh parihar as guri really trio was really brilliant and superb a
s like dil chahta hai and znmd sahi main sk tu dil chahta hai ka saif ali
khan hai just amazing this series is just real it could ever get everythin
g is amazing hats off to the team if you havent watched it yet then i woul
d really suggest this banger series one of the best series i ever seen i l
ike the way the characters played there roles this show is too much motiva
tional for everyone especially the character of sandeep bhaiya was too goo
d so involving and absorbing amazing job all actors have done an amazing j
ob indian film industry barring few theatre actors shame on you so called
hollow commercialized cinema clowns no hiding public is equally responsibl
e for death of indian cinema for someone who has spent prime of their life
in the system of competitive exams this is so relateable the drama between
friends girlfriends gives snippets into the lives of the individuals what
all they go through over the top advertising of unacadamy is a long tooth
```

In [38]:

```
# 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', 7368), ('and', 5294), ('of', 4089), ('a', 3643), ('to', 3499),
('is', 3492), ('series', 2994), ('i', 2673), ('it', 2623), ('in', 2383),
('this', 2378), ('for', 1836), ('you', 1336), ('watch', 1293), ('with', 12
15), ('show', 1204), ('all', 1141), ('are', 1111), ('its', 1095), ('that',
1076), ('best', 1066), ('was', 1012), ('story', 997), ('but', 975), ('hav
e', 947), ('one', 934), ('very', 889), ('so', 882), ('good', 866), ('web',
842), ('not', 839), ('just', 838), ('season', 816), ('on', 795), ('tvf', 7
85), ('like', 767), ('life', 725), ('by', 717), ('as', 700), ('love', 69
4), ('be', 644), ('acting', 628), ('amazing', 624), ('great', 622), ('mor
e', 607), ('must', 585), ('they', 584), ('an', 559), ('has', 544), ('fro
m', 540), ('their', 533), ('really', 529), ('every', 525), ('loved', 503),
('what', 491), ('my', 482), ('will', 474), ('which', 463), ('about', 441),
('can', 438), ('well', 425), ('time', 407), ('awesome', 397), ('aspirant
s', 395), ('indian', 392), ('who', 391), ('we', 391), ('me', 389), ('suc
h', 389), ('ever', 378), ('watching', 378), ('much', 373), ('how', 366),
('at', 365), ('real', 358), ('there', 353), ('or', 353), ('characters', 34
8), ('character', 348), ('no', 345), ('your', 345), ('made', 333), ('if',
325), ('too', 312), ('episode', 308), ('content', 299), ('watched', 298),
('up', 296), ('also', 291), ('some', 289), ('after', 279), ('only', 279),
```

In [39]:

```
# 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 > 1075]
add_stop_words
```

Out[39]:

```
['the',
 'and',
 'of',
 'a',
 'to',
 'is',
 'series',
 'i',
 'it',
 'in',
 'this',
 'for',
 'you',
 'watch',
 'with',
 'show',
 'all',
 'are',
 'its',
 'that']
```

In [40]:

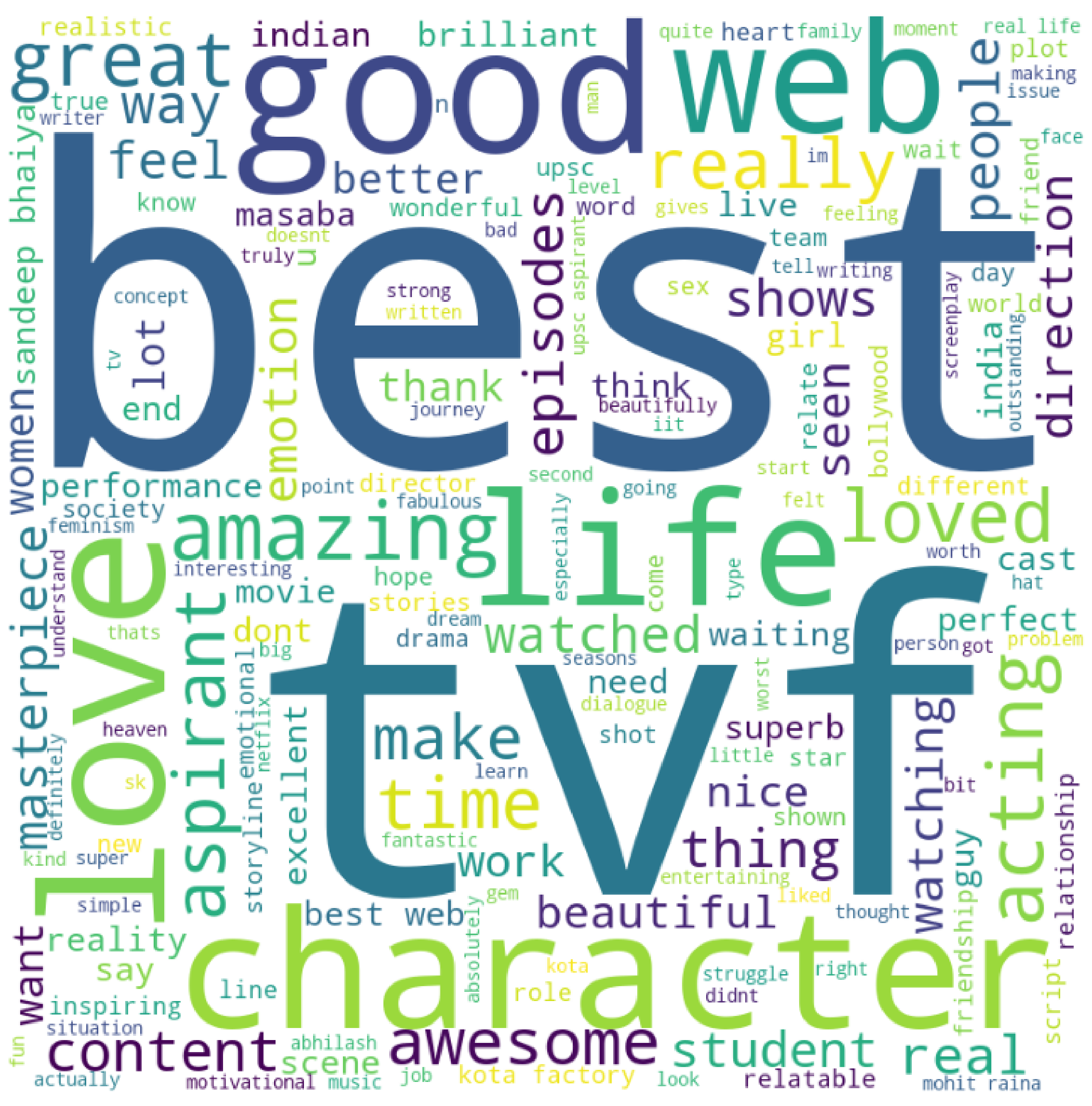
```
#adding more stopwords for better analysis
from sklearn.feature_extraction import text
additional_stop_words = text.ENGLISH_STOP_WORDS
print (additional_stop_words)
```

```
frozenset({'two', 'sixty', 'latter', 'thru', 'your', 'under', 'although', 'a
lso', 'whom', 'detail', 'across', 'further', 'whereafter', 'due', 'again',
'both', 'several', 'any', 'throughout', 'seem', 'empty', 'bottom', 'yourselv
es', 'off', 'whatever', 'something', 'call', 'and', 'beforehand', 'keep', 't
oo', 'own', 'un', 'within', 'four', 'if', 'enough', 'perhaps', 'an', 'hersel
f', 'others', 'eight', 'full', 'seemed', 'few', 'another', 'around', 'how',
'it', 'but', 'wherever', 'hasnt', 'into', 'thin', 'becomes', 'rather', 'co',
'each', 'up', 'from', 'six', 'this', 'elsewhere', 'whoever', 'could', 'eg',
'someone', 'some', 'even', 'ie', 'him', 'mill', 'me', 'serious', 're', 'whil
e', 'she', 'them', 'cant', 'there', 'move', 'hence', 'can', 'third', 'or',
'eleven', 'see', 'down', 'wherein', 'noone', 'almost', 'together', 'after',
'otherwise', 'because', 'once', 'moreover', 'sometime', 'upon', 'about', 'we
ll', 'whole', 'meanwhile', 'now', 'toward', 'part', 'nine', 'had', 'next',
'bill', 'i', 'himself', 'all', 'only', 'else', 'their', 'sincere', 'hers',
'became', 'in', 'interest', 'get', 'whenever', 'none', 'neither', 'anyway',
'already', 'thus', 'would', 'behind', 'somehow', 'should', 'might', 'often',
'yet', 'those', 'whereby', 'her', 'top', 'amoungst', 'ltd', 'its', 'never',
'whereupon', 'these', 'every', 'de', 'since', 'to', 'yours', 'hereby', 'no
r', 'ours', 'beside', 'his', 'hereupon', 'they', 'back', 'though', 'onto',
'so', 'has', 'cry', 'less', 'per', 'three', 'where', 'you', 'as', 'been', 'a
nyhow', 'one', 'put', 'below', 'most', 'whence', 'between', 'name', 'ourselv
es', 'for', 'done', 'system', 'herein', 'ten', 'becoming', 'except', 'etc',
'during', 'latterly', 'be', 'are', 'mostly', 'seems', 'found', 'cannot', 'u
s', 'do', 'here', 'thick', 'until', 'very', 'through', 'amount', 'anywhere',
'everyone', 'whereas', 'then', 'before', 'twelve', 'made', 'take', 'go', 'wh
en', 'amongst', 'who', 'via', 'may', 'myself', 'namely', 'front', 'my', 'des
cribe', 'always', 'which', 'seeming', 'same', 'that', 'thereafter', 'at', 'b
eyond', 'nobody', 'twenty', 'must', 'whether', 'out', 'not', 'hereafter', 's
uch', 'sometimes', 'first', 'anything', 'however', 'nothing', 'much', 'sid
e', 'yourself', 'thereupon', 'five', 'than', 'least', 'nevertheless', 'somew
here', 'besides', 'con', 'anyone', 'everywhere', 'fill', 'fifty', 'he', 'fif
teen', 'couldnt', 'alone', 'towards', 'give', 'former', 'being', 'ever', 'in
c', 'fire', 'nowhere', 'we', 'indeed', 'itself', 'formerly', 'without', 'mor
e', 'was', 'what', 'along', 'will', 'therefore', 'thence', 'therein', 'othe
r', 'show', 'themselves', 'thereby', 'were', 'last', 'against', 'whose', 'fi
nd', 'over', 'forty', 'of', 'is', 'whither', 'am', 'many', 'among', 'have',
'by', 'become', 'a', 'please', 'why', 'still', 'our', 'everything', 'hundre
d', 'with', 'afterwards', 'either', 'the', 'above', 'no', 'mine', 'on'})
```

In [41]:

```
# 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 [42]:

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

```
['two', 'sixty', 'latter', 'thru', 'your', 'under', 'although', 'also', 'who
m', 'detail', 'across', 'further', 'whereafter', 'due', 'again', 'both', 'se
veral', 'any', 'throughout', 'seem', 'empty', 'bottom', 'yourselves', 'off',
'whatever', 'something', 'call', 'and', 'beforehand', 'keep', 'too', 'own',
'un', 'within', 'four', 'if', 'enough', 'perhaps', 'an', 'herself', 'other
s', 'eight', 'full', 'seemed', 'few', 'another', 'around', 'how', 'it', 'bu
t', 'wherever', 'hasnt', 'into', 'thin', 'becomes', 'rather', 'co', 'each',
'up', 'from', 'six', 'this', 'elsewhere', 'whoever', 'could', 'eg', 'someon
e', 'some', 'even', 'ie', 'him', 'mill', 'me', 'serious', 're', 'while', 'sh
e', 'them', 'cant', 'there', 'move', 'hence', 'can', 'third', 'or', 'eleve
n', 'see', 'down', 'wherein', 'noone', 'almost', 'together', 'after', 'other
wise', 'because', 'once', 'moreover', 'sometime', 'upon', 'about', 'well',
'whole', 'meanwhile', 'now', 'toward', 'part', 'nine', 'had', 'next', 'bil
l', 'i', 'himself', 'all', 'only', 'else', 'their', 'sincere', 'hers', 'beca
me', 'in', 'interest', 'get', 'whenever', 'none', 'neither', 'anyway', 'alre
ady', 'thus', 'would', 'behind', 'somehow', 'should', 'might', 'often', 'ye
t', 'those', 'whereby', 'her', 'top', 'amoungst', 'ltd', 'its', 'never', 'wh
ereupon', 'these', 'every', 'de', 'since', 'to', 'yours', 'hereby', 'nor',
'ours', 'beside', 'his', 'hereupon', 'they', 'back', 'though', 'onto', 'so',
'has', 'cry', 'less', 'per', 'three', 'where', 'you', 'as', 'been', 'anyho
w', 'one', 'put', 'below', 'most', 'whence', 'between', 'name', 'ourselves',
'for', 'done', 'system', 'herein', 'ten', 'becoming', 'except', 'etc', 'duri
ng', 'latterly', 'be', 'are', 'mostly', 'seems', 'found', 'cannot', 'us', 'd
o', 'here', 'thick', 'until', 'very', 'through', 'amount', 'anywhere', 'ever
yone', 'whereas', 'then', 'before', 'twelve', 'made', 'take', 'go', 'when',
'amongst', 'who', 'via', 'may', 'myself', 'namely', 'front', 'my', 'describ
e', 'always', 'which', 'seeming', 'same', 'that', 'thereafter', 'at', 'beyon
d', 'nobody', 'twenty', 'must', 'whether', 'out', 'not', 'hereafter', 'suc
h', 'sometimes', 'first', 'anything', 'however', 'nothing', 'much', 'side',
'yourself', 'thereupon', 'five', 'than', 'least', 'nevertheless', 'somewher
e', 'besides', 'con', 'anyone', 'everywhere', 'fill', 'fifty', 'he', 'fiftee
n', 'couldnt', 'alone', 'towards', 'give', 'former', 'being', 'ever', 'inc',
'fire', 'nowhere', 'we', 'indeed', 'itself', 'formerly', 'without', 'more',
'was', 'what', 'along', 'will', 'therefore', 'thence', 'therein', 'other',
'show', 'themselves', 'thereby', 'were', 'last', 'against', 'whose', 'find',
'over', 'forty', 'of', 'is', 'whither', 'am', 'many', 'among', 'have', 'by',
'become', 'a', 'please', 'why', 'still', 'our', 'everything', 'hundred', 'wi
th', 'afterwards', 'either', 'the', 'above', 'no', 'mine', 'on', 'show', 'se
ason', 'one', 'season', 'watch', 'story', 'bajpayee', 'webserie', 'webserie
s', 'bajpai', 'manoj', 'episode', 'review', 'actor', 'actors', 'the', 'and',
'of', 'a', 'to', 'is', 'series', 'i', 'it', 'in', 'this', 'for', 'you', 'wat
ch', 'with', 'show', 'all', 'are', 'its', 'that', "we're", 'these', 'since',
'out', 'k', 'http', 'not', "they're", 'to', 'down', 'your', 'under', 'your
s', 'such', 'also', 'nor', 'ours', 'whom', 'further', 'his', 'however', "whe
re's", 'they', 'does', "doesn't", 'having', "i'll", "she'd", "what's", "sh
e's", 'so', 'after', 'yourself', 'has', 'otherwise', 'because', "aren't", "t
hey'd", 'once', 'where', "when's", 'com', 'than', 'again', 'both', 'you', 'a
s', 'any', 'been', "haven't", 'below', "you're", 'most', 'www', 'about', "wo
n't", "he'll", 'between', 'yourselves', 'off', "shouldn't", 'doing', "she'l
l", "i've", 'ought', 'and', 'ourselves', 'too', 'he', "they've", 'own', "ca
n't", "here's", 'had', 'for', 'if', 'did', "we've", 'being', 'ever', 'i', "w
e'd", "hadn't", "wasn't", "he'd", 'during', 'an', 'herself', "i'd", 'be', 'w
e', "wouldn't", 'himself', 'are', 'theirs', "couldn't", 'itself', 'few', 'al
l', "mustn't", 'only', "we'll", 'cannot', 'else', 'their', 'just', "hasn't",
'r', 'hers', 'like', "you'd", 'more', 'do', 'how', 'here', "weren't", 'it',
"let's", 'until', 'was', 'in', 'very', 'through', 'what', "don't", 'but', 'g
et', 'then', 'before', 'therefore', "he's", 'into', 'other', 'themselves',
"isn't", "you've", "didn't", 'were', 'shall', 'when', 'against', "it's", "th
ere's", 'over', 'each', "i'm", 'up', 'who', "that's", 'from', 'this', 'woul
d', 'of', "shan't", 'is', 'could', 'myself', 'am', "you'll", "who's", 'som
e', 'have', 'my', 'by', "they'll", 'which', 'a', 'should', 'him', 'why', 'sa
me', 'our', 'that', 'me', 'at', 'with', 'those', 'while', "why's", 'she', 't
hem', 'the', 'above', 'her', 'no', "how's", 'there', 'hence', 'can', 'or',
'on', 'its']
```

Findings

We can clearly see that the word cloud has major chunk of positve reviews(roughly 70%) , some negative reviews (roughly 15%), with some neutral reviews(15%).

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 [43]:

```
# 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[43]:

	transcript	polarity	subjectivity
0	reviews of aspirants	0.000000	0.000000
1	this series is really inspiring good message l...	0.508333	0.629167
2	just amazing this series is just real it could...	0.400000	0.575000
3	one of the best series i ever seen i like the ...	0.475000	0.525000
4	so involving and absorbing amazing job all act...	0.185714	0.502381
...	...	...	...
6900	wheres part	0.000000	0.000000
6901	waiting for more shots season	0.500000	0.500000
6902	i want season	0.000000	0.000000
6903	one word worst	-1.000000	1.000000
6904	story pattern is same as season	0.000000	0.125000

6905 rows × 3 columns

In [44]:

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

Out[44]:

	transcript	polarity	subjectivity
5379	very different and amazing story line must watch	0.300	0.84
3360	sad reality of world but truth reality of kota...	-0.500	1.00
1592	the best best ever tv series made in india it ...	0.650	0.35
268	truly awesome	1.000	1.00
4362	an engaging series pretty closely knit	0.325	0.85

In [45]:

```
#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[45]:

	transcript	polarity	subjectivity	Analysis
0	reviews of aspirants	0.000000	0.000000	neutral
1	this series is really inspiring good message l...	0.508333	0.629167	positive
2	just amazing this series is just real it could...	0.400000	0.575000	positive
3	one of the best series i ever seen i like the ...	0.475000	0.525000	positive
4	so involving and absorbing amazing job all act...	0.185714	0.502381	positive
...	...	...	...	...
6900	wheres part	0.000000	0.000000	neutral
6901	waiting for more shots season	0.500000	0.500000	positive
6902	i want season	0.000000	0.000000	neutral
6903	one word worst	-1.000000	1.000000	negative
6904	story pattern is same as season	0.000000	0.125000	neutral

6905 rows × 4 columns

In [46]:

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

Out[46]:

	transcript	polarity	subjectivity	Analysis
2410	one of the best series based on common struggl...	0.350000	0.400000	positive
4698	on which channel did this come i really love this	0.500000	0.600000	positive
840	what a gem	0.000000	0.000000	neutral
3581	heart touching storymohit raina dia mirza live...	0.318182	0.500000	positive
5729	loved the show masaba definitely has the neena...	0.325000	0.550000	positive
5421	amazing and enjoyable	0.550000	0.750000	positive
6288	its a much watch series this show is the true ...	0.183333	0.283333	positive
2247	best show ever this boost your confidence on h...	1.000000	0.300000	positive
5998	its nice how this series builds upon you go gi...	0.228348	0.642262	positive
3430	the story has the innocence of old child purit...	0.266071	0.451071	positive

In [48]:

```
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(Drama
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(Drama Genre) :

No of Negative Reviews from our Total DataSet(around 10k) -> 530
No of Positive Reviews from our Total DataSet(around 10k) -> 5329
No of  Neutral Reviews from our Total DataSet(around 10k) -> 1046

Percentage of Negative Reviews -> 7.675597393193338 %
Percentage of Positive Reviews -> 77.17595944967415 %
Percentage of Neutral  Reviews -> 15.148443157132514  %
```

Sentiment Findings:

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

```
No of Negative Reviews from our Total DataSet(around 10k) -> 530
No of Positive Reviews from our Total DataSet(around 10k) -> 5329
No of  Neutral Reviews from our Total DataSet(around 10k) -> 1046

Percentage of Negative Reviews -> 7.675597393193338 %
Percentage of Positive Reviews -> 77.17595944967415 %
Percentage of Neutral  Reviews -> 15.148443157132514  %


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

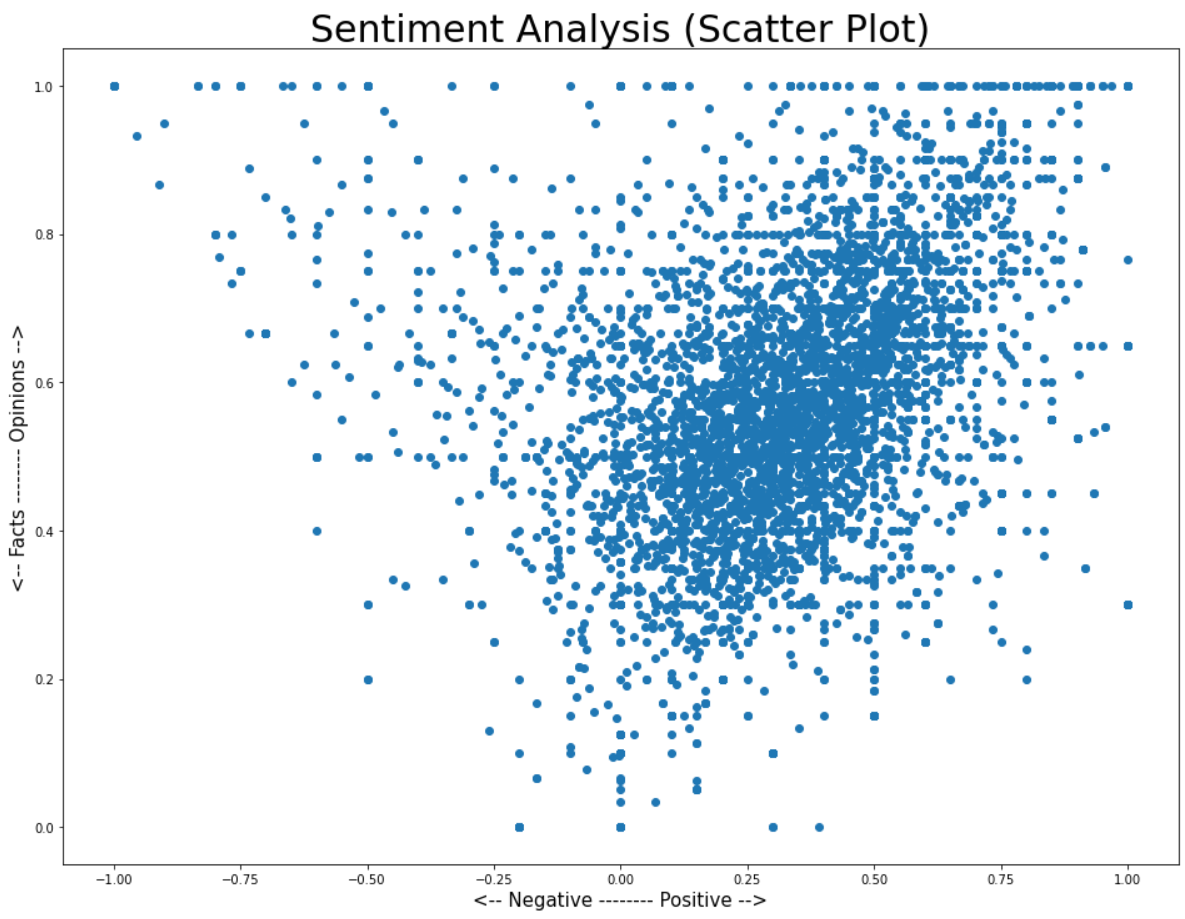
Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When 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.

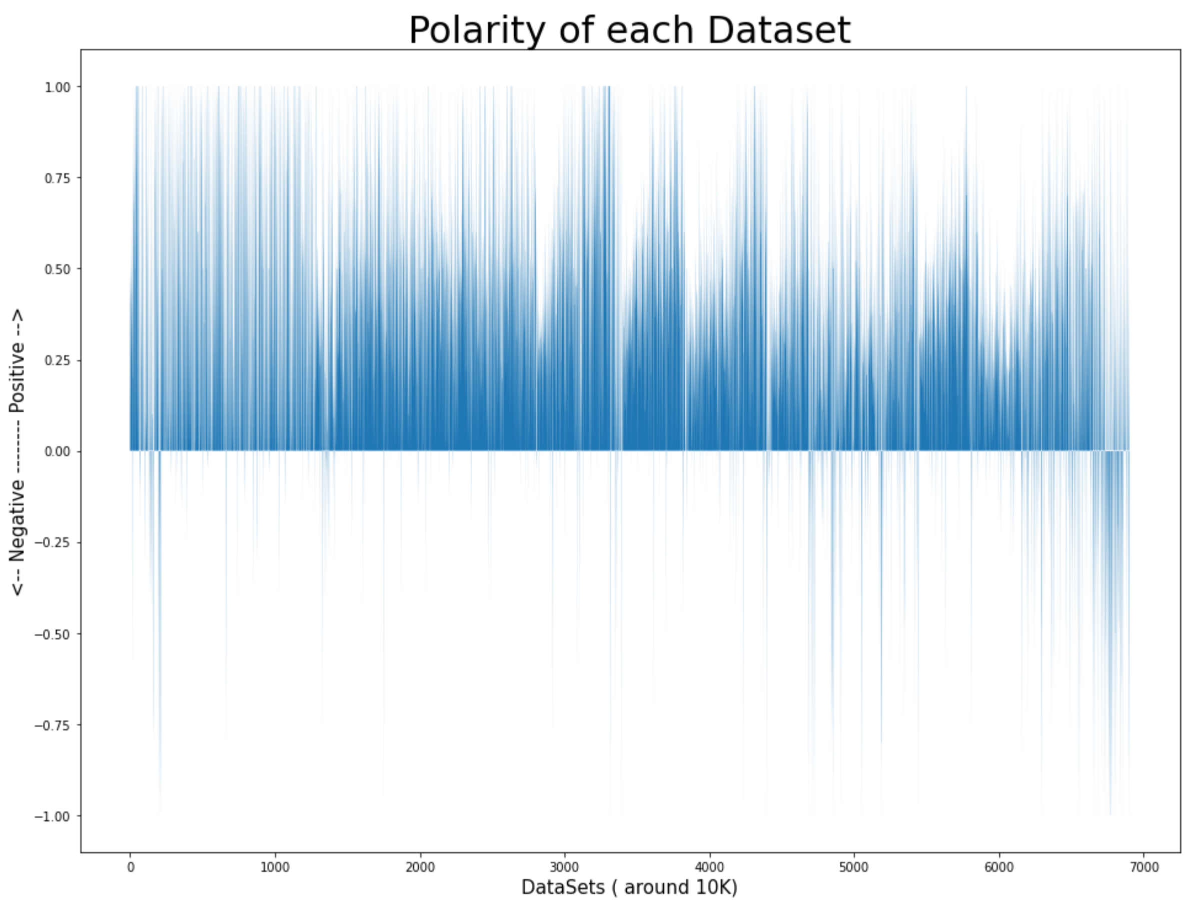
In [49]:

```
# 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()
```



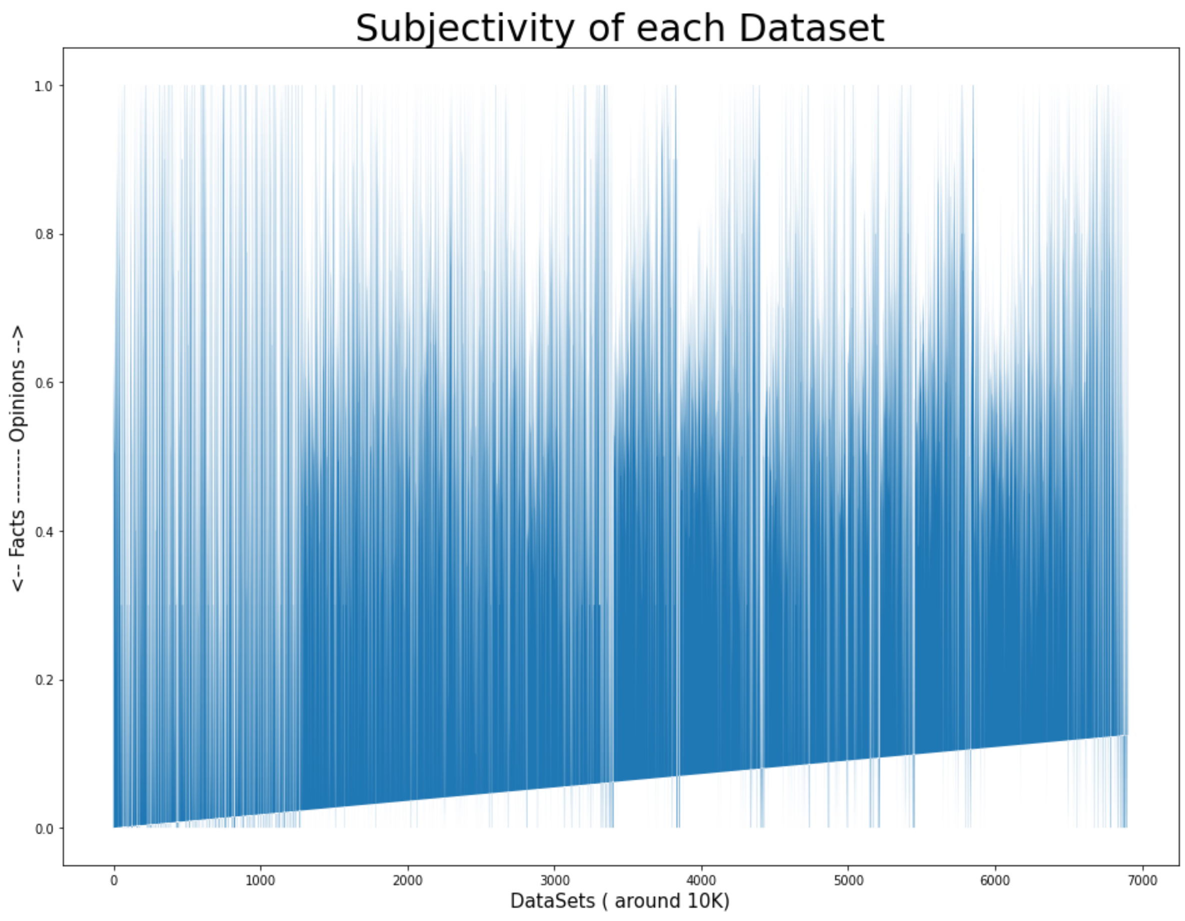
In [51]:

```
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()
```



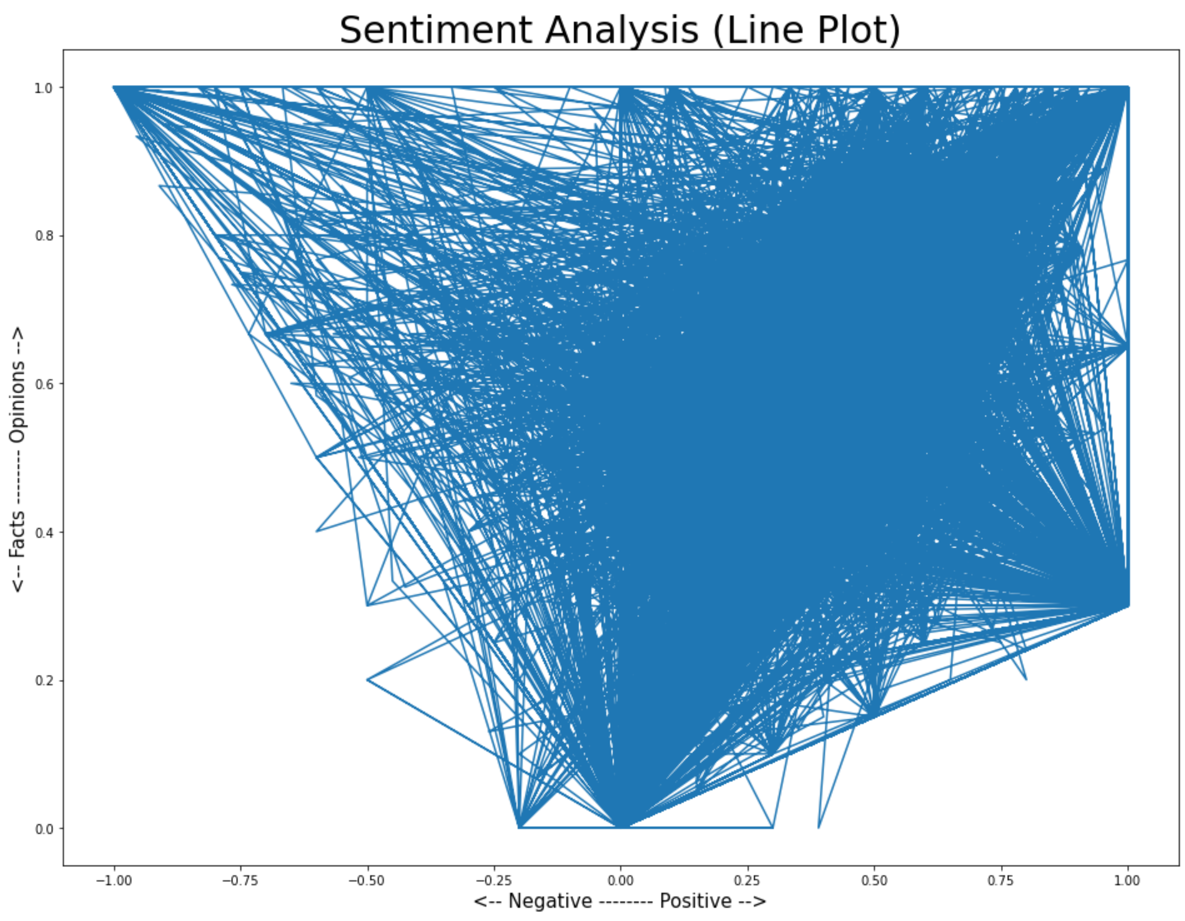
In [52]:

```
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()
```



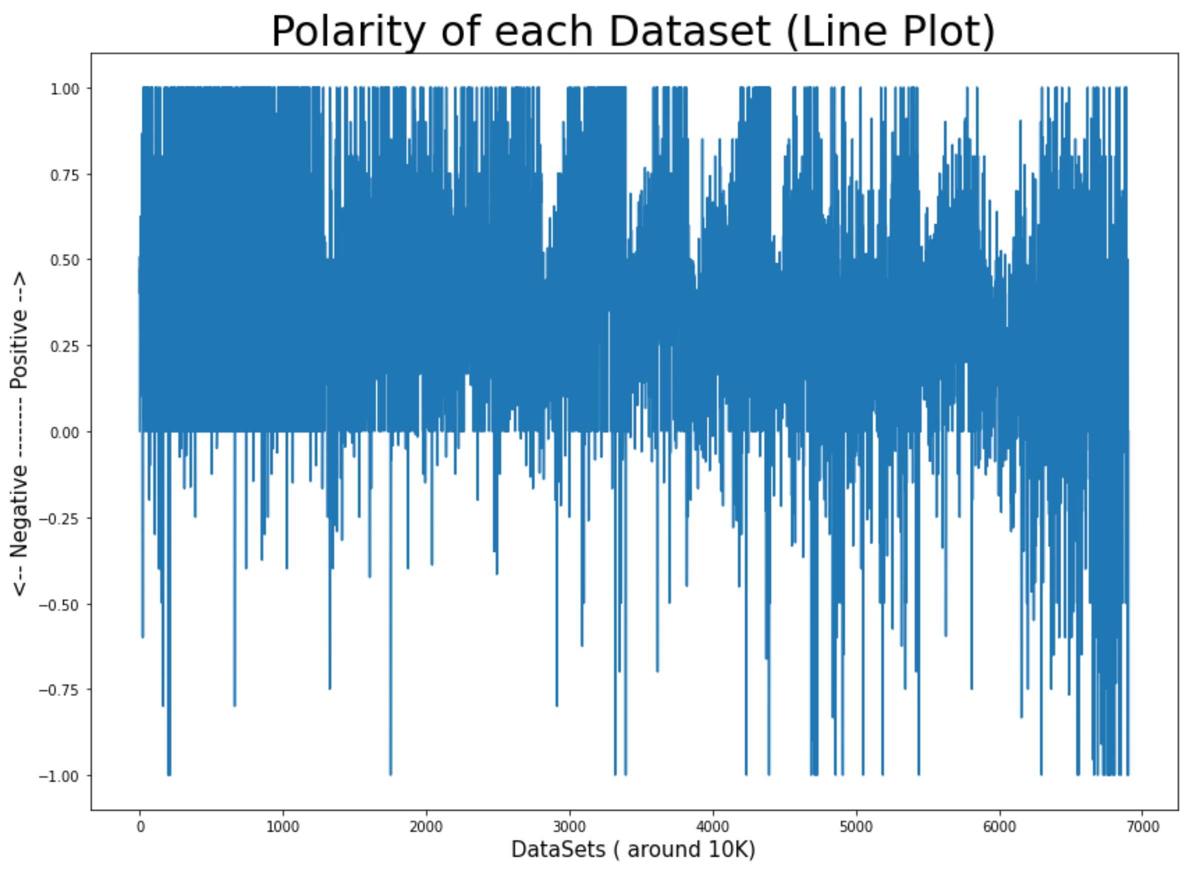
In [53]:

```
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()
```



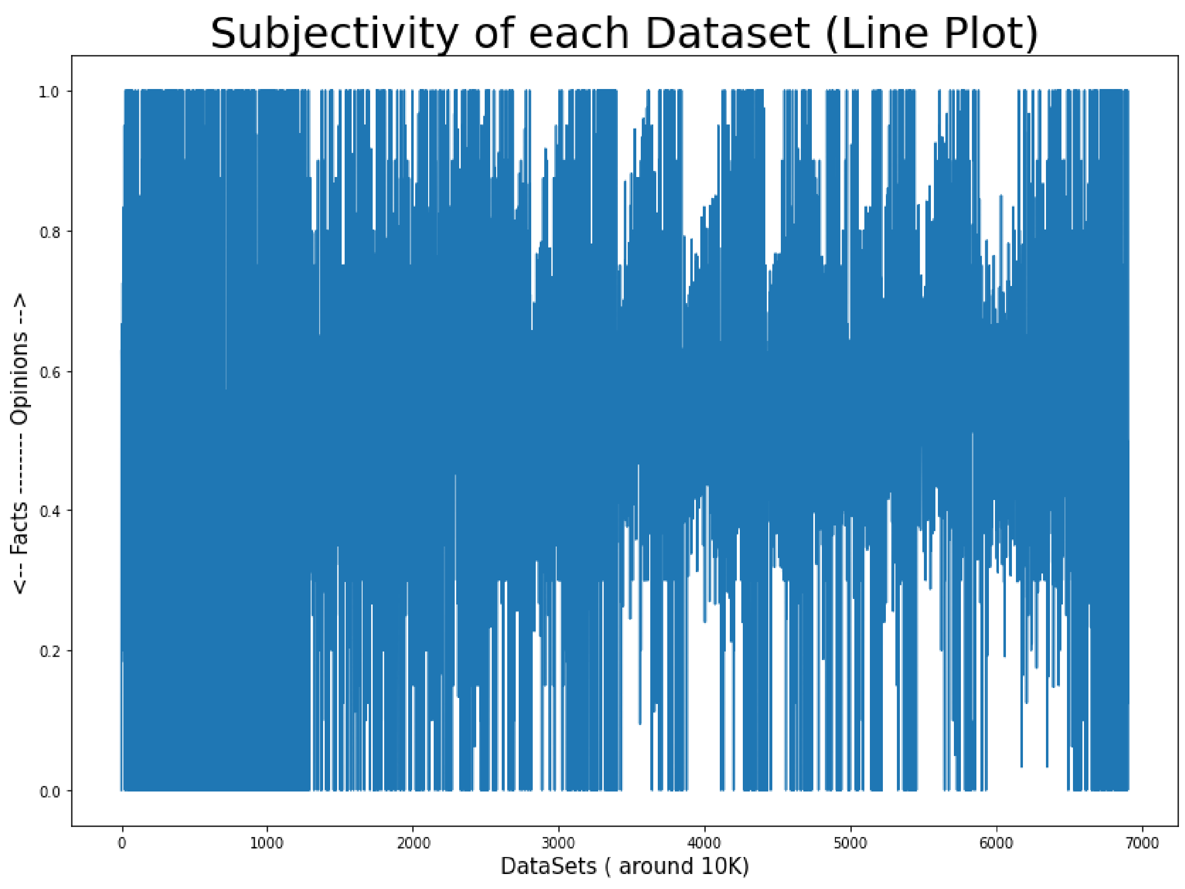
In [54]:

```
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()
```



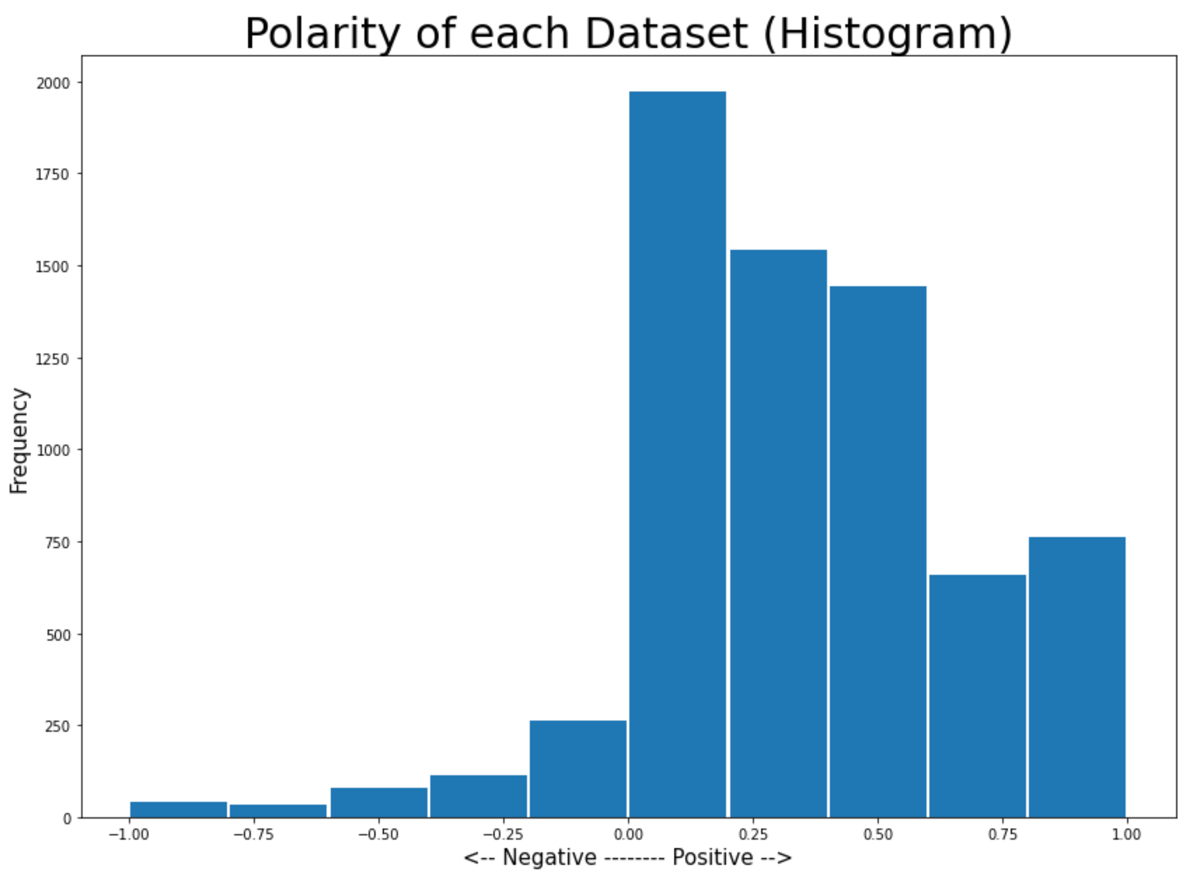
In [55]:

```
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()
```



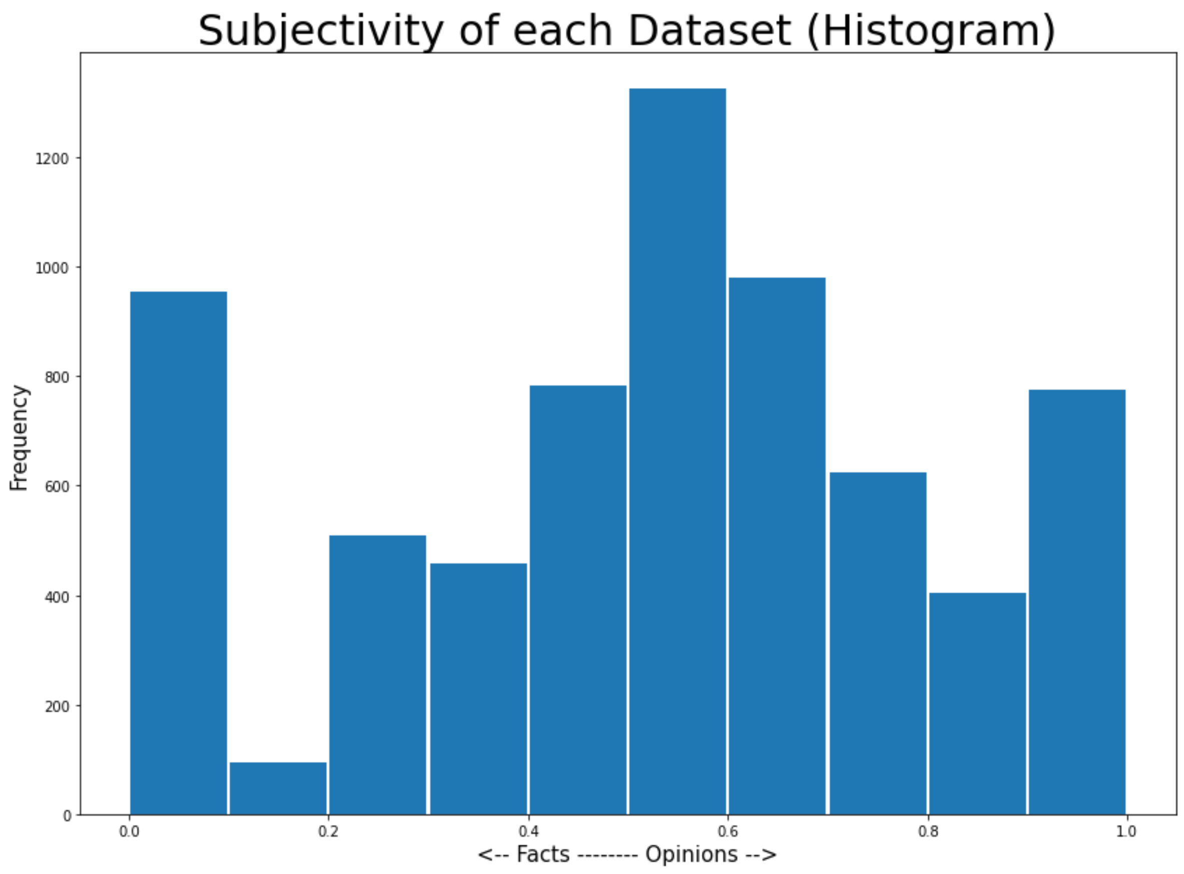
In [56]:

```
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 [57]:

```
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 [58]:

```
data_clean_df
```

Out[58]:

	transcript	polarity	subjectivity	Analysis
0	reviews of aspirants	0.000000	0.000000	neutral
1	this series is really inspiring good message l...	0.508333	0.629167	positive
2	just amazing this series is just real it could...	0.400000	0.575000	positive
3	one of the best series i ever seen i like the ...	0.475000	0.525000	positive
4	so involving and absorbing amazing job all act...	0.185714	0.502381	positive
...	...	...	...	...
6900	wheres part	0.000000	0.000000	neutral
6901	waiting for more shots season	0.500000	0.500000	positive
6902	i want season	0.000000	0.000000	neutral
6903	one word worst	-1.000000	1.000000	negative
6904	story pattern is same as season	0.000000	0.125000	neutral

6905 rows × 4 columns

In [59]:

```
#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 [60]:

```
positive_df
```

Out[60]:

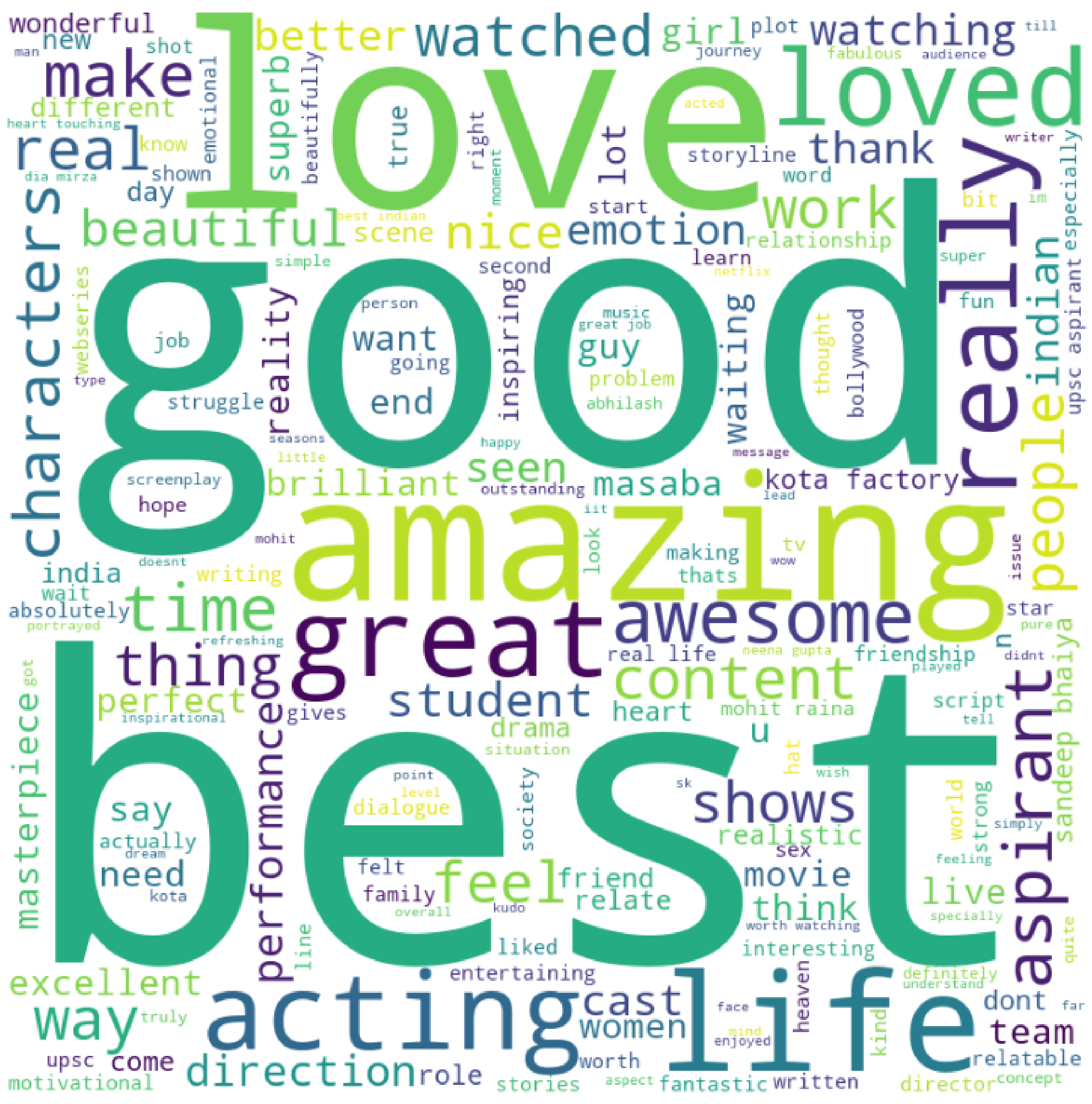
	transcript	polarity	subjectivity	Analysis
1	this series is really inspiring good message l...	0.508333	0.629167	positive
2	just amazing this series is just real it could...	0.400000	0.575000	positive
3	one of the best series i ever seen i like the ...	0.475000	0.525000	positive
4	so involving and absorbing amazing job all act...	0.185714	0.502381	positive
5	for someone who has spent prime of their life ...	0.102000	0.508000	positive
...	...	...	...	...
6881	awesome girl movie	1.000000	1.000000	positive
6889	please upload four more shots web series	0.500000	0.500000	positive
6891	awesome	1.000000	1.000000	positive
6897	story is not good performance great	0.225000	0.675000	positive
6901	waiting for more shots season	0.500000	0.500000	positive

5329 rows × 4 columns

In [64]:

```
# 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','tvf'
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 [65]:

```python
#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 [66]:

```
negative_df
```

Out[66]:

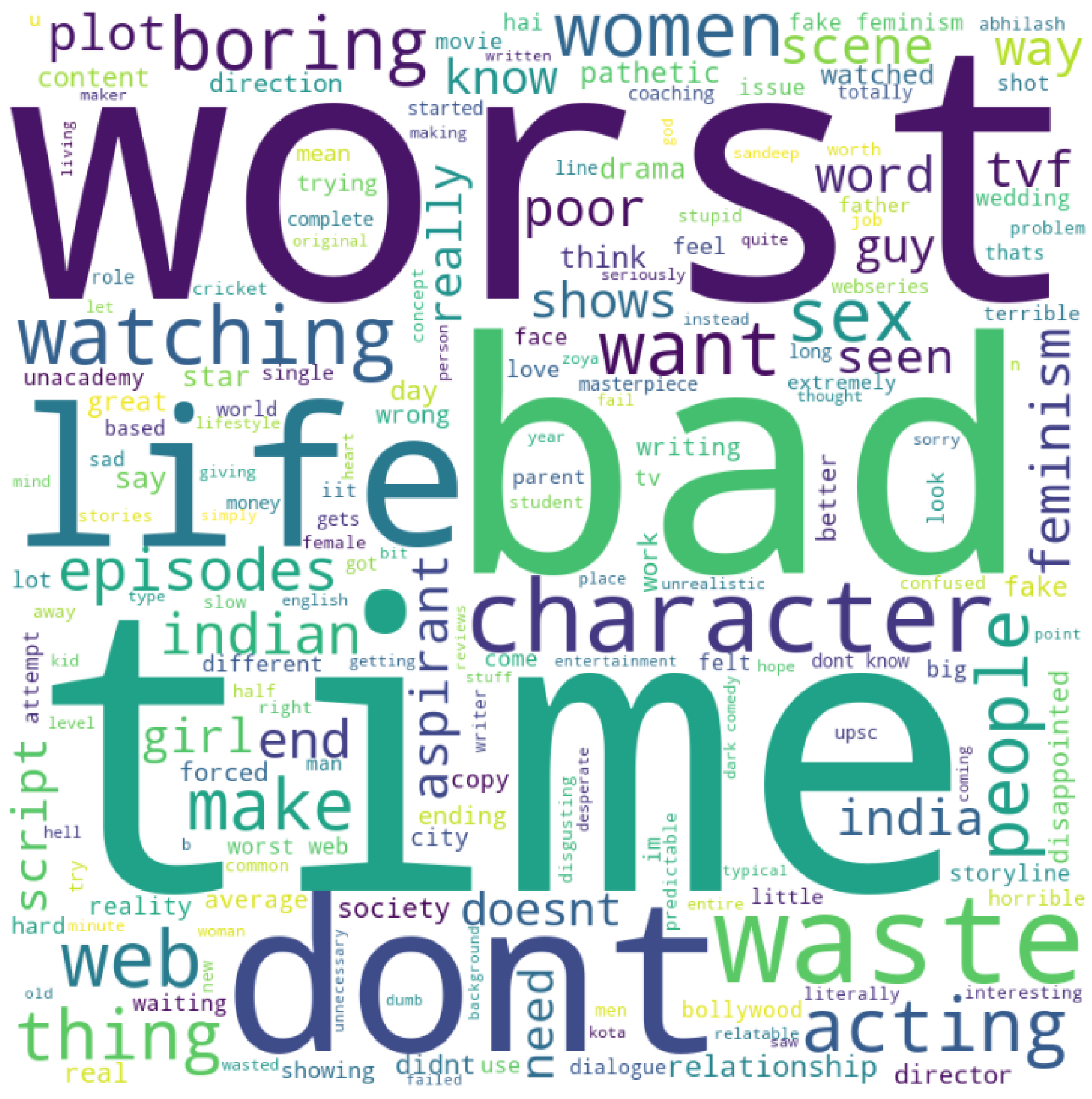
	transcript	polarity	subjectivity	Analysis
21	masterpiece recommended for all upsc aspirants...	-0.600000	0.400000	negative
67	unable to explain my feelings wow	-0.200000	0.750000	negative
71	bollywood is dead tvf is the future masterpiec...	-0.100000	0.262500	negative
106	unacademy ad sucks	-0.300000	0.300000	negative
133	slow plot and selfish friendship from abhilash	-0.400000	0.700000	negative
...	...	...	...	...
6871	nothing new very predictable stuff	-0.061818	0.552273	negative
6884	useless storyno concept	-0.500000	0.200000	negative
6887	when the next episode will update waiting madly	-0.312500	0.500000	negative
6899	worst	-1.000000	1.000000	negative
6903	one word worst	-1.000000	1.000000	negative

530 rows × 4 columns

In [67]:

```
# 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','bajp
stopwords = list(additional_stop_words) + selected_stop_words + add_stop_words + list(STOPW

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

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

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

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

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

plt.show()
```

WordCloud for NEGATVE SENTIMENTS



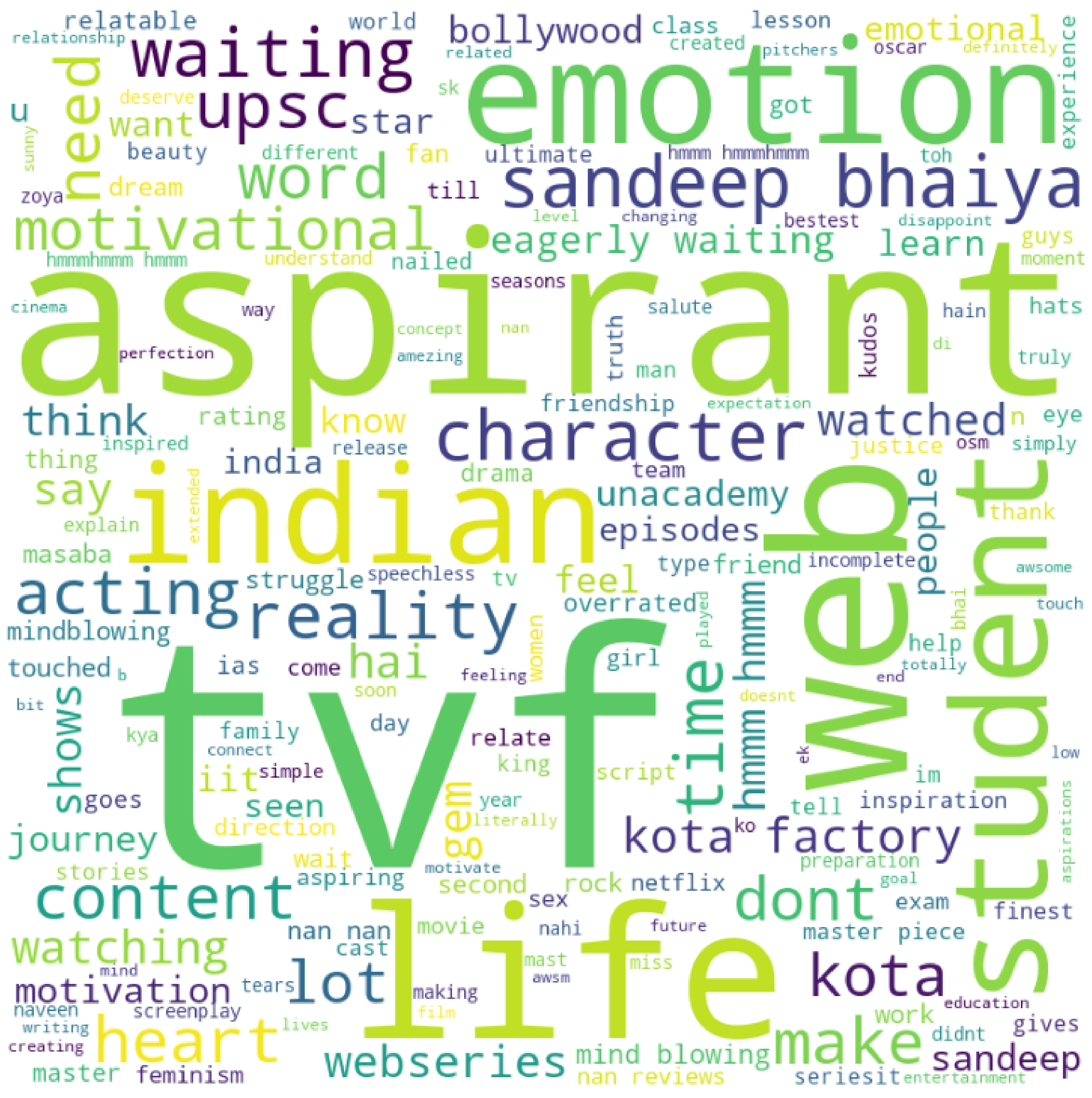
In [68]:

```
#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 [69]:

```
# 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()
```

WordCloud for NEUTRAL SENTIMENTS



Additonal Information

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

In [70]:

```
# 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', 6534), ('and', 4753), ('of', 3542), ('a', 3114), ('to', 3040),
('is', 3025), ('series', 2636), ('i', 2362), ('it', 2303), ('in', 2104),
('this', 2046), ('for', 1569), ('you', 1160), ('watch', 1069), ('best', 10
63), ('with', 1062), ('show', 1039), ('all', 1006), ('are', 969), ('its',
959), ('that', 954), ('was', 893), ('story', 869), ('one', 841), ('but', 8
38), ('good', 836), ('very', 827), ('have', 821), ('so', 805), ('web', 73
7), ('just', 707), ('on', 683), ('love', 683), ('season', 681), ('tvf', 66
0), ('like', 657), ('not', 651), ('by', 630), ('life', 623), ('amazing', 6
21), ('as', 619), ('great', 607), ('more', 586), ('acting', 575), ('be', 5
48), ('really', 511), ('loved', 502), ('they', 492), ('an', 486), ('thei
r', 485), ('has', 483), ('from', 469), ('must', 467), ('every', 454), ('m
y', 413), ('which', 412), ('will', 410), ('awesome', 395), ('what', 392),
('well', 390), ('can', 381), ('about', 374), ('real', 343), ('such', 343),
('we', 342), ('me', 339), ('much', 339), ('who', 339), ('indian', 338),
('aspirants', 326), ('watching', 326), ('ever', 322), ('how', 316), ('char
acters', 315), ('character', 315), ('at', 309), ('time', 308), ('there', 3
05), ('or', 299), ('made', 292), ('your', 289), ('too', 273), ('watched',
269), ('content', 265), ('no', 261), ('if', 257), ('up', 256), ('also', 25
5), ('episode', 255), ('work', 255), ('actors', 254), ('some', 249), ('sho
```

In [71]:

```
# 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', 617), ('and', 415), ('of', 353), ('to', 338), ('a', 322), ('is',
312), ('i', 209), ('in', 202), ('this', 199), ('it', 195), ('series', 18
9), ('not', 132), ('for', 128), ('show', 128), ('with', 111), ('but', 10
5), ('are', 105), ('that', 101), ('was', 97), ('you', 95), ('story', 90),
('watch', 86), ('have', 86), ('its', 84), ('on', 83), ('just', 80), ('wors
t', 80), ('time', 77), ('all', 75), ('like', 72), ('be', 71), ('they', 7
1), ('so', 66), ('no', 64), ('season', 63), ('what', 61), ('dont', 61),
('one', 61), ('very', 61), ('as', 59), ('by', 56), ('bad', 55), ('at', 5
0), ('has', 48), ('about', 48), ('if', 48), ('web', 47), ('life', 47), ('o
nly', 46), ('an', 45), ('waste', 44), ('there', 44), ('from', 43), ('eve
r', 43), ('your', 40), ('or', 40), ('how', 39), ('which', 39), ('feminis
m', 39), ('episode', 38), ('will', 38), ('watching', 37), ('their', 37),
('can', 37), ('my', 36), ('who', 36), ('women', 36), ('such', 34), ('too',
34), ('acting', 33), ('want', 33), ('me', 33), ('much', 32), ('boring', 3
2), ('made', 32), ('sex', 32), ('after', 31), ('every', 30), ('people', 3
0), ('good', 30), ('some', 29), ('we', 29), ('do', 29), ('other', 28), ('i
ndian', 28), ('episodes', 27), ('characters', 27), ('even', 26), ('fake',
26), ('up', 25), ('any', 25), ('then', 23), ('also', 23), ('he', 23), ('we
re', 22), ('know', 22), ('never', 21), ('poor', 21), ('must', 21), ('whe
```

In [72]:

```
# 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', 217), ('a', 207), ('of', 194), ('series', 169), ('is', 155), ('fo
r', 139), ('watch', 138), ('this', 133), ('and', 126), ('it', 125), ('to',
121), ('tvf', 105), ('i', 102), ('must', 97), ('you', 81), ('in', 77), ('s
eason', 72), ('masterpiece', 61), ('all', 60), ('web', 58), ('not', 56),
('life', 55), ('aspirants', 53), ('its', 52), ('just', 51), ('with', 42),
('every', 41), ('have', 40), ('what', 38), ('like', 38), ('story', 38),
('are', 37), ('show', 37), ('sandeep', 36), ('waiting', 35), ('my', 33),
('kota', 33), ('aspirant', 32), ('but', 32), ('one', 32), ('by', 31), ('sh
ould', 30), ('on', 29), ('an', 28), ('from', 28), ('bhaiya', 27), ('india
n', 26), ('will', 26), ('nan', 25), ('be', 25), ('time', 22), ('was', 22),
('as', 22), ('upsc', 21), ('cant', 21), ('they', 21), ('reality', 21), ('t
hat', 21), ('no', 20), ('we', 20), ('if', 20), ('can', 20), ('acting', 2
0), ('only', 20), ('after', 19), ('content', 19), ('about', 19), ('hmmm',
19), ('dont', 18), ('another', 18), ('factory', 18), ('heart', 18), ('emot
ion', 18), ('character', 18), ('our', 18), ('me', 17), ('motivational', 1
7), ('everyone', 17), ('well', 17), ('who', 16), ('need', 16), ('your', 1
6), ('when', 16), ('next', 16), ('gem', 15), ('something', 15), ('hai', 1
5), ('make', 15), ('up', 15), ('student', 15), ('watching', 15), ('episod
e', 15), ('webseries', 14), ('emotions', 14), ('shows', 14), ('or', 14),
```

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)