Sentiment Analysis_WordCloud

@Author : Woojin Park, Nidhi Bhaskar

@Email : woojinpa@andrew.cmu.edu , nidhibha@andrew.cmu.edu

@Status : In-Progress

In [1]:

```
### Import Relevant Libraries
import os
import pandas as pd
import numpy as np
import collections
import datetime as dt
import requests
import json
import re
import time

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from scipy.stats import norm

import string
import re
import nltk
from nltk.util import ngrams
from nltk import pos_tag,word_tokenize
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer,PorterStemmer
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
```

In [2]:

```
### Build a get_date function to convert date format
#### Build a data_creation function to read json data into pandas dataframe

def get_date(created):
    return dt.datetime.fromtimestamp(created)

def data_creation(subreddit) :
    with open('submissions_'+subreddit+'.json') as f:
        data = json.loads("[" +
            f.read().replace("}\n{", "},\n{") +
        "]")
    data =pd.DataFrame(data)
    reddit_data = data[['author','over_18','title','selftext','num_comments', 's
core', 'full_link','created_utc']]
    reddit_data = reddit_data.dropna()
    _timestamp = reddit_data["created_utc"].apply(get_date)
    reddit_data = reddit_data.assign(timestamp = _timestamp)
    reddit_data['over_18'] = reddit_data['over_18'].astype('str')
    reddit_data['subreddit']= subreddit
    reddit_data['title_with_selftext']= reddit_data['title'] + reddit_data['self
text']


    # Do one more extra cleaning : keep updating this part
    reddit_data=reddit_data[~reddit_data['title_with_selftext'].isin([ '[removed
]', '[deleted]',''])]
    subreddit = reddit_data

    return subreddit

def empty_words_clean(text):
    text = text.replace('[removed]','')
    text= text.replace('[deleted]','')
    text= text.replace('\n','')
    return (text)
```

In [3]:

```
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
```

In [4]:

```
### Dataframing 4 subreddit Datasets
SuicideWatch_df = data_creation('SuicideWatch')
depressed_df = data_creation('depressed')
happy_df = data_creation('happy')
selfimprovement_df = data_creation('selfimprovement')

### Concat all 4 dataframes into one merged file
all_subreddit_df = pd.concat([SuicideWatch_df,depressed_df,happy_df,selfimprovem
ent_df])
all_subreddit_df.head(2)
```

Out[4]:

	author	over_18	title	selftext	num_comments	score	
0	DespressoCafe	False	I don't know where to go or what to do. I can'...	Let's make it quick. I'm almost 20. I've been ...	5	1	https://www.reddit.com/r/
1	LifeisCrumbling	False	I'm having an existencial crisis	If I only helped people either as a defense me...	1	1	https://www.reddit.com/r/

In [5]:

```
SuicideWatch_df["title_with_selftext_cleaned"] = SuicideWatch_df["title_with_sel
ftext"].apply(lambda x: empty_words_clean(x))
depressed_df["title_with_selftext_cleaned"] = depressed_df["title_with_selftext"
].apply(lambda x: empty_words_clean(x))
happy_df["title_with_selftext_cleaned"] = happy_df["title_with_selftext"].apply(
lambda x: empty_words_clean(x))
selfimprovement_df["title_with_selftext_cleaned"] = selfimprovement_df["title_wi
th_selftext"].apply(lambda x: empty_words_clean(x))
```

Sentiment Analysis

-Word Cloud

Word Cloud on most frequent Words/Nouns/Adjectives for each subreddit

- Spot the different topics/terms for each subreddit

1.Suicide Watch

In [6]:

```
## All Words
```

In [7]:

```
SuicideWatch_df.head(2)
```

Out[7]:

	author	over_18	title	selftext	num_comments	score	
0	DespressoCafe	False	I don't know where to go or what to do. I can'...	Let's make it quick. I'm almost 20. I've been ...	5	1	https://www.reddit.com/r/
1	LifeisCrumbling	False	I'm having an existencial crisis	If I only helped people either as a defense me...	1	1	https://www.reddit.com/r/

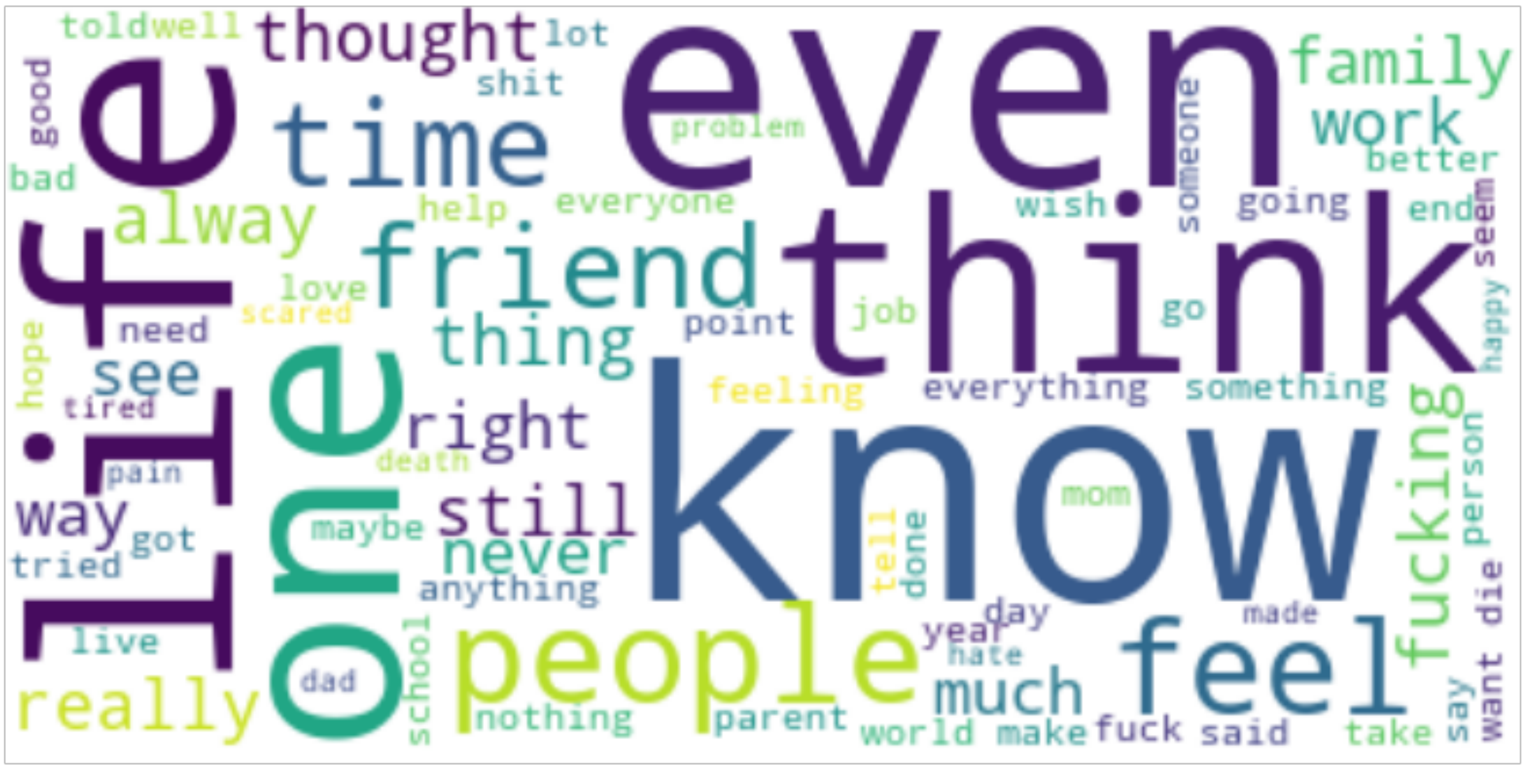
In [8]:

```
cachedStopWords = set(stopwords.words("english"))

####Keep Updating custom stop words
cachedStopWords.update(('nt', 'wo', 're', 'im', 'yall','u','ca','ive', 'wan','na
','gon','nov','x200b','amp',\
                        'wwwyoutubecomwatch','http','vbjkbl5olvm8','lt', 'br', '
gt', 'amp','tsp','tbsp','nbsp', 'le'))

def alpha_filter(w):
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False

## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now
sampleSuicideWatch_list = SuicideWatch_df.sample(frac=0.5, replace=True, random_
state=1)
SuicideWatch_list = sampleSuicideWatch_list['title_with_selftext_cleaned'].tolis
t()
SuicideWatch_list_lower = [tok.lower() for i in SuicideWatch_list for tok in nlt
k.word_tokenize(i)]
nltk_stopwords = set(stopwords.words('english'))
SuicideWatch_list_lower_stop = [x for x in SuicideWatch_list_lower if not x in n
ltk_stopwords]
SuicideWatch_list_lower_stop_pun = [y for y in SuicideWatch_list_lower_stop if n
ot alpha_filter(y)]
SuicideWatch_list_lower_stop_pun_extra = [''.join(x for x in par if x not in str
ing.punctuation) for par in\
                                          SuicideWatch_list_lower_stop_pun]

porter = WordNetLemmatizer()
SuicideWatch_list_lower_stop_pun_extra_lemmatized = []
for a in SuicideWatch_list_lower_stop_pun_extra :
    SuicideWatch_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize(a)
)
SuicideWatch_list = [x for x in SuicideWatch_list_lower_stop_pun_extra_lemmatize
d if not x in cachedStopWords]

# Combining all the posts in the list to one single element to generate the Word
Cloud
SuicideWatch_allWords =(" ").join(SuicideWatch_list)
```

In [57]:

```
print('SuicideWatch All words WordCloud')
# Generating a word cloud image for a sample of SuicideWatch:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Suicide
Watch_allWords)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
SuicideWatch All words WordCloud
```



In [ ]:

```
#Findings:
#pain #tried #shit #fucking #fuck #end #life #hate #die #death #school #help #sc
ared
#positive words mentioned in the negative sense - #good #hope #happy #love #wish
#better #live
#Don't mention a lot about sadness because they have passed that phase.
#The mention of "school" tells us that school students and teens are highly suic
ide driven. Schools need to take
#necessary measures and monitor each students' mental health condition
```

In [10]:

```
# SuicideWatch WordClouds for Nouns and Adjectives
```

In [11]:

```
SuicideWatch_pos = nltk.pos_tag(SuicideWatch_list)
# generate Noun list and adjective
SuicideWatch_NN_list = []
SuicideWatch_AJ_list = []
for i,j in SuicideWatch_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        SuicideWatch_NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        SuicideWatch_AJ_list.append(i)
print('There are',len(SuicideWatch_NN_list),'nouns in the list ')
print('There are',len(SuicideWatch_AJ_list),'adjectives in the list')
```

```
There are 627019 nouns in the list
There are 302035 adjectives in the list
```

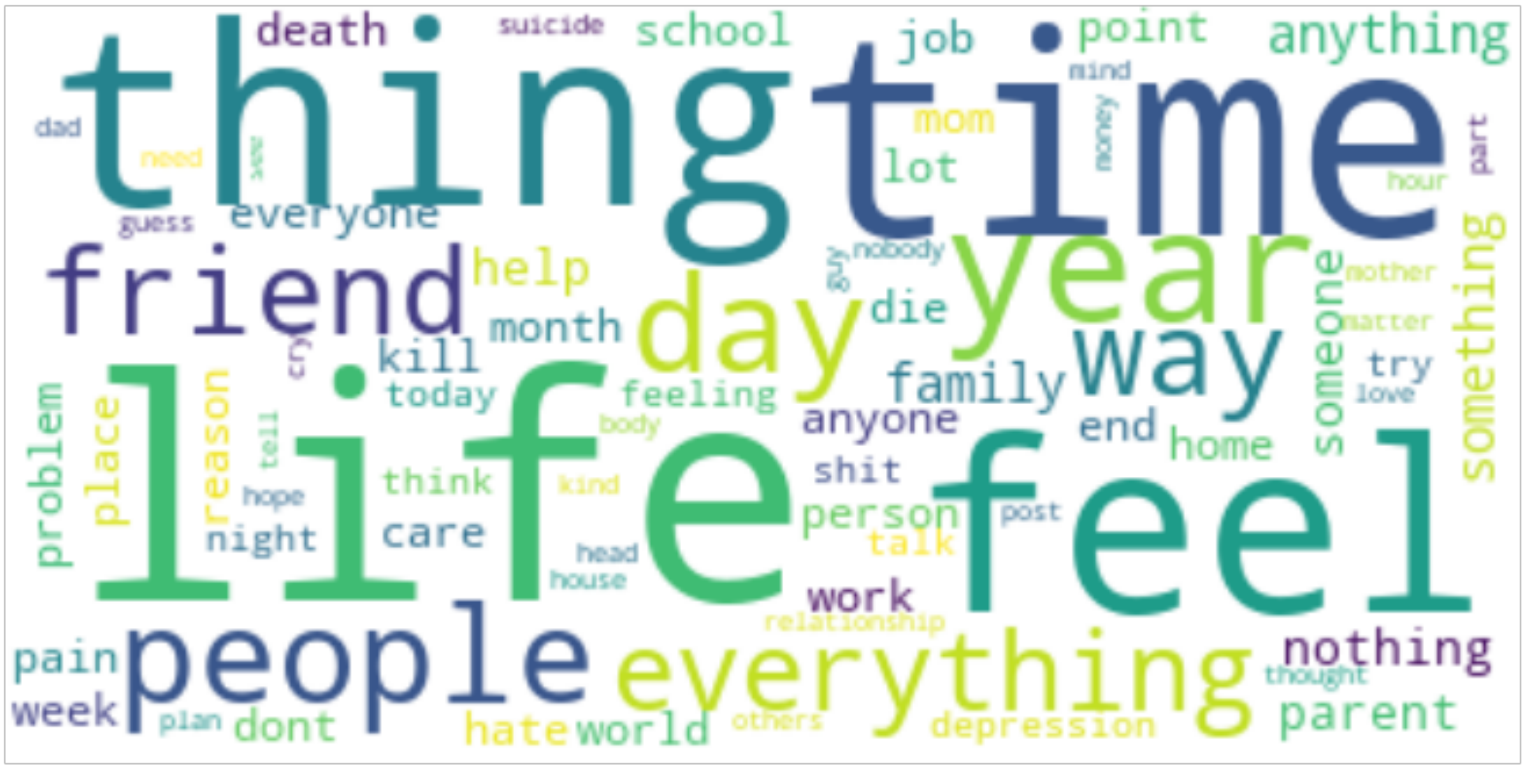
In [12]:

```
# Combining all the posts to one single list
SuicideWatch_NN_text =(" ").join(SuicideWatch_NN_list)
```

In [53]:

```
print('SuicideWatch WordCloud for Nouns')
# Generating a word cloud image for a sample of SuicideWatch:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Suicide
Watch_NN_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
SuicideWatch WordCloud for Nouns
```



In [ ]:

```
#The presence of the word "dont" explain the presence of positive words mentione
d in the negative sense
```

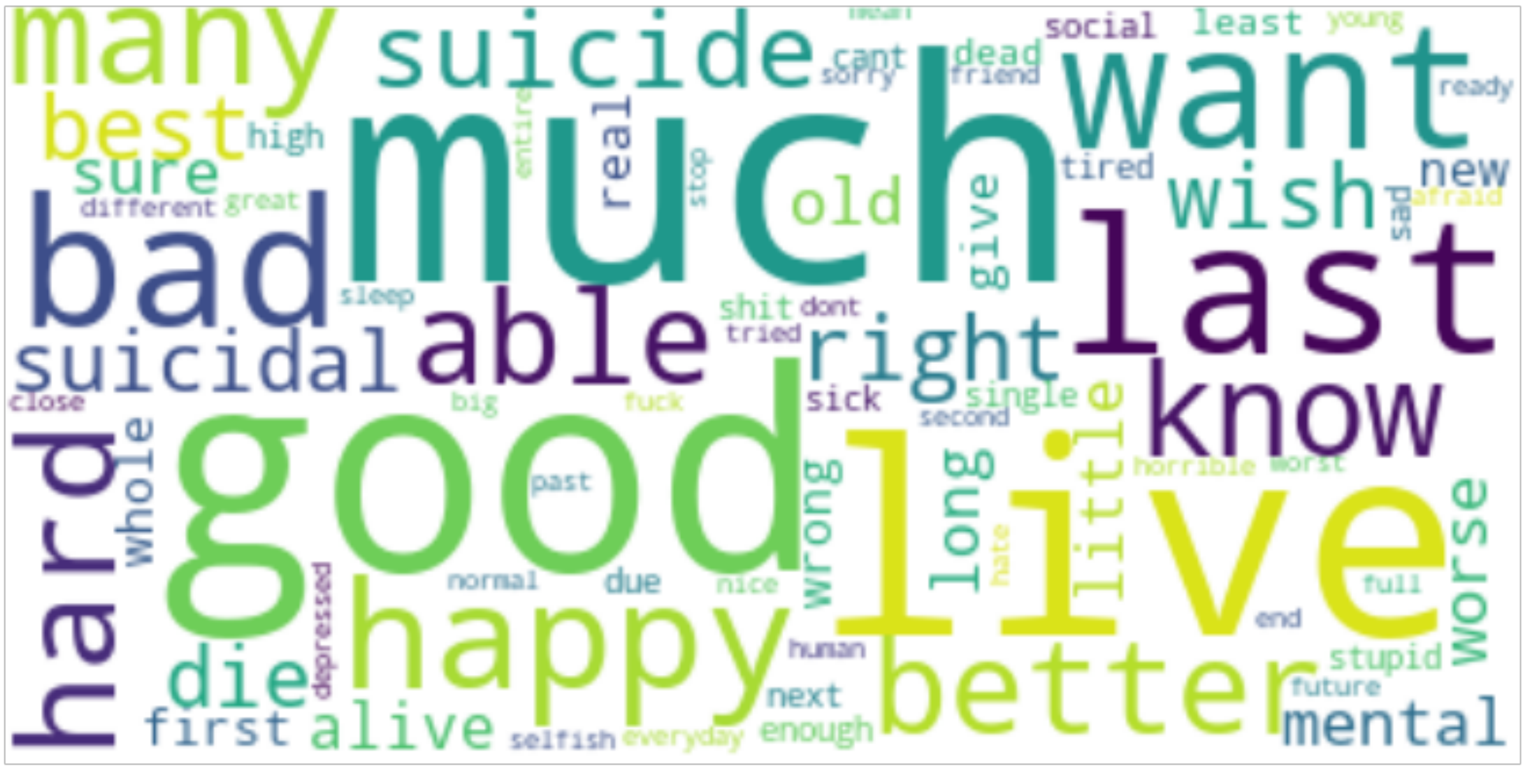
In [14]:

```
# Combining all the posts to one single list
SuicideWatch_AJ_text =(" ").join(SuicideWatch_AJ_list)
```

In [54]:

```python
print('SuicideWatch WordCloud for Adjectives')
# Generating a word cloud image for a sample of SuicideWatch:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Suicide
Watch_AJ_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
SuicideWatch WordCloud for Adjectives
```



In [ ]:

```python
#suicide #suicidal #worse #afraid #dead #sick #sad #depressed #bad #wrong #shit
#hard #hate #horrible
#The term good is used more than bad but most possibly in the negative way
```

In [ ]:

In [ ]:

In [ ]:

1. Depression

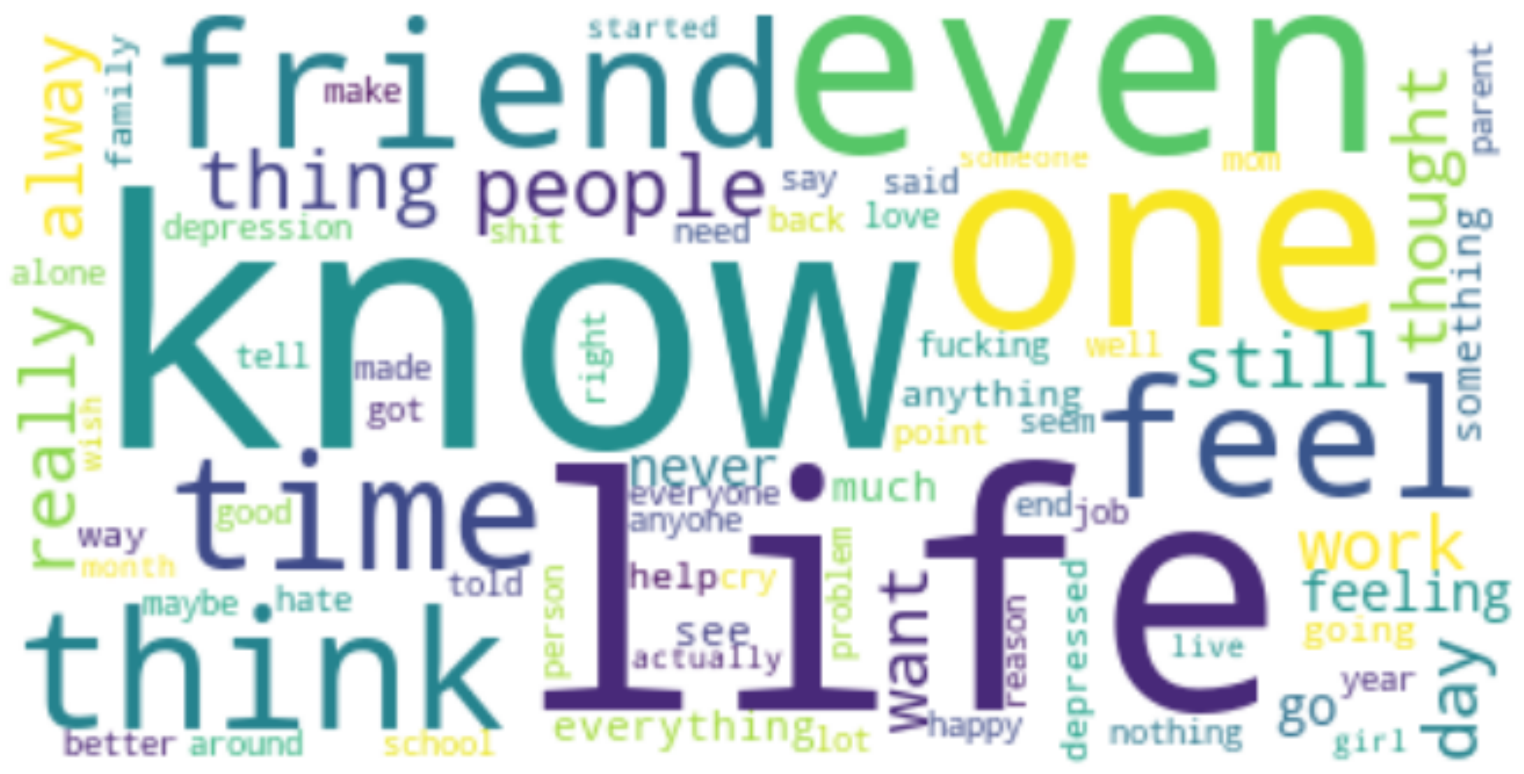
In [16]:

```
cachedStopWords = set(stopwords.words("english"))

####Keep Updating custom stop words
cachedStopWords.update(('nt', 'wo', 're', 'im', 'yall','u','ca','ive', 'wan','na
','gon','nov','x200b','amp',\
                        'wwwyoutubecomwatch','http','vbjkbl5olvm8','lt', 'br', '
gt', 'amp','tsp','tbsp','nbsp', 'le'))

def alpha_filter(w):
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False

## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now
sampleDepression_list = depressed_df.sample(frac=0.5, replace=True, random_state
=1)
Depression_list = sampleDepression_list['title_with_selftext_cleaned'].tolist()
Depression_list_lower = [tok.lower() for i in Depression_list for tok in nltk.wo
rd_tokenize(i)]
nltk_stopwords = set(stopwords.words('english'))
Depression_list_lower_stop = [x for x in Depression_list_lower if not x in nltk_
stopwords]
Depression_list_lower_stop_pun = [y for y in Depression_list_lower_stop if not a
lpha_filter(y)]
Depression_list_lower_stop_pun_extra = [''.join(x for x in par if x not in strin
g.punctuation) for par in\
                                          Depression_list_lower_stop_pun]

porter = WordNetLemmatizer()
Depression_list_lower_stop_pun_extra_lemmatized = []
for a in Depression_list_lower_stop_pun_extra :
    Depression_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize(a))
Depression_list = [x for x in Depression_list_lower_stop_pun_extra_lemmatized if
not x in cachedStopWords]

# Combining all the posts in the list to one single element to generate the Word
Cloud
Depression_allWords =(" ").join(Depression_list)
```

In [58]:

```
print('Depression All words WordCloud')
# Generating a word cloud image for a sample of Depression:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Depress
ion_allWords)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Depression All words WordCloud
```



In [ ]:

```
#friend #family #mom #school #parent #day #month #year #hate #depression
#We see the repetition of terms related to relationships such as friend, parent,
family etc. Are the relationships
#the reason why the person is depressed? Or he/she starts to think about them wh
en they are depressed?
#The term happy is mentioned more than sad but most possibly in the negative sen
se
#Again, they don't talk a lot about sadness because they seem to have passed tha
t phase
```

In [18]:

```
# Depression WordClouds for Nouns and Adjectives
```

In [19]:

```
Depression_pos = nltk.pos_tag(Depression_list)
# generate Noun list and adjective
Depression_NN_list = []
Depression_AJ_list = []
for i,j in Depression_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        Depression_NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        Depression_AJ_list.append(i)
print('There are',len(Depression_NN_list),'nouns in the list ')
print('There are',len(Depression_AJ_list),'adjectives in the list')
```

```
There are 396471 nouns in the list
There are 186827 adjectives in the list
```

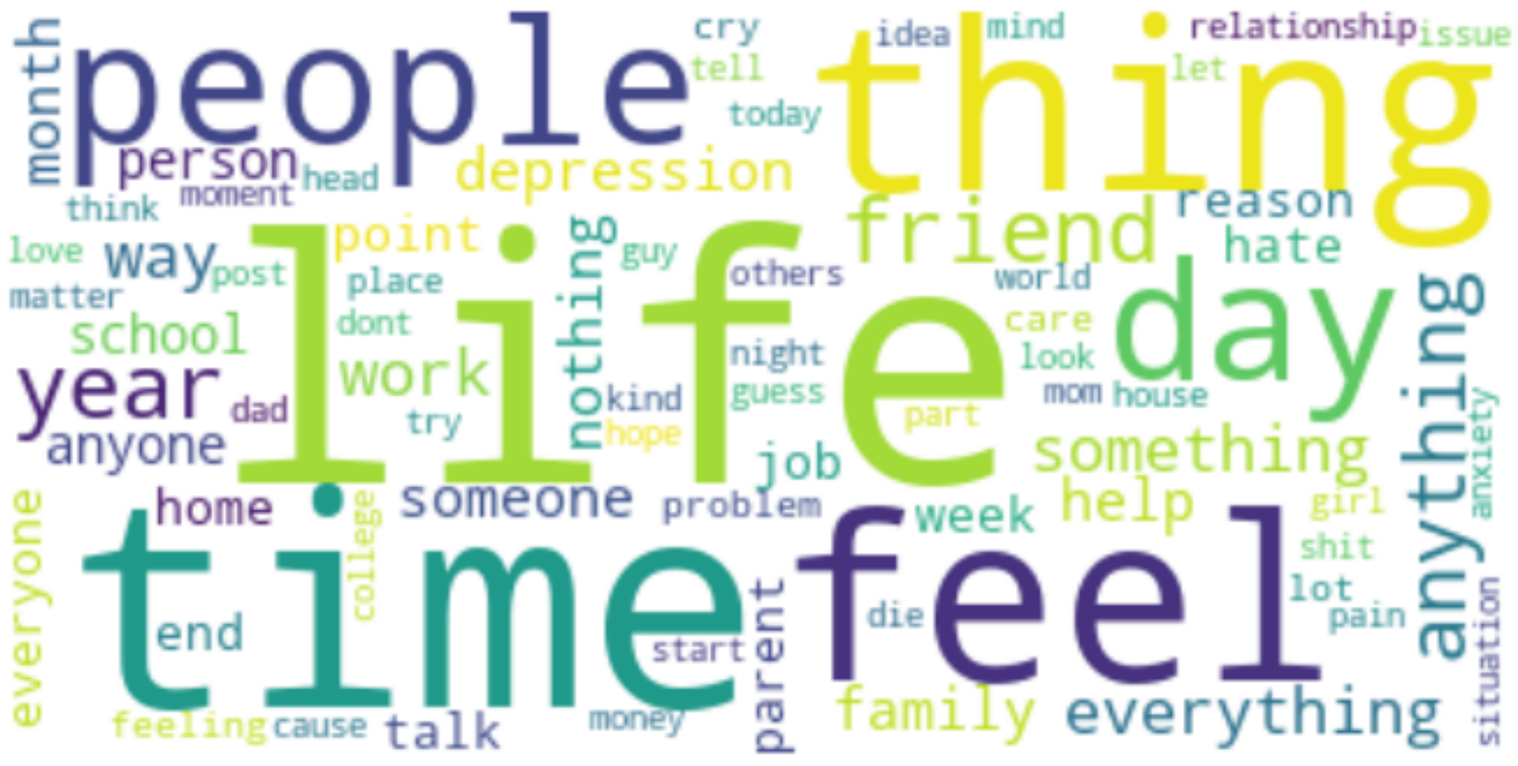
In [20]:

```
# Combining all the posts to one single list
Depression_NN_text =(" ").join(Depression_NN_list)
```

In [61]:

```
print('Depression WordCloud for Nouns')
# Generating a word cloud image for a sample of Depression:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Depress
ion_NN_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Depression WordCloud for Nouns
```



In [ ]:

```
#friend #family #dad #feeling #relationship #home #parent #mom
#cry #anxiety #hate #die #issue #idea
#college #school #job #work
#We say that words in this pool are more expressive compared to that in SuicideW
atch
```

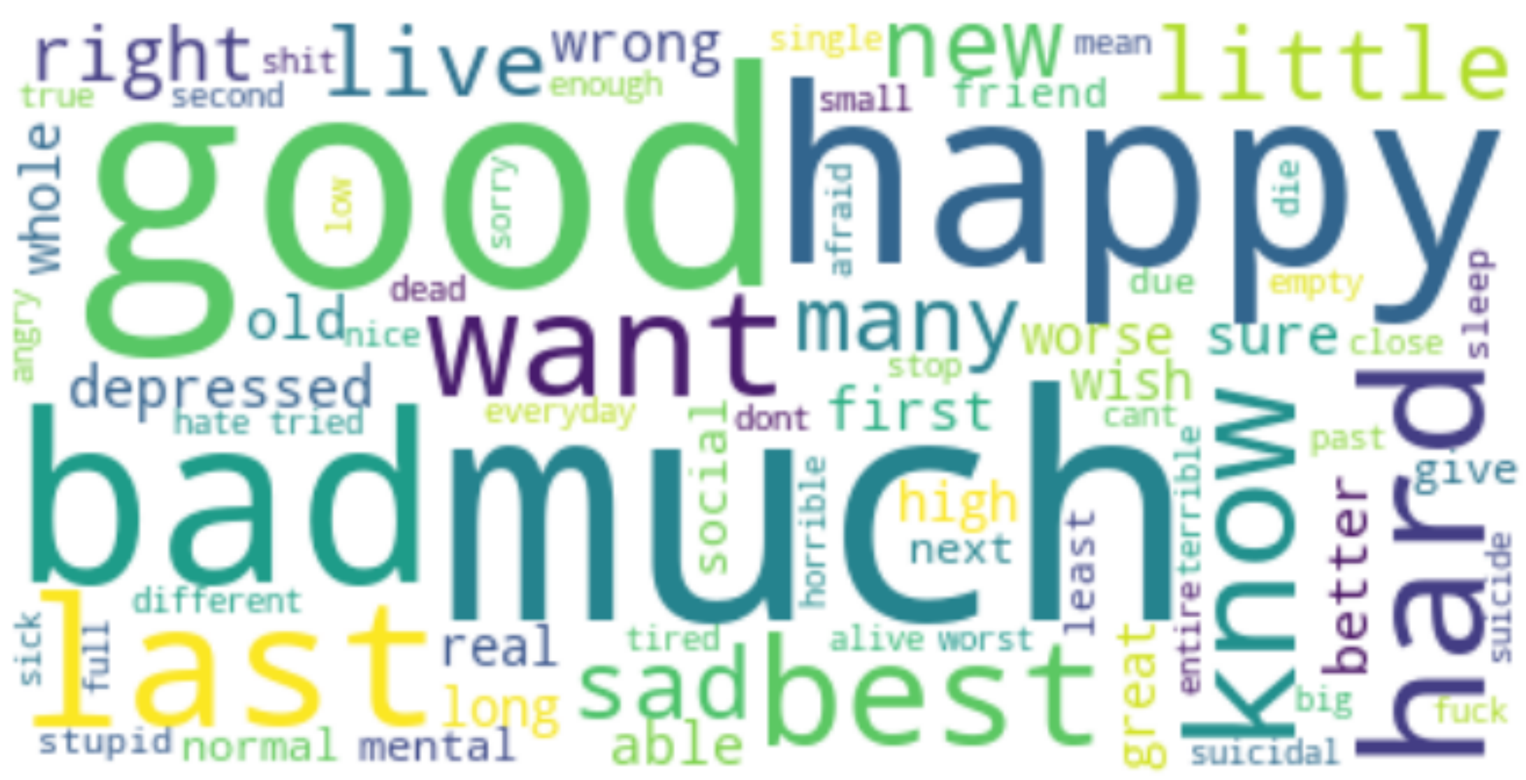
In [22]:

```
# Combining all the posts to one single list
Depression_AJ_text =(" ").join(Depression_AJ_list)
```

In [60]:

```
print('Depression WordCloud for Adjectives')
# Generating a word cloud image for a sample of SuicideWatch:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Depress
ion_AJ_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Depression WordCloud for Adjectives
```


right shit live wrong single new mean little
true second enough small friend
good happy
whole low sorry afraid die
dead old nice want many worse sure close sleep angry
depressed hate tried everyday stop wish cant dont first due empty
bad much social horrible high next least terrible entire know better hard give past suicide
different sick full last real tired alive worst great
long sad best big fuck
stupid normal mental able suicidal

In [ ]:

```
#Ironically, good and happy are the most frequently occuring words. People tend
to use them a lot in the negative sense
#We see suicide and suicidal in this pool too indicating that people are in the
stage of suicide ideation in this stage
```

In [ ]:

In [ ]:

In [ ]:

1. Self-Improvement

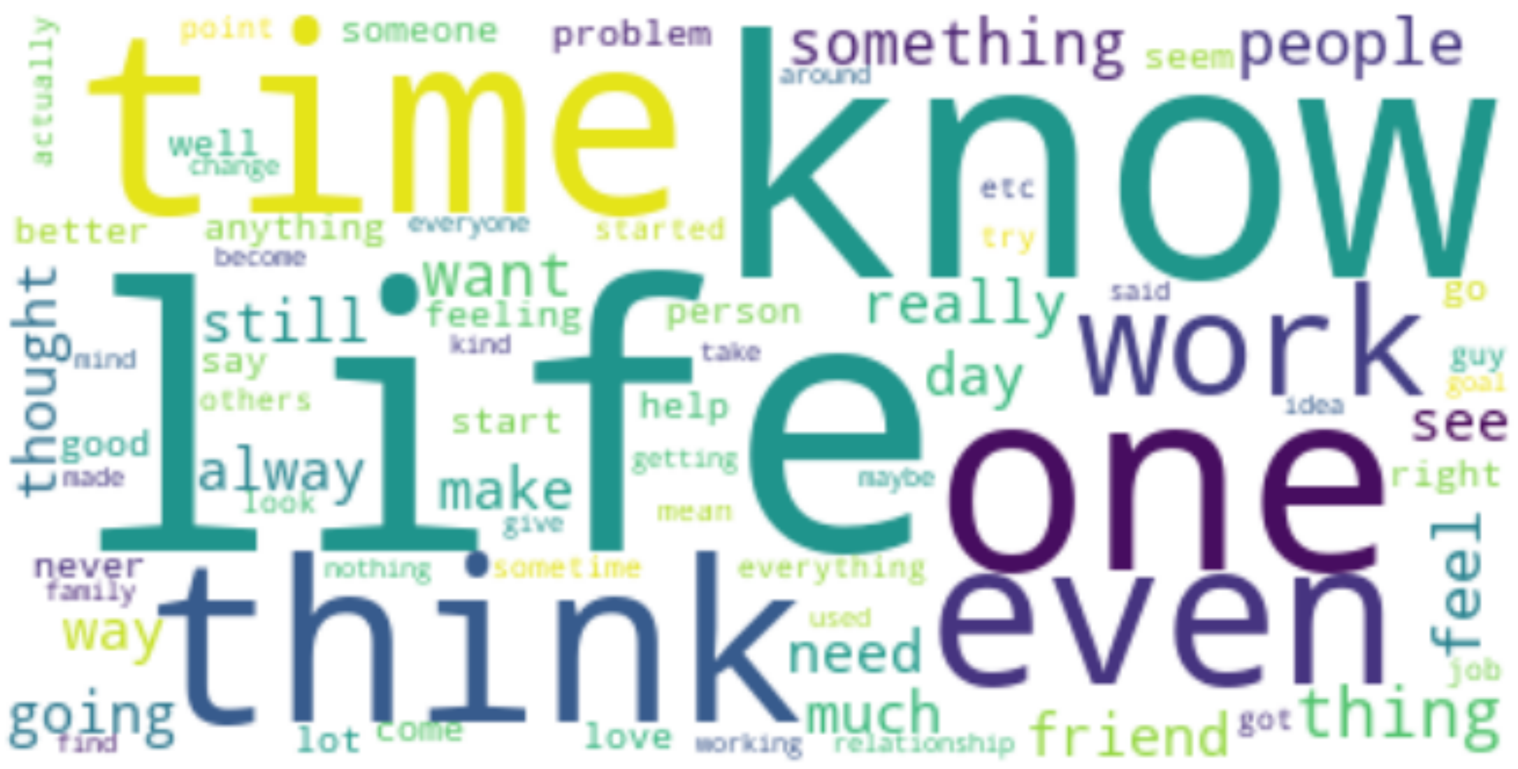
In [24]:

```
cachedStopWords = set(stopwords.words("english"))

####Keep Updating custom stop words
cachedStopWords.update(('nt', 'wo', 're', 'im', 'yall','u','ca','ive', 'wan','na
','gon','nov','x200b','amp',\
                        'wwwyoutubecomwatch','http','vbjkbl5olvm8','lt', 'br', '
gt', 'amp','tsp','tbsp','nbsp', 'le'))

def alpha_filter(w):
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False

## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now
sampleSelfImprovement_list = selfimprovement_df.sample(frac=0.5, replace=True, r
andom_state=1)
SelfImprovement_list = sampleSelfImprovement_list['title_with_selftext_cleaned']
.tolist()
SelfImprovement_list_lower = [tok.lower() for i in SelfImprovement_list for tok
in nltk.word_tokenize(i)]
nltk_stopwords = set(stopwords.words('english'))
SelfImprovement_list_lower_stop = [x for x in SelfImprovement_list_lower if not
x in nltk_stopwords]
SelfImprovement_list_lower_stop_pun = [y for y in SelfImprovement_list_lower_sto
p if not alpha_filter(y)]
SelfImprovement_list_lower_stop_pun_extra = [''.join(x for x in par if x not in
string.punctuation) for par in\
                                          SelfImprovement_list_lower_stop_pun]

porter = WordNetLemmatizer()
SelfImprovement_list_lower_stop_pun_extra_lemmatized = []
for a in SelfImprovement_list_lower_stop_pun_extra :
    SelfImprovement_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize
(a))
SelfImprovement_list = [x for x in SelfImprovement_list_lower_stop_pun_extra_lem
matized if not x in cachedStopWords]

# Combining all the posts in the list to one single element to generate the Word
Cloud
SelfImprovement_allWords =(" ").join(SelfImprovement_list)
```

In [62]:

```
print('SelfImprovement All words WordCloud')
# Generating a word cloud image for a sample of SuicideWatch:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(SelfImp
rovement_allWords)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
SelfImprovement All words WordCloud
```



In [ ]:

```

```

In [26]:

```
# Self-Improvement WordClouds for Nouns and Adjectives
```

In [27]:

```
SelfImprovement_pos = nltk.pos_tag(SelfImprovement_list)
# generate Noun list and adjective
SelfImprovement_NN_list = []
SelfImprovement_AJ_list = []
for i,j in SelfImprovement_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        SelfImprovement_NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        SelfImprovement_AJ_list.append(i)
print('There are',len(SelfImprovement_NN_list),'nouns in the list ')
print('There are',len(SelfImprovement_AJ_list),'adjectives in the list')
```

```
There are 851924 nouns in the list
There are 405758 adjectives in the list
```

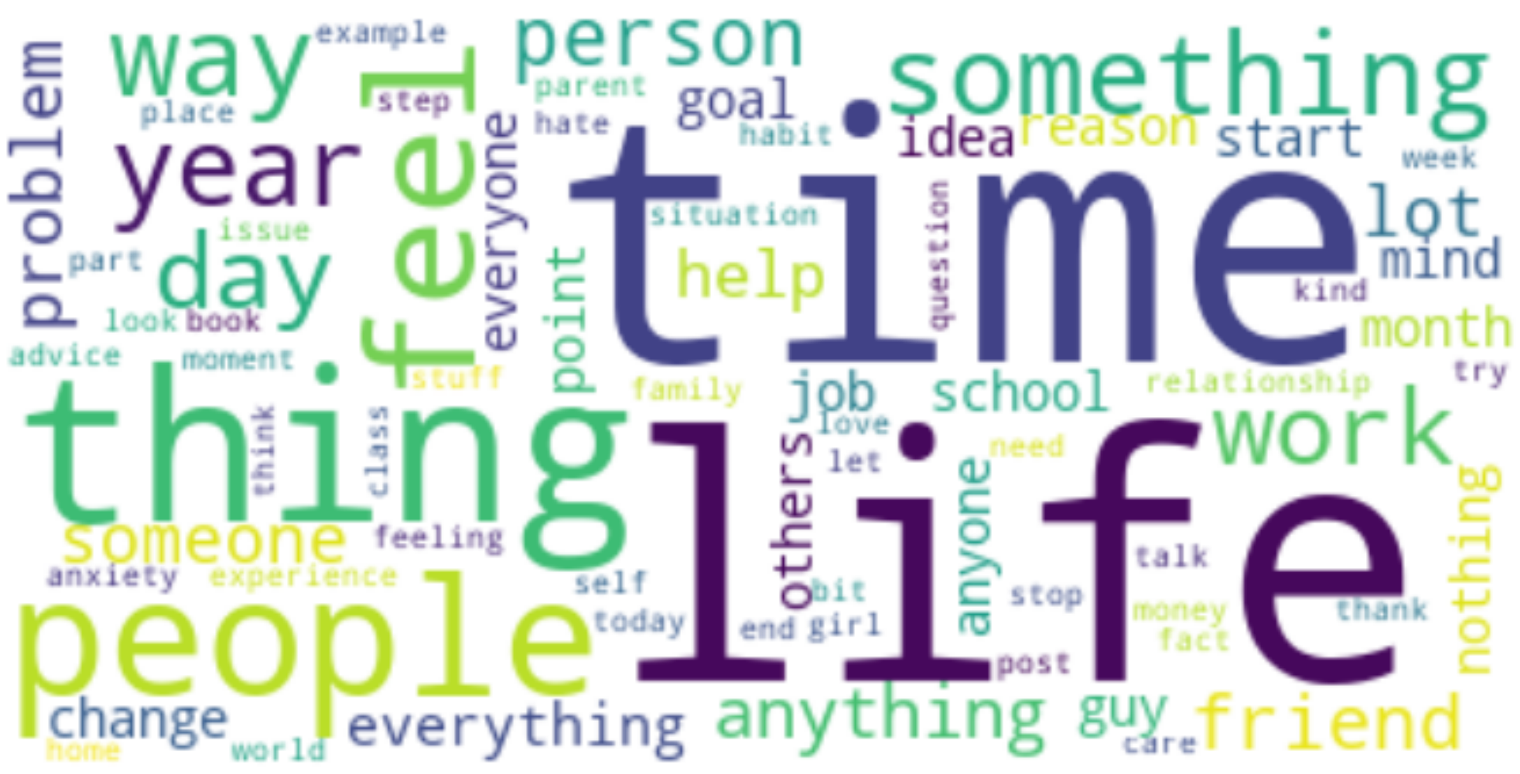
In [28]:

```
# Combining all the posts to one single list
SelfImprovement_NN_text =(" ").join(SelfImprovement_NN_list)
```

In [63]:

```
print('SelfImprovement WordCloud for Nouns')
# Generating a word cloud image for a sample of Self-Improvement:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(SelfImp
rovement_NN_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
SelfImprovement WordCloud for Nouns
```



In [ ]:

```
#Positive words - #care #advice #change #kind #goal #experience #example #others
#world #family #friend
#Again, like we saw in Depression, we see relations frequently repeating in this
```

In [ ]:

```

```

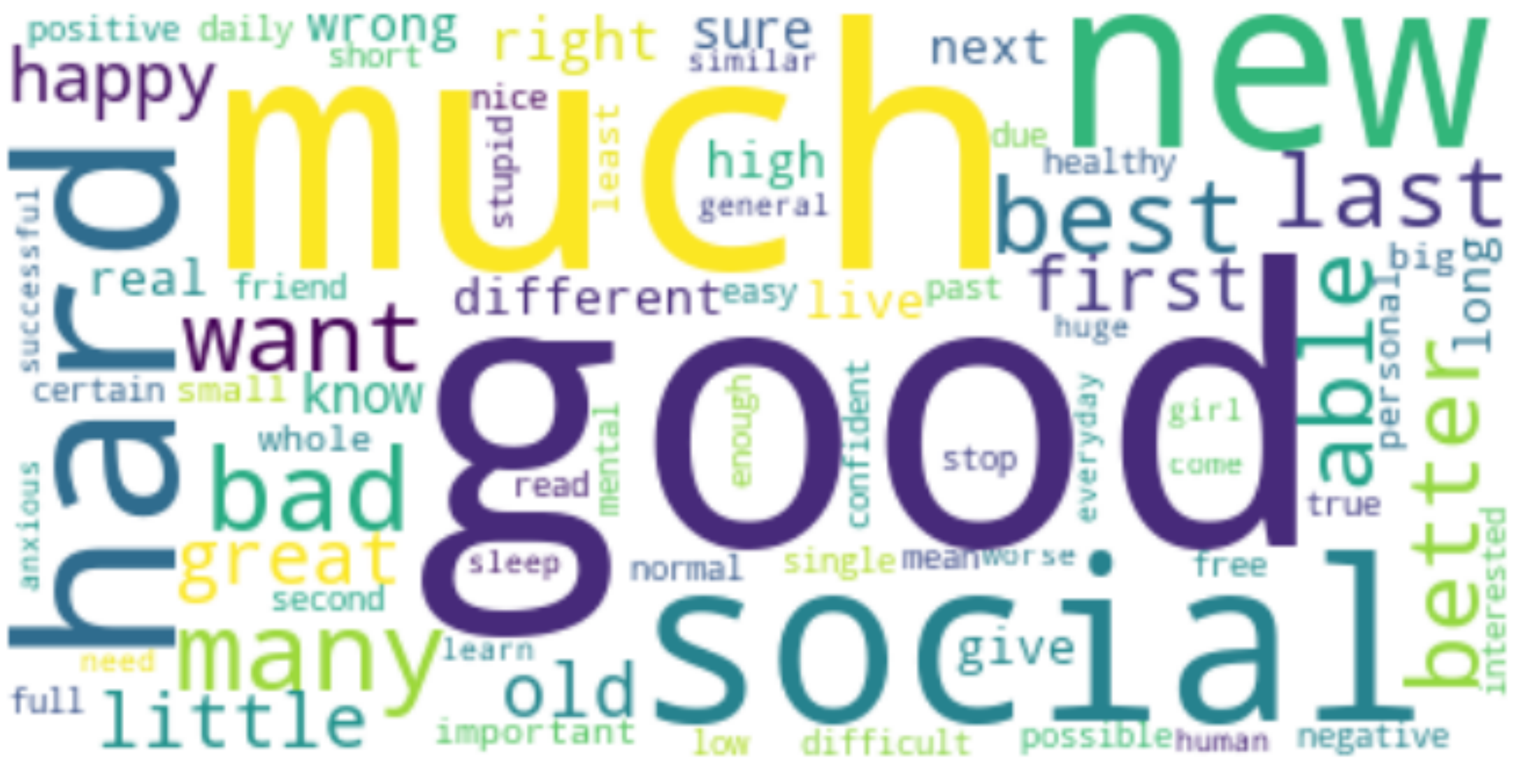
In [30]:

```
# Combining all the posts to one single list
SelfImprovement_AJ_text =(" ").join(SelfImprovement_AJ_list)
```

In [64]:

```
print('SelfImprovement WordCloud for Adjectives')
# Generating a word cloud image for a sample of SelfImprovement:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(SelfImp
rovement_AJ_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
SelfImprovement WordCloud for Adjectives
```



In [ ]:

```
#social #good #best #possible #important #successful #confident #positive #healt
hy #best #great #better #happy #great
#low #difficult #negative #little #anxious #stop #worse
#We see a range of positive and words used while being motivated
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

1. Happy

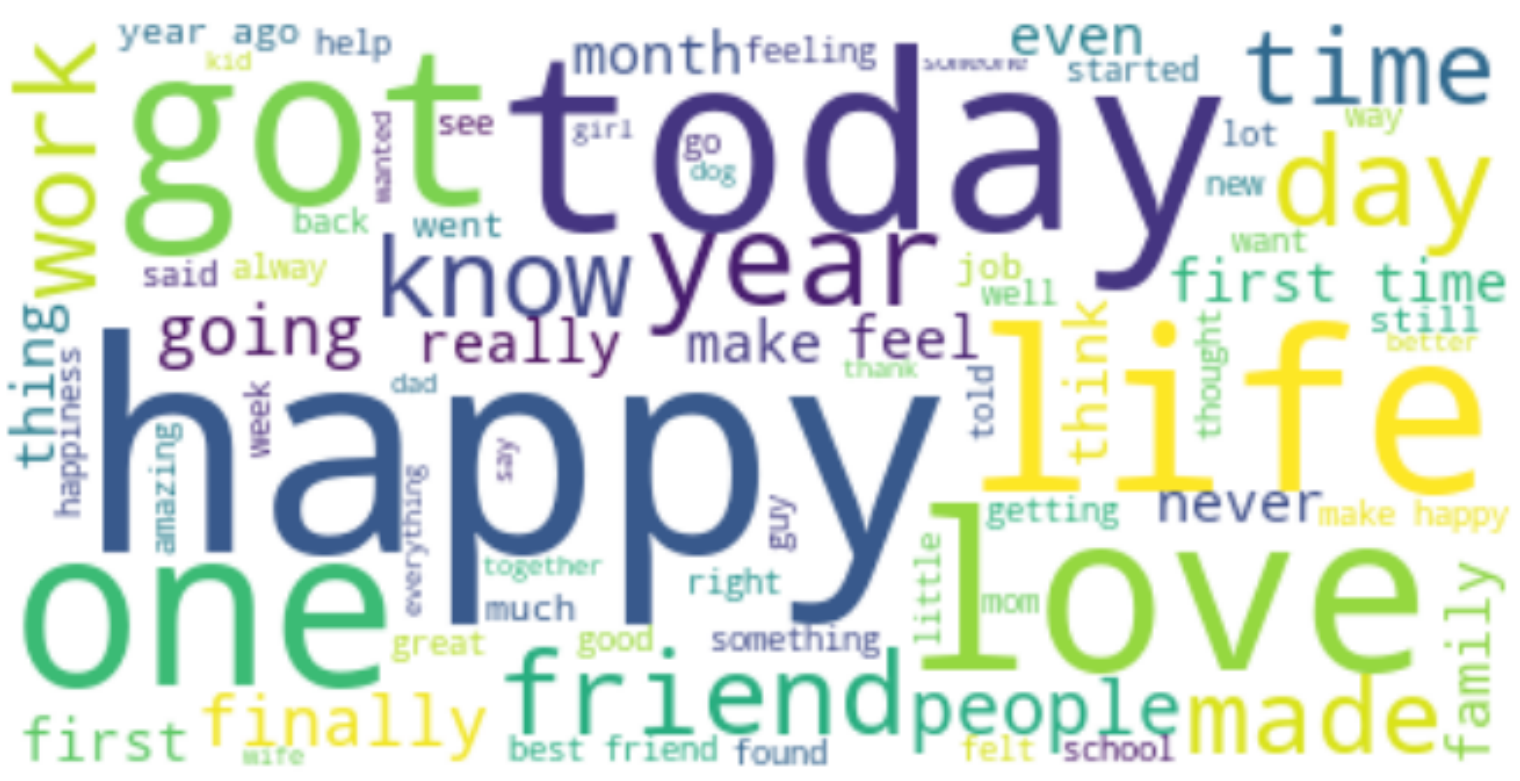
In [32]:

```
cachedStopWords = set(stopwords.words("english"))

####Keep Updating custom stop words
cachedStopWords.update(('nt', 'wo', 're', 'im', 'yall','u','ca','ive', 'wan','na
','gon','nov','x200b','amp',\
                        'wwwyoutubecomwatch','http','vbjkbl5olvm8','lt', 'br', '
gt', 'amp','tsp','tbsp','nbsp', 'le'))

def alpha_filter(w):
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False

## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now
sampleHappy_list = happy_df.sample(frac=0.5, replace=True, random_state=1)
Happy_list = sampleHappy_list['title_with_selftext_cleaned'].tolist()
Happy_list_lower = [tok.lower() for i in Happy_list for tok in nltk.word_tokeniz
e(i)]
nltk_stopwords = set(stopwords.words('english'))
Happy_list_lower_stop = [x for x in Happy_list_lower if not x in nltk_stopwords]
Happy_list_lower_stop_pun = [y for y in Happy_list_lower_stop if not alpha_filte
r(y)]
Happy_list_lower_stop_pun_extra = [''.join(x for x in par if x not in string.pun
ctuation) for par in\
                                          Happy_list_lower_stop_pun]

porter = WordNetLemmatizer()
Happy_list_lower_stop_pun_extra_lemmatized = []
for a in Happy_list_lower_stop_pun_extra :
    Happy_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize(a))
Happy_list = [x for x in Happy_list_lower_stop_pun_extra_lemmatized if not x in
cachedStopWords]

# Combining all the posts in the list to one single element to generate the Word
Cloud
Happy_allWords =(" ").join(Happy_list)
```

In [65]:

```
print('Happy All words WordCloud')
# Generating a word cloud image for a sample of Happy:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Happy_a
llWords)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Happy All words WordCloud
```



In [ ]:

```
#dad #mom #friend #wife #dog #family #together
#great #good #amazing #happiness #thank #love
#year #week #month
#Things are going great over a period of time
```

In [34]:

```
# Happy WordClouds for Nouns and Adjectives
```

In [35]:

```
Happy_pos = nltk.pos_tag(Happy_list)
# generate Noun list and adjective
Happy_NN_list = []
Happy_AJ_list = []
for i,j in Happy_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        Happy_NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        Happy_AJ_list.append(i)
print('There are',len(Happy_NN_list),'nouns in the list ')
print('There are',len(Happy_AJ_list),'adjectives in the list')
```

```
There are 178383 nouns in the list
There are 86253 adjectives in the list
```

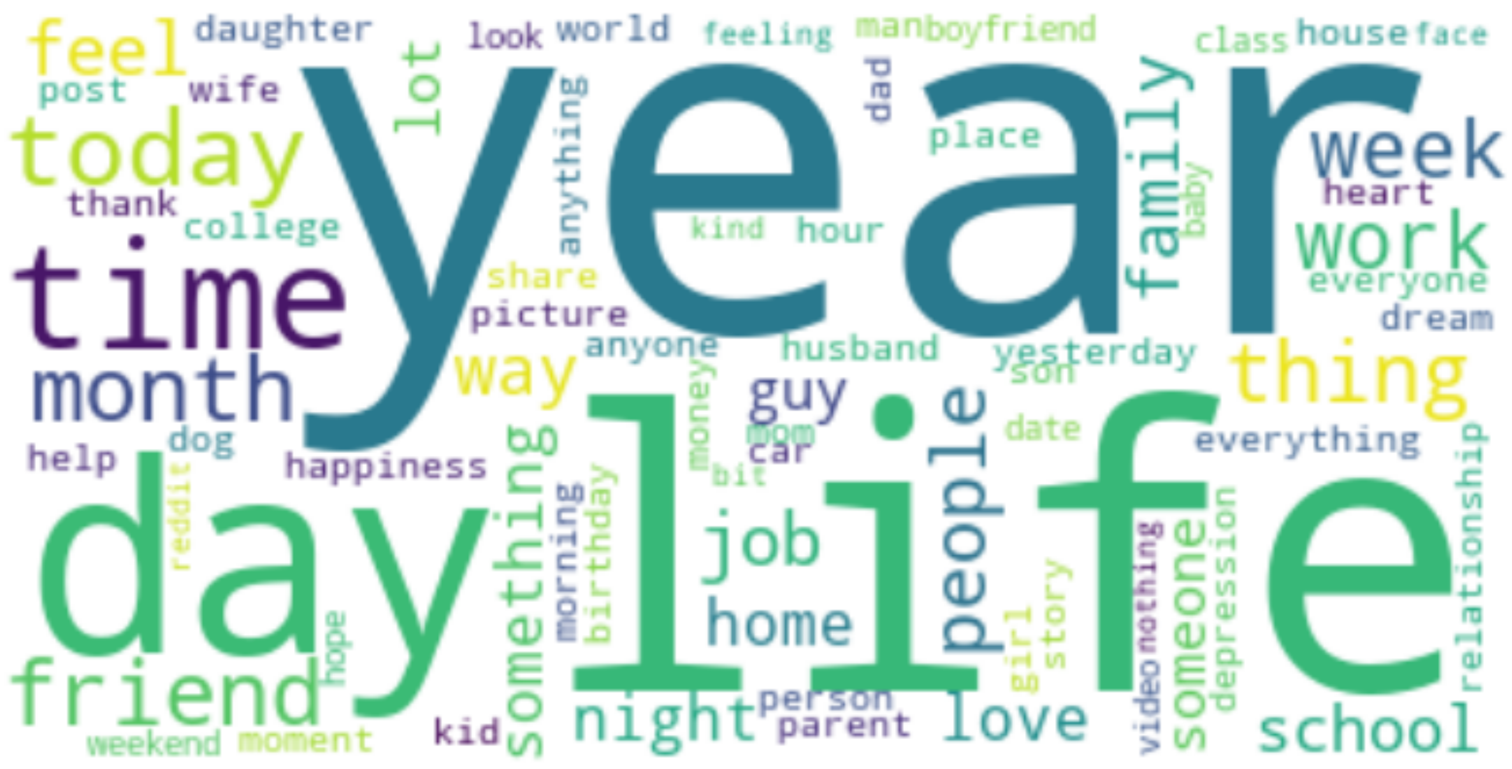
In [36]:

```
# Combining all the posts to one single list
Happy_NN_text = (" ").join(Happy_NN_list)
```

In [66]:

```
print('Happy WordCloud for Nouns')
# Generating a word cloud image for a sample of Happy:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Happy_N
N_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Happy WordCloud for Nouns
```



In [ ]:

```

```

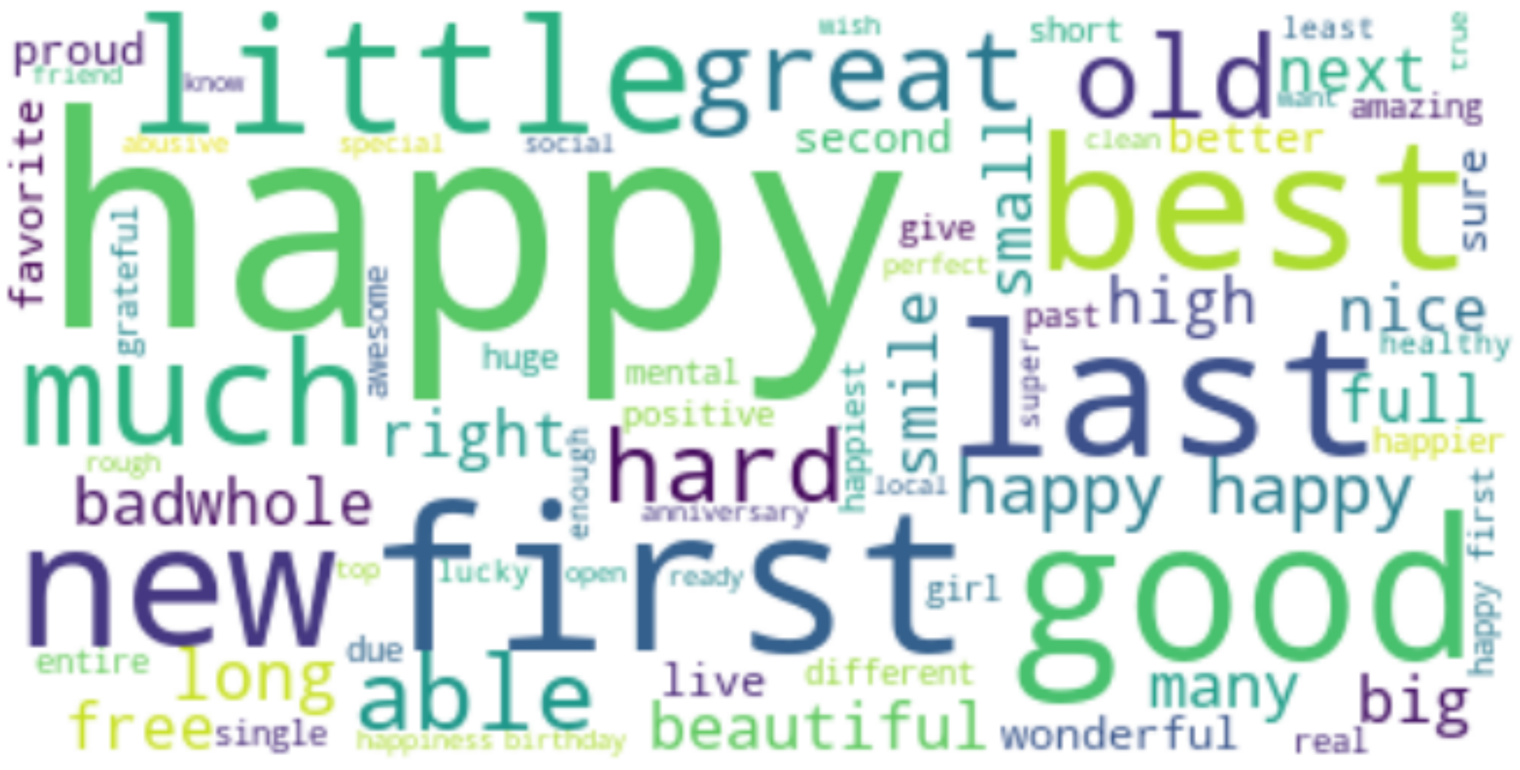
In [38]:

```
# Combining all the posts to one single list
Happy_AJ_text =(" ").join(Happy_AJ_list)
```

In [67]:

```python
print('Happy WordCloud for Adjectives')
# Generating a word cloud image for a sample of SuicideWatch:
wordcloud = WordCloud(background_color="white", max_words = 75).generate(Happy_A
J_text)
# Display the generated image:
# the matplotlib way:
plt.rcParams["figure.figsize"] = (12,6)
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis("off")
plt.show()
```

```
Happy WordCloud for Adjectives
```



In [ ]:

```python
#proud #happiest #wonderful #positive #special #lucky #wish #favorite #healthy #
super #awesome #smile #nice #high #perfect
#rough #abusive #bad
#birthday #anniversary
```