In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.model_selection import train_test_split # function for splitting data to train and te
st sets
```

In [2]:

```
import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier
```

In [4]:

```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:

```
from subprocess import check_output
```

In [6]:

```
data = pd.read_csv('/home/yeshua/Documents/study/excel/Sentiment.csv')
# Keeping only the neccessary columns
data = data[['text','sentiment']]
```

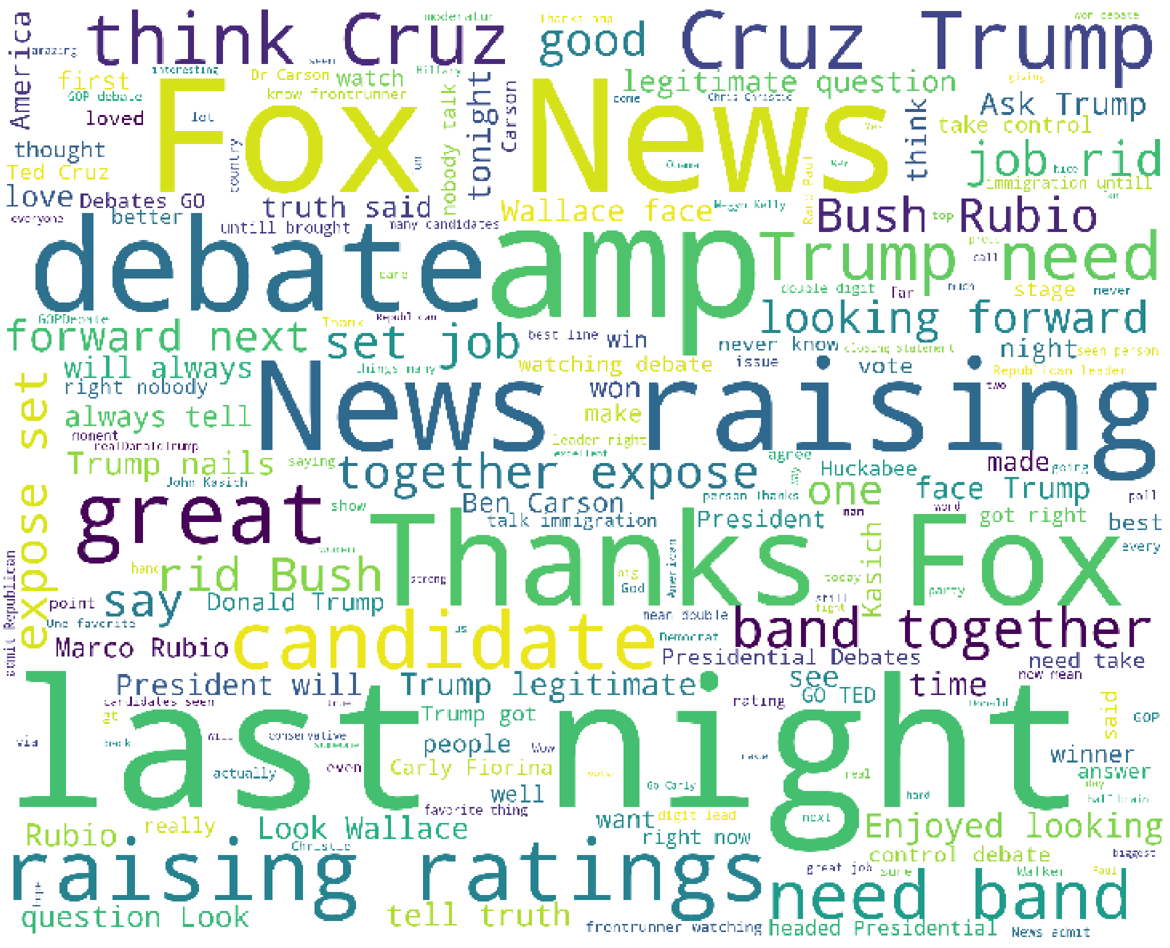
In [7]:

```
# Splitting the dataset into train and test set
train, test = train_test_split(data,test_size = 0.1)
# Removing neutral sentiments
train = train[train.sentiment != "Neutral"]
```

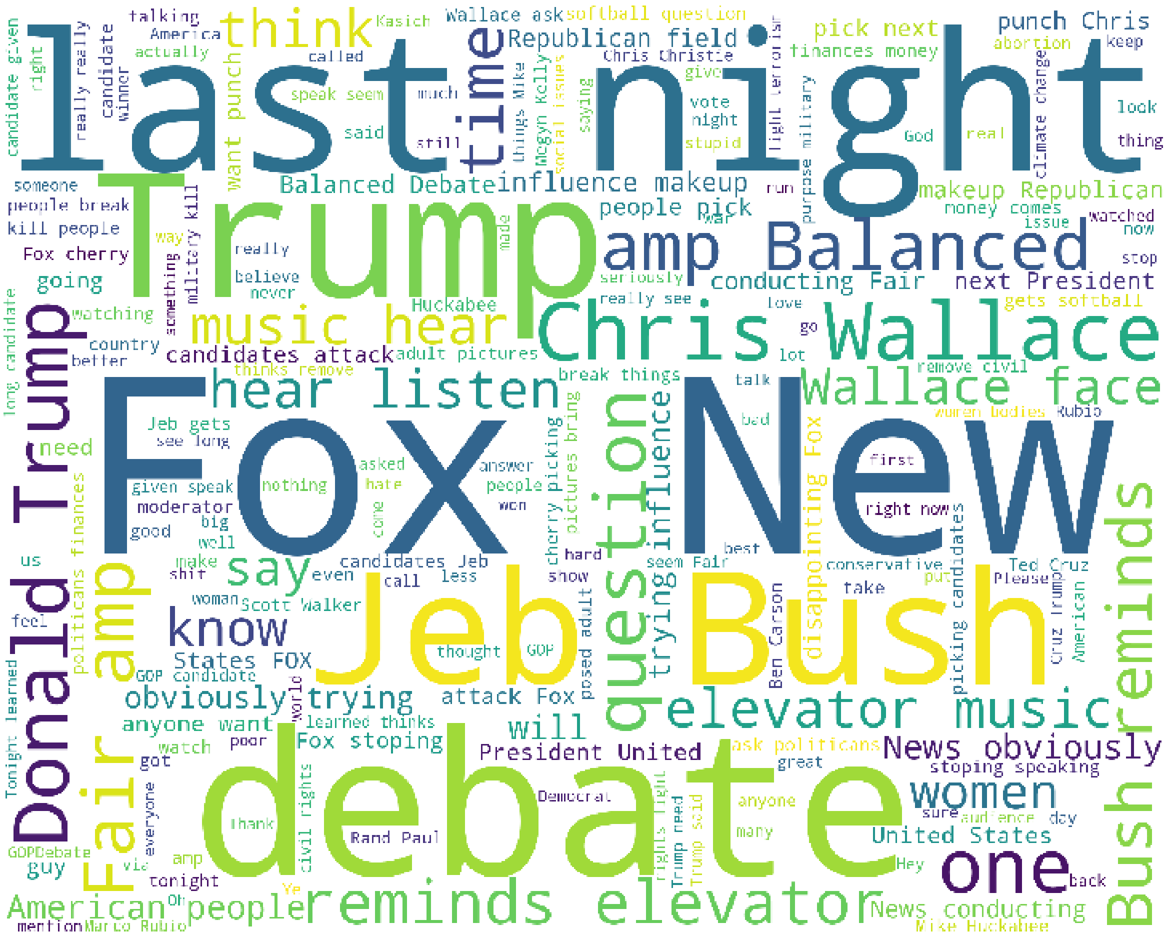
In [13]:

```
train_pos = train[ train['sentiment'] == 'Positive']
train_pos = train_pos['text']
train_neg = train[ train['sentiment'] == 'Negative']
train_neg = train_neg['text']

def wordcloud_draw(data, color = 'white'):
    words = ' '.join(data)
    cleaned_word = " ".join([word for word in words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and not word.startswith('#')
                                and word != 'RT'
                            ])
    wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color=color,
                      width=2500,
                      height=2000
                     ).generate(cleaned_word)
    plt.figure(1,figsize=(13, 13))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()

print("Positive words")
wordcloud_draw(train_pos,'white')
print("Negative words")
wordcloud_draw(train_neg)
```

```
Positive words
```



Negative words



In [15]:

```
import nltk
nltk.download('stopwords')

tweets = []
stopwords_set = set(stopwords.words("english"))

for index, row in train.iterrows():
    words_filtered = [e.lower() for e in row.text.split() if len(e) >= 3]
    words_cleaned = [word for word in words_filtered
        if 'http' not in word
        and not word.startswith('@')
        and not word.startswith('#')
        and word != 'RT']
    words_without_stopwords = [word for word in words_cleaned if not word in stopwords_set]
    tweets.append((words_without_stopwords, row.sentiment))

test_pos = test[ test['sentiment'] == 'Positive']
test_pos = test_pos['text']
test_neg = test[ test['sentiment'] == 'Negative']
test_neg = test_neg['text']
```

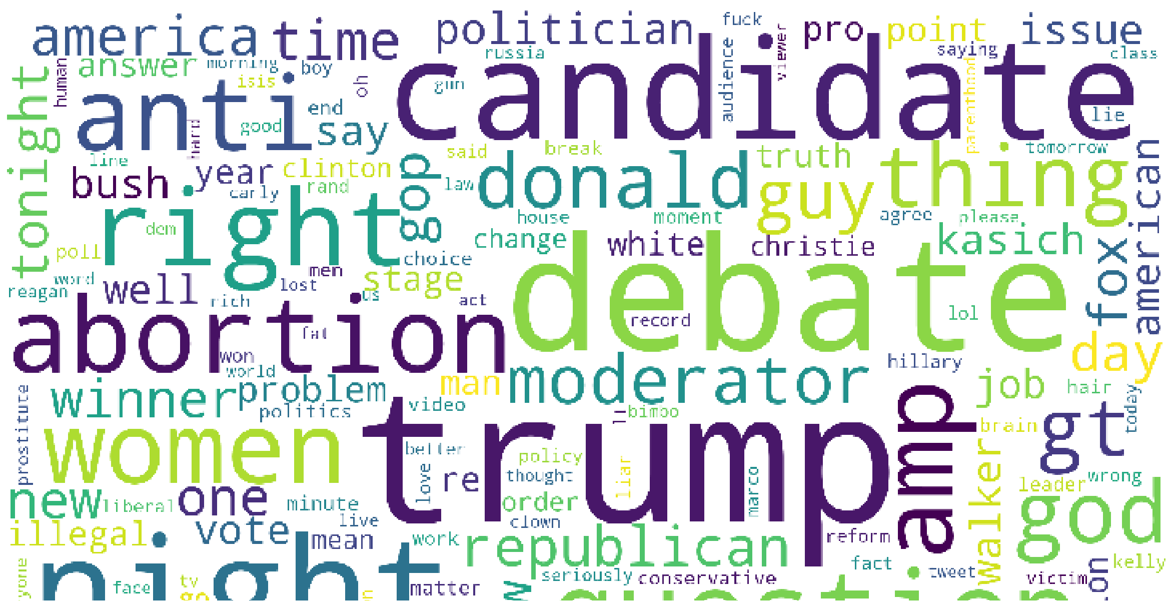
```
[nltk_data] Downloading package stopwords to /home/yeshua/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

In [16]:

```
def get_words_in_tweets(tweets):
    all = []
    for (words, sentiment) in tweets:
        all.extend(words)
    return all

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    features = wordlist.keys()
    return features
w_features = get_word_features(get_words_in_tweets(tweets))

def extract_features(document):
    document_words = set(document)
    features = {}
    for word in w_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
```

In [17]:

```
wordcloud_draw(w_features)
```



gopdebate
question
carson
rubio
cruz
jeb
game
president
party
war
obama
huckabee
state
immigration
show
people
paul
democrat
idiot
statement
country
loser
megynkelly
voter
woman
care
realdonaldtrump
speak
wallace
something
dog
foxnews
money
megyn
life
win
now
shit
idea
air
way
cnn
uni
gay
black
run
fan
hug
want
wait
facebook
friend
think
joke
pig
florina
performance
much
know
going
race
made
great
pimp
bad
non
animal
attack
stupid
name
ever
soo

```
In [18]:
```

```python
# Training the Naive Bayes classifier
training_set = nltk.classify.apply_features(extract_features,tweets)
classifier = nltk.NaiveBayesClassifier.train(training_set)
```

```
In [19]:
```

```python
neg_cnt = 0
pos_cnt = 0
for obj in test_neg:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Negative'):
        neg_cnt = neg_cnt + 1
for obj in test_pos:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Positive'):
        pos_cnt = pos_cnt + 1

print('[Negative]: %s/%s '  % (len(test_neg),neg_cnt))
print('[Positive]: %s/%s '  % (len(test_pos),pos_cnt))
```

```
[Negative]: 878/842
[Positive]: 207/75
```