STAT 656

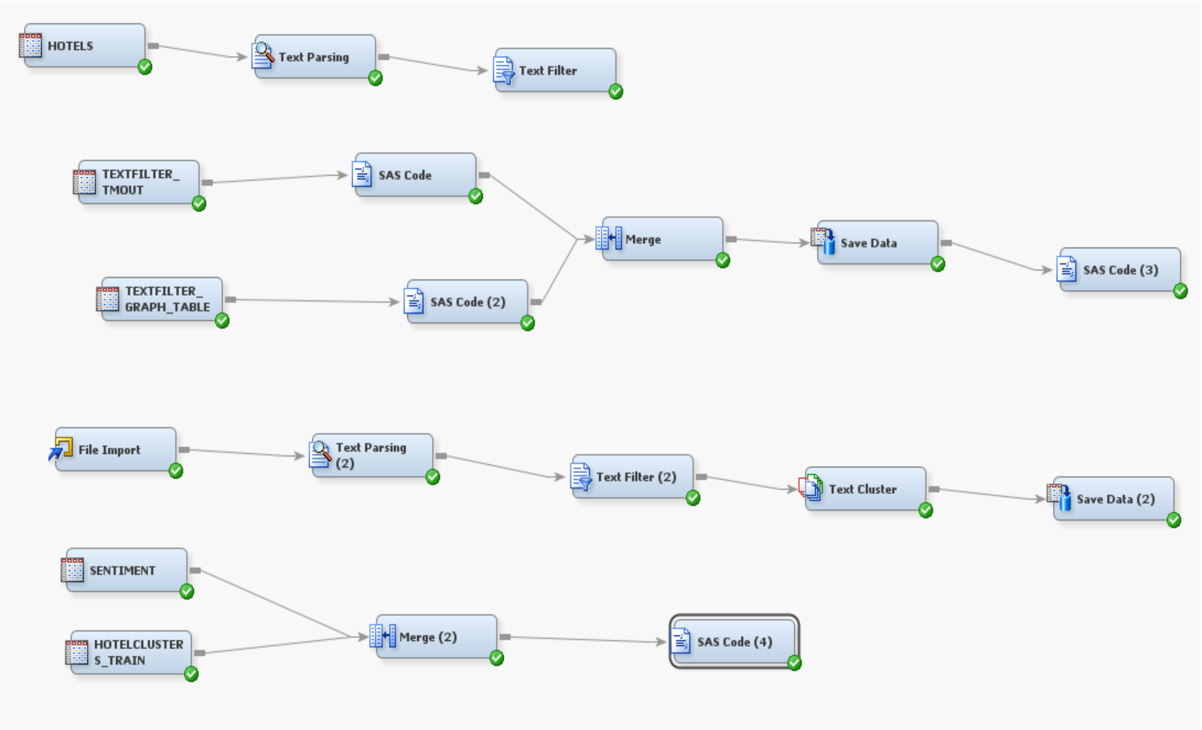
Week 13 Assignment

Name-Mayank Jaggi

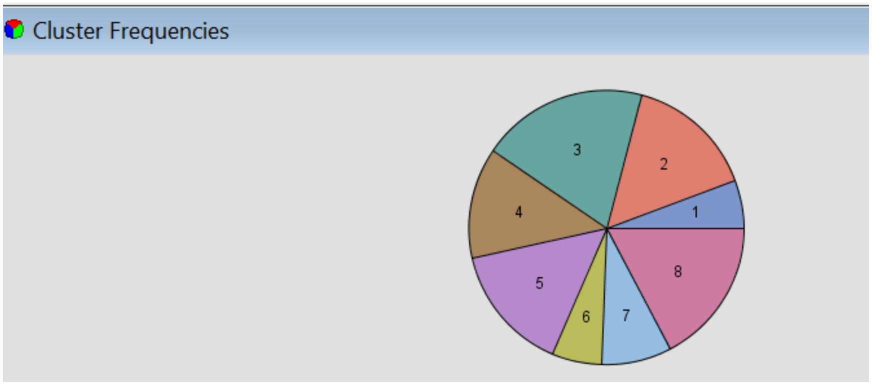
UIN-526005299

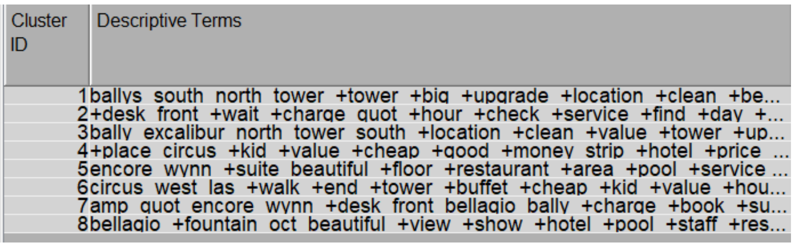
PART 1 SAS EM

1) Project Diagram



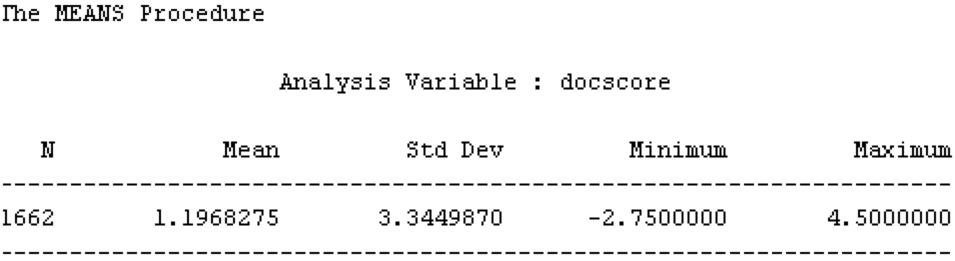
2) Cluster Frequency and Descriptive Terms





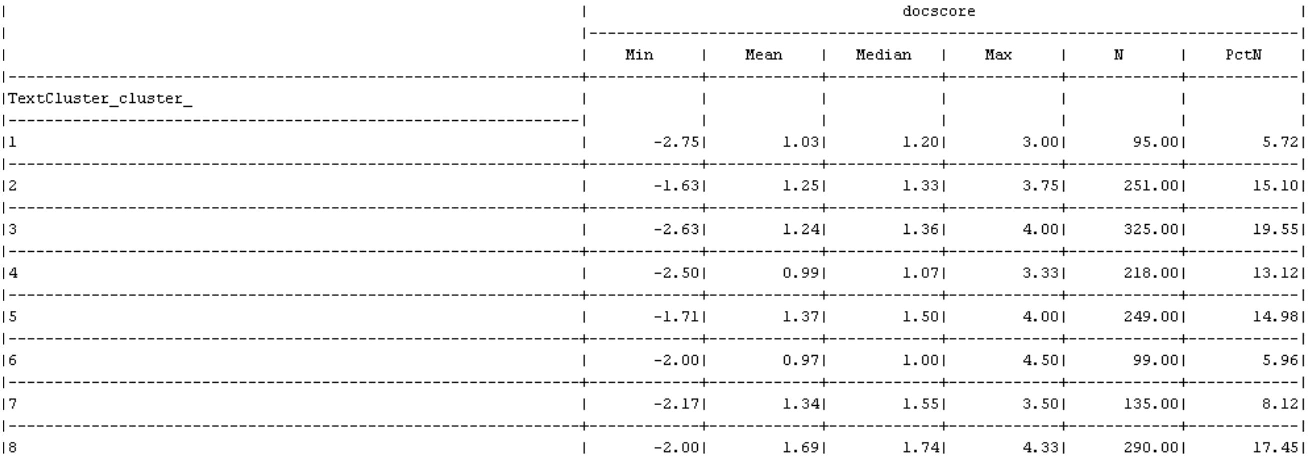
Cluster ID	Descriptive Terms
1	ballys south north tower +tower +big +upgrade +location +clean +be...
2	+desk front +wait +charge quot +hour +check +service +find +day +...
3	bally excalibur north tower south +location +clean +value +tower +up...
4	+place circus +kid +value +cheap +good +money strip +hotel +price ...
5	encore wynn +suite beautiful +floor +restaurant +area +pool +service ...
6	circus west las +walk +end +tower +buffet +cheap +kid +value +hou...
7	amp quot encore wynn +desk front bellagio bally +charge +book +su...
8	bellagio +fountain oct beautiful +view +show +hotel +pool +staff +res...

3) Average sentiment for the entire corpus



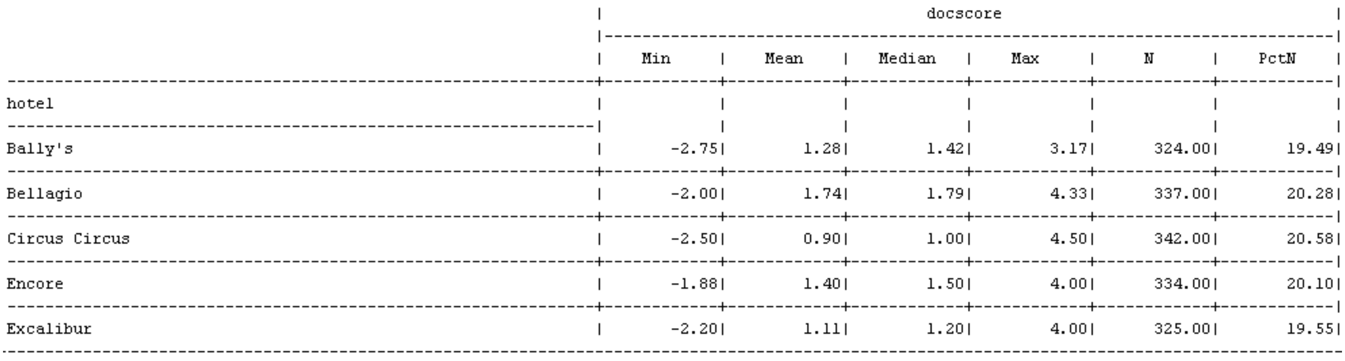
```
The MEANS Procedure

                    Analysis Variable : docscore

  N            Mean          Std Dev          Minimum          Maximum
----------------------------------------------------------------------
1662       1.1968275       3.3449870       -2.7500000        4.5000000
----------------------------------------------------------------------
```

4) Average sentiment for each text cluster



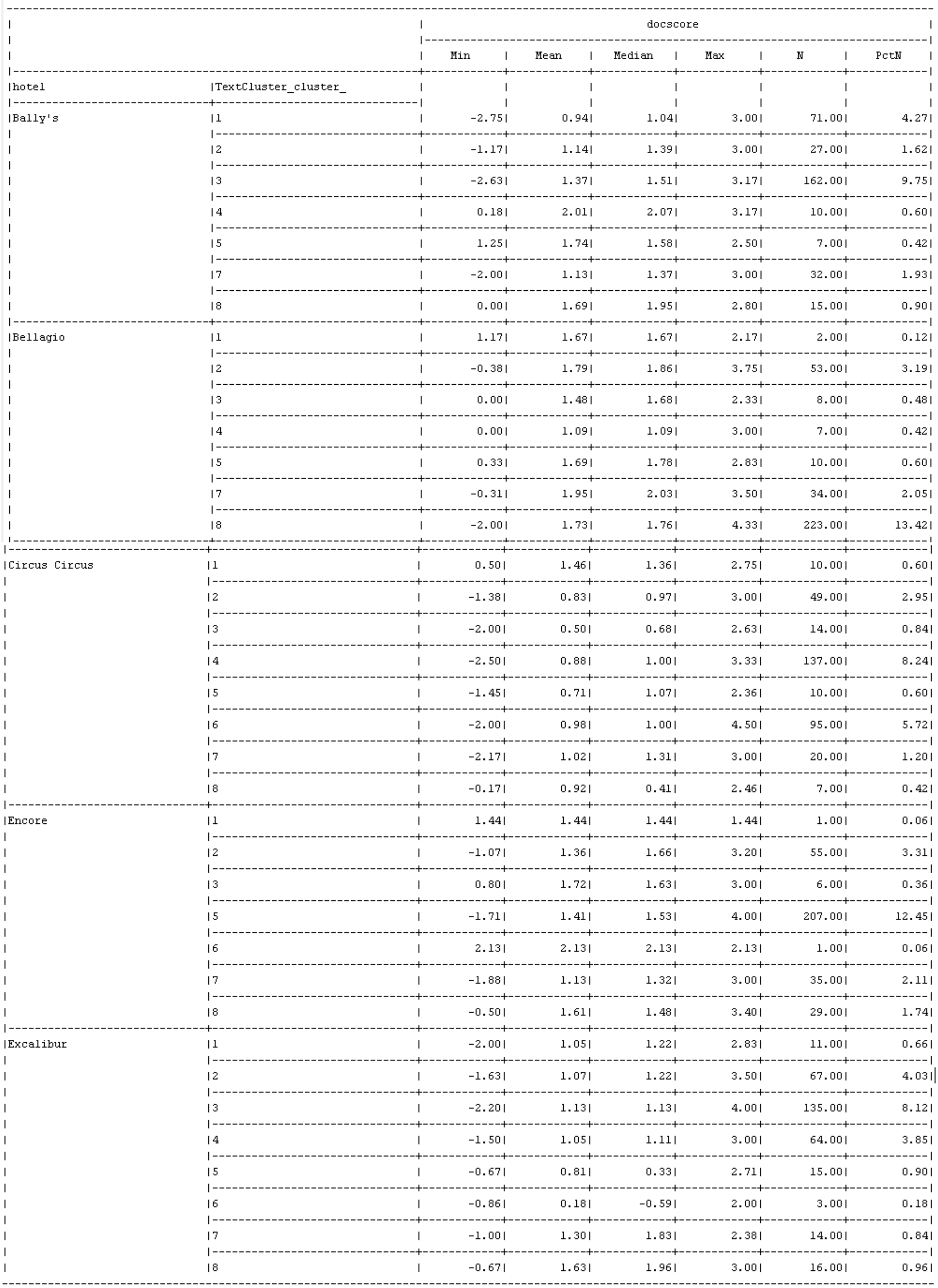
```
|                                                          |                            docscore                                   |
|                                                          |-----------------------------------------------------------------------|
|                                                          |   Min    |   Mean   |  Median  |   Max    |    N     |   PctN   |
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|TextCluster_cluster_                                      |          |          |          |          |          |          |
|----------------------------------------------------------|          |          |          |          |          |          |
|1                                                         |     -2.75|      1.03|      1.20|      3.00|     95.00|      5.72|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|2                                                         |     -1.63|      1.25|      1.33|      3.75|    251.00|     15.10|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|3                                                         |     -2.63|      1.24|      1.36|      4.00|    325.00|     19.55|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|4                                                         |     -2.50|      0.99|      1.07|      3.33|    218.00|     13.12|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|5                                                         |     -1.71|      1.37|      1.50|      4.00|    249.00|     14.98|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|6                                                         |     -2.00|      0.97|      1.00|      4.50|     99.00|      5.96|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|7                                                         |     -2.17|      1.34|      1.55|      3.50|    135.00|      8.12|
|----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
|8                                                         |     -2.00|      1.69|      1.74|      4.33|    290.00|     17.45|
```

5) Average sentiment for each hotel



```
                                                           |                            docscore                                   |
                                                           |-----------------------------------------------------------------------|
                                                           |   Min    |   Mean   |  Median  |   Max    |    N     |   PctN   |
-----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
hotel                                                      |          |          |          |          |          |          |
-----------------------------------------------------------|          |          |          |          |          |          |
Bally's                                                    |     -2.75|      1.28|      1.42|      3.17|    324.00|     19.49|
-----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
Bellagio                                                   |     -2.00|      1.74|      1.79|      4.33|    337.00|     20.28|
-----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
Circus Circus                                              |     -2.50|      0.90|      1.00|      4.50|    342.00|     20.58|
-----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
Encore                                                     |     -1.88|      1.40|      1.50|      4.00|    334.00|     20.10|
-----------------------------------------------------------+----------+----------+----------+----------+----------+----------|
Excalibur                                                  |     -2.20|      1.11|      1.20|      4.00|    325.00|     19.55|
------------------------------------------------------------------------------------------------------------------------------------
```

6) Average sentiment for each hotel cluster combination



		docscore					
		Min	Mean	Median	Max	N	PctN
hotel	TextCluster_cluster_						
Bally's	1	-2.75	0.94	1.04	3.00	71.00	4.27
	2	-1.17	1.14	1.39	3.00	27.00	1.62
	3	-2.63	1.37	1.51	3.17	162.00	9.75
	4	0.18	2.01	2.07	3.17	10.00	0.60
	5	1.25	1.74	1.58	2.50	7.00	0.42
	7	-2.00	1.13	1.37	3.00	32.00	1.93
	8	0.00	1.69	1.95	2.80	15.00	0.90
Bellagio	1	1.17	1.67	1.67	2.17	2.00	0.12
	2	-0.38	1.79	1.86	3.75	53.00	3.19
	3	0.00	1.48	1.68	2.33	8.00	0.48
	4	0.00	1.09	1.09	3.00	7.00	0.42
	5	0.33	1.69	1.78	2.83	10.00	0.60
	7	-0.31	1.95	2.03	3.50	34.00	2.05
	8	-2.00	1.73	1.76	4.33	223.00	13.42
Circus Circus	1	0.50	1.46	1.36	2.75	10.00	0.60
	2	-1.38	0.83	0.97	3.00	49.00	2.95
	3	-2.00	0.50	0.68	2.63	14.00	0.84
	4	-2.50	0.88	1.00	3.33	137.00	8.24
	5	-1.45	0.71	1.07	2.36	10.00	0.60
	6	-2.00	0.98	1.00	4.50	95.00	5.72
	7	-2.17	1.02	1.31	3.00	20.00	1.20
	8	-0.17	0.92	0.41	2.46	7.00	0.42
Encore	1	1.44	1.44	1.44	1.44	1.00	0.06
	2	-1.07	1.36	1.66	3.20	55.00	3.31
	3	0.80	1.72	1.63	3.00	6.00	0.36
	5	-1.71	1.41	1.53	4.00	207.00	12.45
	6	2.13	2.13	2.13	2.13	1.00	0.06
	7	-1.88	1.13	1.32	3.00	35.00	2.11
	8	-0.50	1.61	1.48	3.40	29.00	1.74
Excalibur	1	-2.00	1.05	1.22	2.83	11.00	0.66
	2	-1.63	1.07	1.22	3.50	67.00	4.03
	3	-2.20	1.13	1.13	4.00	135.00	8.12
	4	-1.50	1.05	1.11	3.00	64.00	3.85
	5	-0.67	0.81	0.33	2.71	15.00	0.90
	6	-0.86	0.18	-0.59	2.00	3.00	0.18
	7	-1.00	1.30	1.83	2.38	14.00	0.84
	8	-0.67	1.63	1.96	3.00	16.00	0.96

########################PART 2 PYTHON##################

1) PYTHON CODE

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 24 10:15:57 2019

@author: mayank
"""

import pandas as pd
import numpy as np
import string
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import random
from nltk import pos_tag
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet as wn
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.decomposition import LatentDirichletAllocation

# Increase column width to let pandas read large text columns
pd.set_option('max_colwidth', 32000)
# Read N=13,575 California Cabernet Savignon Reviews
df = pd.read_excel("hotels.xlsx")
sw = pd.read_excel("D:\\Work\\Course Work\\Semester 4\\STAT 656\\Lectures & 
Assignment\\Week 13\\Sentiment Files\\afinn_sentiment_words.xlsx")

#Do cluster analysis to identify 7 word clusters using TFIDF and SVD
def my_analyzer(s):
    # Synonym List
    syns = {"n't":'not', 'wont':'would not', 'cant':'can not', \
            'cannot':'can not', 'couldnt':'could not', \
            'shouldnt':'should not', 'wouldnt':'would not'}

    # Preprocess String s
    s = s.lower()
    s = s.replace(',', '. ')
    # Tokenize
    tokens = word_tokenize(s)
    tokens = [word.replace(',','') for word in tokens ]
    tokens = [word for word in tokens if ('*' not in word) and \
              ("''" != word) and ("``" != word) and \
              (word!='description') and (word !='dtype') \
              and (word != 'object') and (word!="'s")]

    # Map synonyms
    for i in range(len(tokens)):
        if tokens[i] in syns:
            tokens[i] = syns[tokens[i]]

    # Remove stop words
    punctuation = list(string.punctuation)+['..', '...']
    pronouns = ['i', 'he', 'she', 'it', 'him', 'they', 'we', 'us', 'them']
    other = ['go','get','one','room','hotel','day','casino','quot','strip',\
             'amp','get','also','night','would','take','place','time']
    stop = stopwords.words('english') + punctuation + pronouns + other
    filtered_terms = [word for word in tokens if (word not in stop) and \
                  (len(word)>1) and (not word.replace('.','',1).isnumeric()) \
                  and (not word.replace("'",'',2).isnumeric())]

    # Lemmatization & Stemming - Stemming with WordNet POS
    # Since lemmatization requires POS need to set POS
    tagged_words = pos_tag(filtered_terms, lang='eng')
    # Stemming with for terms without WordNet POS
    stemmer = SnowballStemmer("english")
    wn_tags = {'N':wn.NOUN, 'J':wn.ADJ, 'V':wn.VERB, 'R':wn.ADV}
    wnl = WordNetLemmatizer()
    stemmed_tokens = []
    for tagged_token in tagged_words:
        term = tagged_token[0]
        pos  = tagged_token[1]
        pos  = pos[0]
        try:
            pos   = wn_tags[pos]
            stemmed_tokens.append(wnl.lemmatize(term, pos=pos))
        except:
            stemmed_tokens.append(stemmer.stem(term))
    return stemmed_tokens

# Further Customization of Stopping and Stemming using NLTK
def my_preprocessor(s):
    # Preprocess String s
    s = s.lower()
    # Replace special characters with spaces
    s = s.replace('-', ' ')
    s = s.replace('_', ' ')
    s = s.replace(',', '. ')
    # Replace not contraction with not
    s = s.replace("'nt", " not")
    s = s.replace("n't", " not")
    return s

def tokenizer(s):
    # Tokenize
    print("Tokenizer")
    tokens = word_tokenize(s)
    tokens = [word.replace(',','') for word in tokens ]
    tokens = [word for word in tokens if word.find('*')!=True and \
              word != "''" and word !="``" and word!='description' \
              and word !='dtype']
    return tokens

def display_topics(lda, terms, n_terms=15):
    for topic_idx, topic in enumerate(lda):
        if topic_idx > 8:
            break
        message  = "Topic #%d: " %(topic_idx+1)
        print(message)
        abs_topic = abs(topic)
        topic_terms_sorted = \
                [[terms[i], topic[i]] \
                     for i in abs_topic.argsort()[:-n_terms - 1:-1]]
        k = 5
        n = int(n_terms/k)
        m = n_terms - k*n
        for j in range(n):
            l = k*j
            message = ''
            for i in range(k):
                if topic_terms_sorted[i+l][1]>0:
                    word = "+"+topic_terms_sorted[i+l][0]
                else:
                    word = "-"+topic_terms_sorted[i+l][0]
                message += '{:<15s}'.format(word)
            print(message)
        if m> 0:
            l = k*n
            message = ''
            for i in range(m):
                if topic_terms_sorted[i+l][1]>0:
                    word = "+"+topic_terms_sorted[i+l][0]
                else:
                    word = "-"+topic_terms_sorted[i+l][0]
                message += '{:<15s}'.format(word)
            print(message)
        print("")
    return

# Setup program constants
n_comments  = len(df['Review'])     # Number of hotel reviews
m_features = None                    # Number of SVD Vectors
s_words    = 'english'               # Stop Word Dictionary
comments = df['Review']             # place all text reviews in reviews
```

```
n_topics =  7                       # number of topic clusters to extract
max_iter = 100                        # maximum number of itertions
max_df   = 0.7                      # learning offset for LDAmax proportion of
docs/reviews allowed for a term
learning_offset = 10.               # learning offset for LDA
learning_method = 'online'           # learning method for LDA
tfidf = True                    # Set to True for TF-IDF Weighting

# Create Word Frequency by Review Matrix using Custom Analyzer
cv = CountVectorizer(max_df=max_df, min_df=4, max_features=m_features,\
                     analyzer=my_analyzer, ngram_range=(1,2))
tf    = cv.fit_transform(comments)
terms = cv.get_feature_names()
term_sums = tf.sum(axis=0)
term_counts = []
for i in range(len(terms)):
    term_counts.append([terms[i], term_sums[0,i]])
def sortSecond(e):
    return e[1]
term_counts.sort(key=sortSecond, reverse=True)
print("\nTerms with Highest Frequency:")
for i in range(50):
        print('{:<15s}{:>5d}'.format(term_counts[i][0], term_counts[i][1]))
print("")
# Modify tf, term frequencies, to TF/IDF matrix from the data
print("Conducting Term/Frequency Matrix using TF-IDF")
tfidf_vect = TfidfTransformer(norm=None, use_idf=True) #set norm=None
tf         = tfidf_vect.fit_transform(tf)

term_idf_sums = tf.sum(axis=0)
term_idf_scores = []
for i in range(len(terms)):
    term_idf_scores.append([terms[i], term_idf_sums[0,i]])
term_idf_scores.sort(key=sortSecond, reverse=True)

# In sklearn, SVD is synonymous with LSA (Latent Semantic Analysis)
lda = LatentDirichletAllocation(n_components=n_topics, max_iter=max_iter,\
learning_method=learning_method, \
learning_offset=learning_offset, \
random_state=12345)
lda.fit_transform(tf)

# Display the topic selections
lda_norm = lda.components_ / lda.components_.sum(axis=1)[:, np.newaxis]
# ** SCORE REVIEWS **
rev_scores = [[0]*(n_topics+1)] * n_comments
# Last topic count is number of reviews without any topic words
topic_counts = [0] * (n_topics+1)
for r in range(n_comments):
    idx = n_topics
    max_score = 0
    # Calculate Review Score
    j0 = tf[r].nonzero()
    nwords = len(j0[1])
    rev_score = [0]*(n_topics+1)
    # get scores for rth doc, ith topic
    for i in range(n_topics):
        score = 0
        for j in range(nwords):
            j1 = j0[1][j]
            if tf[r,j1] != 0:
                score += lda_norm[i][j1] * tf[r,j1]
        rev_score [i+1] = score
        if score>max_score:
            max_score = score
            idx = i
    # Save review's highest scores
    rev_score[0] = idx
    rev_scores [r] = rev_score
    topic_counts[idx] += 1
print('{:<6s}{:>8s}{:>8s}'.format("TOPIC", "REVIEWS", "PERCENT"))
for i in range(n_topics):
    print('{:>3d}{:>10d}{:>8.1%}'.format((i+1), topic_counts[i], \
          topic_counts[i]/n_comments))

topics = pd.DataFrame(rev_scores)
topics=topics.iloc[:,0]
topics = pd.DataFrame(topics)
topics.columns=['Topic']
df=pd.concat([df,topics],axis=1)

# Setup Sentiment dictionary
sentiment_dic = {}
for i in range(len(sw)):
    sentiment_dic[sw.iloc[i][0]] = sw.iloc[i][1]

#Define the preprocessor for use with sentiment analysis
cv = CountVectorizer(max_df=1.0, min_df=1, max_features=None, \
                     preprocessor=my_preprocessor, ngram_range=(1,2))
tf = cv.fit_transform(df['Review'])
terms = cv.get_feature_names()
n_reviews = tf.shape[0]
n_terms = tf.shape[1]
print('{:.<22s}{:>6d}'.format("Number of Reviews", n_reviews))
print('{:.<22s}{:>6d}'.format("Number of Terms", n_terms))
print("\nTopics Identified using LDA with TF_IDF")
display_topics(lda.components_, terms, n_terms=15)

#Caculate semtiment for each document and for the whole corpus
min_sentiment = +5
```

```
max_sentiment = -5
avg_sentiment, min, max = 0,0,0
min_list, max_list = [],[]
sentiment_score = [0]*n_reviews
for i in range(n_reviews):
    # Iterate over the terms with nonzero scores
    n_sw = 0
    term_list = tf[i].nonzero()[1]
    if len(term_list)>0:
        for t in np.nditer(term_list):
            3
            score = sentiment_dic.get(terms[t])
            if score != None:
                sentiment_score[i] += score * tf[i,t]
                n_sw += tf[i,t]
    if n_sw>0:
        sentiment_score[i] = sentiment_score[i]/n_sw
    if sentiment_score[i]==max_sentiment and n_sw>3:
        max_list.append(i)
    if sentiment_score[i]>max_sentiment and n_sw>3:
        max_sentiment=sentiment_score[i]
        max = i
        max_list = [i]
    if sentiment_score[i]==min_sentiment and n_sw>3:
        min_list.append(i)
    if sentiment_score[i]<min_sentiment and n_sw>3:
        min_sentiment=sentiment_score[i]
        min = i
        min_list = [i]
    avg_sentiment += sentiment_score[i]
avg_sentiment = avg_sentiment/n_reviews
print("\nCorpus Average Sentiment: ", avg_sentiment)
print("\nMost Negative Reviews with 4 or more Sentiment Words:")
for i in range(len(min_list)):
    print("{:<s}{:<d}{:<s}{:<5.2f}".format(" Review ", min_list[i], \
          " Sentiment is ", min_sentiment))
print("\nMost Positive Reviews with 4 or more Sentiment Words:")
for i in range(len(max_list)):
    print("{:<s}{:<d}{:<s}{:<5.2f}".format(" Review ", max_list[i], \
          " Sentiment is ", max_sentiment))

#Calculate the averages by hotel, cluster and hotel x cluster
#Add sentiment score into a new df
sens=pd.DataFrame({'Score':sentiment_score})
df=pd.concat([df,sens],axis=1)
hotels = ["Bally's",'Bellagio','Circus Circus','Encore','Excalibur']

print("\n**** Average Sentiment by Hotel ****")
for h in hotels:
    idx = df.index[df['hotel'] == h]
    this_hotel = df.loc[idx]
    sc = this_hotel['Score'].mean()
    print("\nAverage Sentiment for", h, ":", sc)

print("\n**** Average Sentiment by Topic ****")
for c in range(0,7):
    idx = df.index[df['Topic'] == c]
    this_topic = df.loc[idx]
    sc = this_topic['Score'].mean()
    print("\nAverage Sentiment for Topic", c, ":", sc)
hi_topic = 2
low_topic = 4

print("\n**** Average Sentiment by Hotel x Topic ****")
for h in hotels:
    idx = df.index[df['hotel'] == h]
    this_hotel = df.loc[idx]
    for c in range(0,7):
        idx = this_hotel.index[this_hotel['Topic'] == c]
        this_topic = this_hotel.loc[idx]
        sc = this_topic['Score'].mean()
        print("\nAverage Sentiment for" ,h, "Topic", c, ":", sc)

#Word Clouds
def shades_of_grey(word, font_size, position, orientation, random_state=None, \
                   **kwargs):
    return "hsl(0, 0%%, %d%%)" % random.randint(60,1000)
#Word cloud for all word in the reviews
st = set(STOPWORDS)
st.add("hotel")
st.add("room")
st.add("quot")
st.add("one")
st.add("casino")
wc = WordCloud(stopwords=st,width=600, height=400)
s = ""
for i in range(len(comments)):
    s += comments[i]
wc.generate(s)
# Display the word cloud.
plt.imshow(wc, interpolation="bilinear")
plt.axis("off")
plt.figure()
plt.show()

#From the sentiment words of all words
corpus_sentiment = {}
n_sw = 0
for i in range(n_reviews):
    # Iterate over the terms with nonzero scores
    term_list = tf[i].nonzero()[1]
    if len(term_list)>0:
            for t in np.nditer(term_list):
                score = sentiment_dic.get(terms[t])
                if score != None:
                    n_sw += tf[i,t]
                    current_count = corpus_sentiment.get(terms[t])
                    if current_count == None:
                        corpus_sentiment[terms[t]] = tf[i,t]
                    else:
                        corpus_sentiment[terms[t]] += tf[i,t]
print("The Corpus contains a total of ", len(corpus_sentiment), " unique sentiment
words")
print("The total number of sentiment words in the Corpus is", n_sw)

wc = WordCloud(background_color="green", max_words=200, stopwords=sw, \
max_font_size=40, min_font_size=10, prefer_horizontal=0.7, \
relative_scaling=0.5, width=400, height=200, \
margin=10, random_state=341)
wc.generate_from_frequencies(corpus_sentiment)
plt.imshow(wc.recolor(color_func=shades_of_grey, random_state=3),
interpolation="bilinear")
plt.axis("off")
plt.figure()

#Words in cluster with lowest sentiment score
idx=df.index[df['Topic'] == low_topic]
low = df.loc[idx]
wc = WordCloud(stopwords=st, width=600, height=400)
s = ""
low_topics=low['Review']
for i in idx:
    s += low_topics[i]
wc.generate(s)
plt.imshow(wc, interpolation="bilinear")
plt.axis("off")
plt.figure()
plt.show()

#Sentiment Words in cluster with lowest sentiment score
low_sentiment = {}
n_sw = 0
for i in idx:
    # Iterate over the terms with nonzero scores
    term_list = tf[i].nonzero()[1]
    if len(term_list)>0:
            for t in np.nditer(term_list):
                score = sentiment_dic.get(terms[t])
                if score != None:
                    n_sw += tf[i,t]
                    current_count = low_sentiment.get(terms[t])
                    if current_count == None:
                        low_sentiment[terms[t]] = tf[i,t]
                    else:
                        low_sentiment[terms[t]] += tf[i,t]

wc = WordCloud(background_color="red", max_words=200, stopwords=sw, \
max_font_size=40, min_font_size=10, prefer_horizontal=0.7, \
relative_scaling=0.5, width=400, height=200, \
margin=10, random_state=341)
wc.generate_from_frequencies(low_sentiment)
plt.imshow(wc.recolor(color_func=shades_of_grey, random_state=3),
interpolation="bilinear")
plt.axis("off")
plt.figure()

#Words in cluster with higest sentiment score
idx=df.index[df['Topic'] == hi_topic]
hi = df.loc[idx]
wc = WordCloud(stopwords=st,width=600, height=400)
s = ""
hi_topics=hi['Review']
for i in idx:
    s += hi_topics[i]
wc.generate(s)
plt.imshow(wc, interpolation="bilinear")
plt.axis("off")
plt.figure()
plt.show()

#Sentiment Words in cluster with highest sentiment score
hi_sentiment = {}
n_sw = 0
for i in idx:
    # Iterate over the terms with nonzero scores
    term_list = tf[i].nonzero()[1]
    if len(term_list)>0:
            for t in np.nditer(term_list):
                score = sentiment_dic.get(terms[t])
                if score != None:
                    n_sw += tf[i,t]
                    current_count = hi_sentiment.get(terms[t])
                    if current_count == None:
                        hi_sentiment[terms[t]] = tf[i,t]
                    else:
                        hi_sentiment[terms[t]] += tf[i,t]

wc = WordCloud(background_color="blue", max_words=200, stopwords=sw, \
max_font_size=40, min_font_size=10, prefer_horizontal=0.7, \
relative_scaling=0.5, width=400, height=200, \
margin=10, random_state=341)
wc.generate_from_frequencies(hi_sentiment)
plt.imshow(wc.recolor(color_func=shades_of_grey, random_state=3),
interpolation="bilinear")
plt.axis("off")
plt.figure()
```

OUTPUT

Description of 7 topics using TFIDF with LDA

```
Conducting Term/Frequency Matrix using TF-IDF

TOPIC  REVIEWS PERCENT
  1       39    2.3%
  2       44    2.6%
  3      181   10.8%
  4      235   14.1%
  5      518   31.0%
  6      416   24.9%
  7      238   14.2%
Number of Reviews.....  1671
Number of Terms.......132979

Topics Identified using LDA with TF_IDF
Topic #1:
+40 bucks     +2009 birthday +45          +3pm told     +18 for
+2008 and     +100k          +104 38      +15 am        +21c there
+1st          +80s with      +4am so      +2009 wonder  +11am on

Topic #2:
+2009 lock    +40 understand +21 in       +150 per      +2009 fancy
+2009 clean   +27 june       +21c there   +13th floor   +16 year
+90 for       +12            +1980 but    +3770         +5pm to

Topic #3:
+12 yr        +11am on       +60 and      +16 year      +12 we
+100 minimum  +2009 bellagio +2008 whats  +137          +2009 write
+10th this    +40ft          +80 the      +2009 birthday +2009 bally

Topic #4:
+1980s complete+99 have      +30 more     +10 minutes   +40ft
+55th floor   +50 deposit    +39          +1030         +7th trip
+107          +38 plus       +2009 castle +2009 for     +25 ll

Topic #5:
+12th the     +1st timers    +2009 staying +2009 bravo  +7pm my
+50 it        +300 in        +7nights firstly+30 more   +2009 castle
+2009 implode +12 wonderful  +2008 pleasantly+23 seconds  +20us

Topic #6:
+107          +2009 castle   +7th trip    +10am         +20per day
+2008 keep    +3pm told      +50 so       +13 18        +7pm my
+30 more      +7nights firstly+34         +2009 bravo   +1st wedding

Topic #7:
```

```
+6nyts      +100 the     +2009 castle  +25 which    +25 ll
+20per day    +50 cheaply   +13 18      +800 or      +295
+3pm told     +2009 bravo   +104 38      +100 times    +100 on
```

Overall average sentiment (weighted)

```
Corpus Average Sentiment:  1.2694687660496435
```

****** Average Sentiment by Hotel ******

```
Average Sentiment for Bally's : 1.2638660586224275
Average Sentiment for Bellagio : 1.7098813474447252
Average Sentiment for Circus Circus : 0.917568471858523
Average Sentiment for Encore : 1.3656068366129321
Average Sentiment for Excalibur : 1.0947263179104267
```

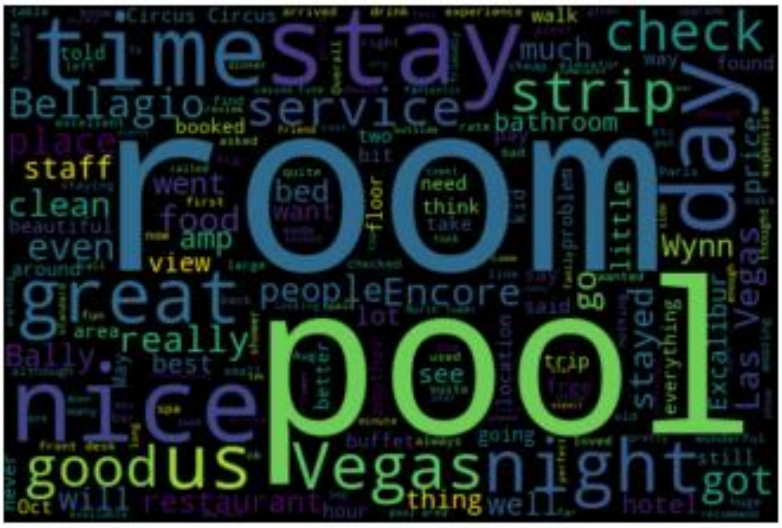
****** Average Sentiment by Topic ******

```
Average Sentiment for Topic 0 : -0.13334079924862588
Average Sentiment for Topic 1 : 0.5186941372283264
Average Sentiment for Topic 2 : 0.23695616317869178
Average Sentiment for Topic 3 : 1.527337763324764
Average Sentiment for Topic 4 : 1.1931866926972352
Average Sentiment for Topic 5 : 1.9188684178236513
Average Sentiment for Topic 6 : 1.199691801063593
```

****** Average Sentiment by Hotel x Topic ******

```
Average Sentiment for Bally's Topic 0 : -0.057372039724980874
Average Sentiment for Bally's Topic 1 : 0.182183908045977
Average Sentiment for Bally's Topic 2 : 0.19468277866293984
Average Sentiment for Bally's Topic 3 : 2.3333333333333335
Average Sentiment for Bally's Topic 4 : 1.212375992063492
Average Sentiment for Bally's Topic 5 : 1.9052496184577545
Average Sentiment for Bally's Topic 6 : 1.2702945934463898
Average Sentiment for Bellagio Topic 0 : 0.45256916996047436
Average Sentiment for Bellagio Topic 1 : 1.2352400178132474

Average Sentiment for Bellagio Topic 2 : 0.7328583059308266
Average Sentiment for Bellagio Topic 3 : 1.5081568337238314
Average Sentiment for Bellagio Topic 4 : 1.8608225108225107
Average Sentiment for Bellagio Topic 5 : 1.9050388308128507
Average Sentiment for Bellagio Topic 6 : 1.2986243386243386
Average Sentiment for Circus Circus Topic 0 : -0.582054821185256
Average Sentiment for Circus Circus Topic 1 : 0.12525661155523854
Average Sentiment for Circus Circus Topic 2 : -0.3398889180702692
Average Sentiment for Circus Circus Topic 3 : nan
Average Sentiment for Circus Circus Topic 4 : 1.1220181599190802
Average Sentiment for Circus Circus Topic 5 : 1.6630045109211777
Average Sentiment for Circus Circus Topic 6 : 1.0172239836105383
Average Sentiment for Encore Topic 0 : 0.4045113806732997
Average Sentiment for Encore Topic 1 : 0.8045600830588868
Average Sentiment for Encore Topic 2 : 0.3564646434530408
Average Sentiment for Encore Topic 3 : 1.5189474681589716
Average Sentiment for Encore Topic 4 : 1.8370146520146524
Average Sentiment for Encore Topic 5 : 2.0833609948433054
Average Sentiment for Encore Topic 6 : 1.2025545634920634
Average Sentiment for Excalibur Topic 0 : 0.012412730243612606
Average Sentiment for Excalibur Topic 1 : 0.07080983709273181
Average Sentiment for Excalibur Topic 2 : 0.049852202035183125
Average Sentiment for Excalibur Topic 3 : 3.0
Average Sentiment for Excalibur Topic 4 : 1.2198909535101556
Average Sentiment for Excalibur Topic 5 : 1.987781884781885
Average Sentiment for Excalibur Topic 6 : 0.8495660606084283
```

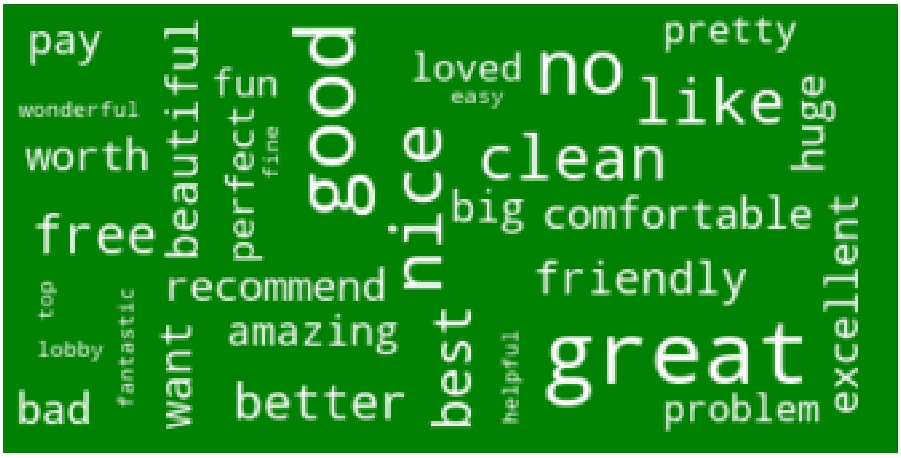
Word Cloud for the words in all the reviews



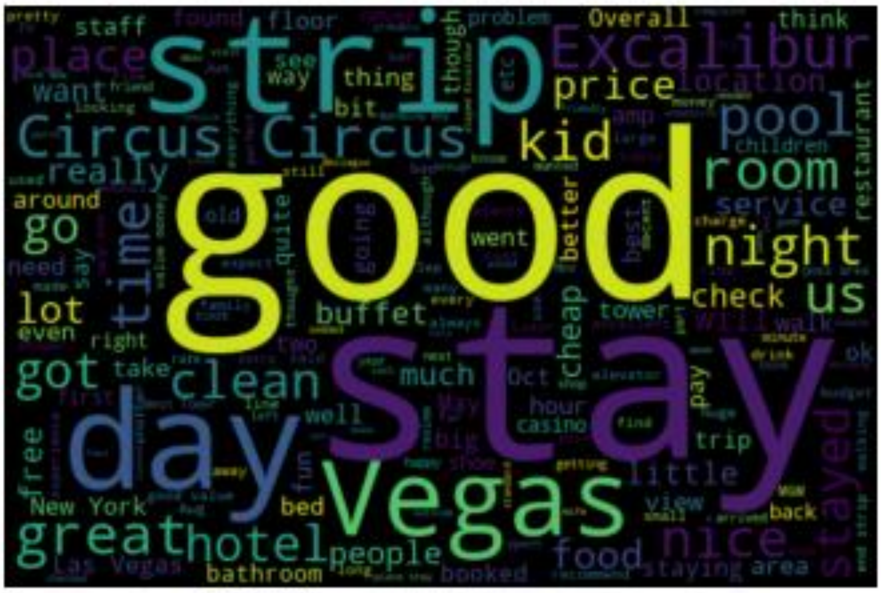
Word Cloud for only the Sentiment words in all the reviews

The Corpus contains a total of **1150** unique sentiment words

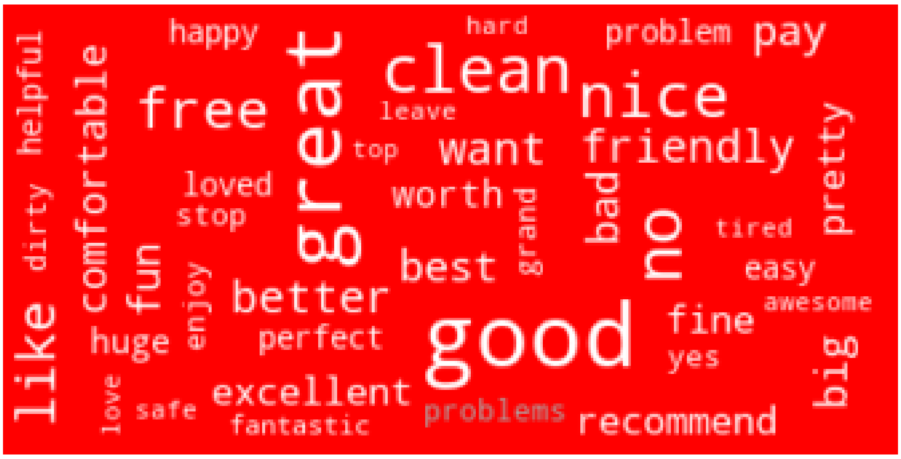
The total number of sentiment words in the Corpus is **23572**



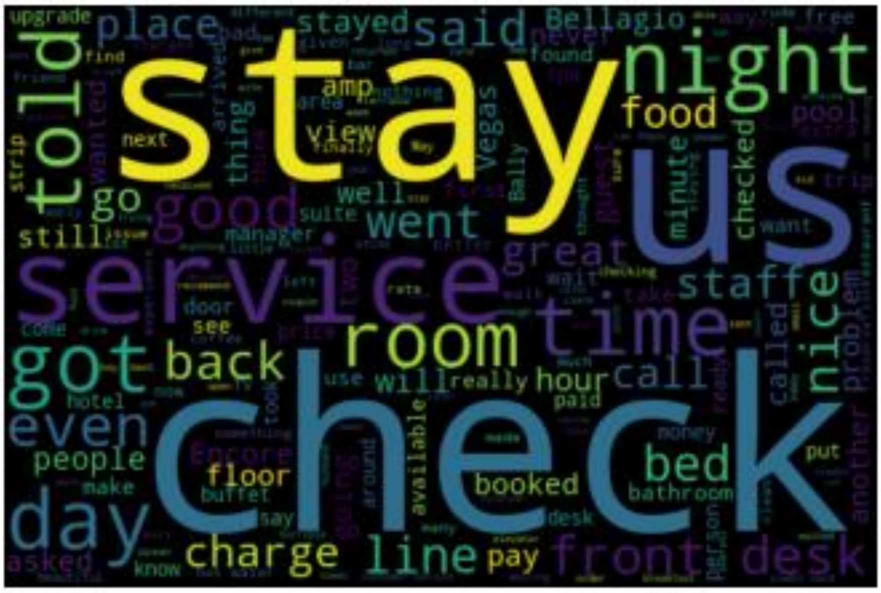
Word Cloud for the words in the topic cluster with the lowest sentiment score.



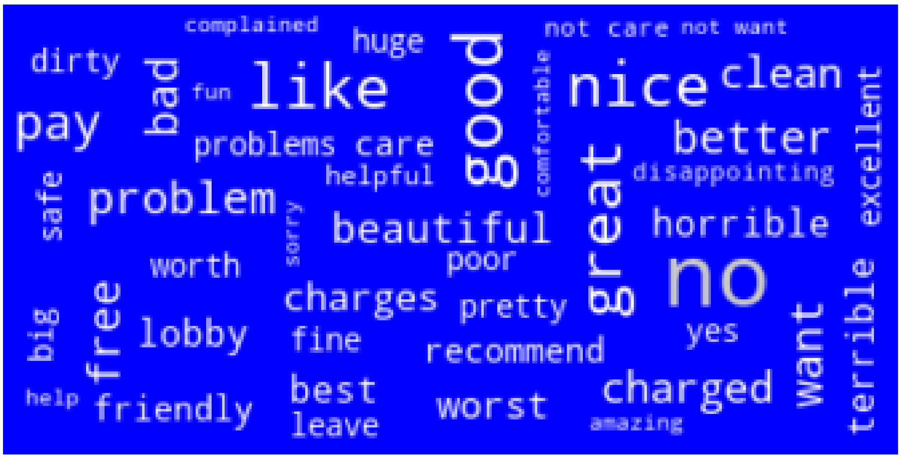
Word Cloud for the Sentiment words in the topic cluster with the lowest sentiment score.



Word Cloud for the words in the topic cluster with the highest sentiment score.



Word Cloud for the Sentiment words in the topic cluster with the highest sentiment score


complained
huge
not care
not want
dirty
bad
fun
like
good
comfortable
nice
clean
pay
problems
care
better
excellent
helpful
disappointing
safe
problem
sorry
beautiful
great
horrible
worth
poor
no
charges
pretty
big
free
lobby
fine
yes
want
recommend
terrible
help
friendly
best
leave
worst
charged
amazing