Topic modelling

May 3, 2021

1 1.Obtaining and Viewing the Data

[1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
plt.style.use('seaborn')
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

import time
import datetime
```

[2]:
```
df_1 = pd.read_csv(r'AB_BEIJING_reviews.csv')
# checking shape ...
print("The dataset has {} rows and {} columns.".format(*df_1.shape))

# ... and duplicates
print("It contains {} duplicates.".format(df_1.duplicated().sum()))
```

```
The dataset has 12124 rows and 1 columns.
It contains 284 duplicates.
```

[3]:
```
df_1.head()
```

[3]:

	name
0	Mavis is such a great host - very precise dire…
1	Mary is the best and competent landlord I ever…
2	marvelous
3	Many thanks to Kevin for the beautiful and lux…
4	_______ _ _ _ _, _ _ _ __ ,__ ___!_____ _…

It may be valuable to have more details, such as the latitude and longitude of the accommodation that has been reviewed, the neighbourhood it's in, the host id, etc.

To get this information, let's combine our reviews_dataframe with the listings_dataframe and take

only the columns we need from the latter one:

Hosts with many properties

By the way, I am curious to find out if any private hosts have started to run a professional business through Airbnb - at least this is what was in the press. Let's work this out:

2 Language Detection

[25]:
```
# we use Python's langdetect
from langdetect import detect
```

[26]:
```
# the function that detects the language
def language_detection(text):
    try:
        return detect(text)
    except:
        return None
```

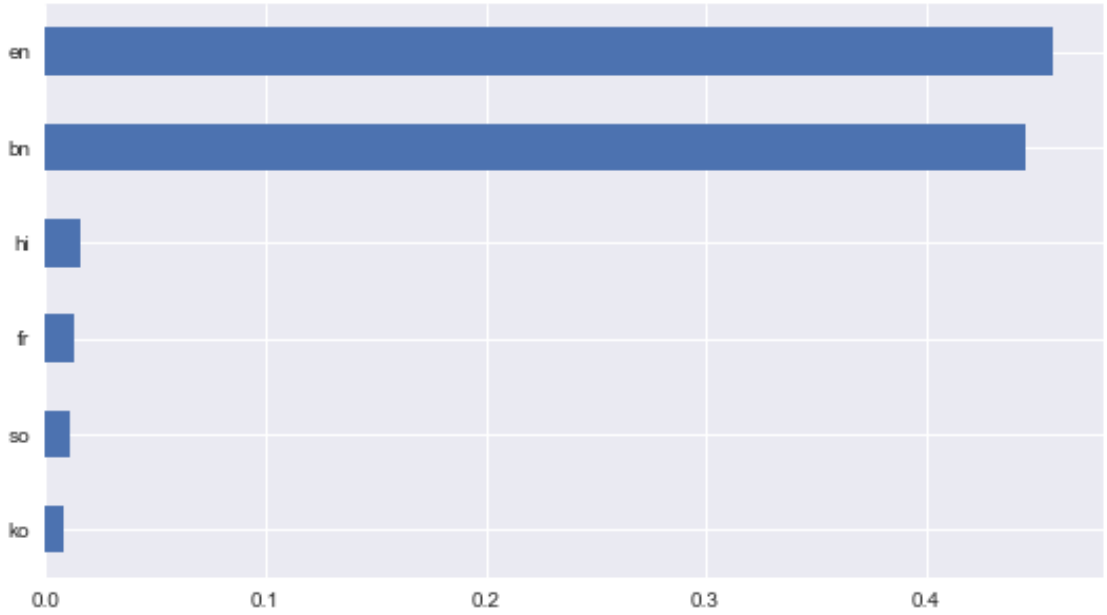
[28]:
```
%%time
df['language'] = df['name'].apply(language_detection)
```

```
CPU times: user 4min 12s, sys: 710 ms, total: 4min 13s
Wall time: 4min 13s
```

[29]:
```
df.language.value_counts().head(10)
```

[29]:
```
en       5534
bn       5392
hi        188
fr        155
so        133
ko        102
es         97
zh-cn      84
de         61
ro         41
Name: language, dtype: int64
```

[30]:
```
# visualizing the comments' languages a) quick and dirty
ax = df.language.value_counts(normalize=True).head(6).sort_values().
 ↪plot(kind='barh', figsize=(9,5));
```



[31]:
```
# splitting the dataframes in language related sub-dataframes
df_eng = df[(df['language']=='en')]
df_de  = df[(df['language']=='de')]
df_fr  = df[(df['language']=='fr')]
```

3 3.Visualizing the Data with WordClouds

[32]:
```
# import necessary libraries
from nltk.corpus import stopwords
from wordcloud import WordCloud
from collections import Counter
from PIL import Image

import re
import string
```

[33]:
```
# word cloud function
def plot_wordcloud(wordcloud, language):
    plt.figure(figsize=(12, 10))
    plt.imshow(wordcloud, interpolation = 'bilinear')
    plt.axis("off")
    plt.title(language + ' name\n', fontsize=18, fontweight='bold')
    plt.show()
```

English WordCloud

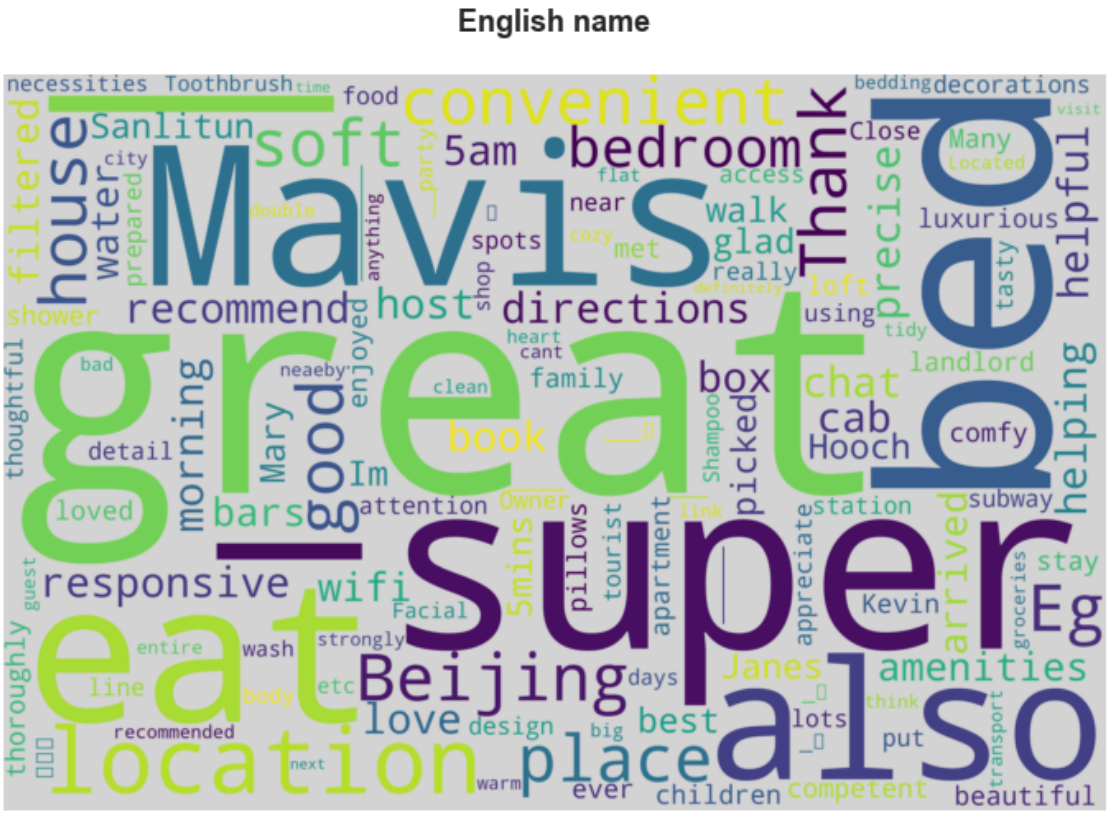
[35]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/utility/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[35]: True

[37]:
```
wordcloud = WordCloud(max_font_size=None, max_words=200,␣
↪background_color="lightgrey",
                      width=3000, height=2000,
                      stopwords=stopwords.words('english')).generate(str(df_eng.
↪name.values))

plot_wordcloud(wordcloud, 'English')
```



4 4.Sentiment Analysis

using to VADER package

[38]:
```
import nltk
nltk.downloader.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/utility/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

[38]: True

[39]:
```
# load the SentimentIntensityAnalyser object in
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# assign it to another name to make it easier to use
analyzer = SentimentIntensityAnalyzer()
```

[40]:
```
# use the polarity_scores() method to get the sentiment metrics
def print_sentiment_scores(sentence):
    snt = analyzer.polarity_scores(sentence)
    print("{} {}".format(sentence, str(snt)))
```

[41]:
```
print_sentiment_scores("This raspberry cake is good.")
```

```
This raspberry cake is good. {'neg': 0.0, 'neu': 0.58, 'pos': 0.42, 'compound':
0.4404}
```

VADER produces four sentiment metrics from these word ratings, which you can see above. The first three - positive, neutral and negative - represent the proportion of the text that falls into those categories. As you can see, our example sentence was rated as 42% positive, 58% neutral, and 0% negative.

The final metric, the compound score, is the sum of all of the lexicon ratings which have been standardised to range between -1 and 1. In this case, our example sentence has a rating of 0.44, which is pretty neutral.

[42]:
```
print_sentiment_scores("This raspberry cake is good.")
print_sentiment_scores("This raspberry cake is VERY GOOD!!")
print_sentiment_scores("This raspberry cake is really GOOD! But the coffee is␣
↪dreadful.")
```

```
This raspberry cake is good. {'neg': 0.0, 'neu': 0.58, 'pos': 0.42, 'compound':
0.4404}
This raspberry cake is VERY GOOD!! {'neg': 0.0, 'neu': 0.488, 'pos': 0.512,
'compound': 0.7386}
This raspberry cake is really GOOD! But the coffee is dreadful. {'neg': 0.18,
'neu': 0.558, 'pos': 0.262, 'compound': 0.3222}
```

[43]:
```
# getting only the negative score
def negative_score(text):
    negative_value = analyzer.polarity_scores(text)['neg']
    return negative_value
```

```
# getting only the neutral score
def neutral_score(text):
    neutral_value = analyzer.polarity_scores(text)['neu']
    return neutral_value

# getting only the positive score
def positive_score(text):
    positive_value = analyzer.polarity_scores(text)['pos']
    return positive_value

# getting only the compound score
def compound_score(text):
    compound_value = analyzer.polarity_scores(text)['compound']
    return compound_value
```

Calculating Sentiment Scores for english language

[45]:
```
%%time

df_eng['sentiment_neg'] = df_eng['name'].apply(negative_score)
df_eng['sentiment_neu'] = df_eng['name'].apply(neutral_score)
df_eng['sentiment_pos'] = df_eng['name'].apply(positive_score)
df_eng['sentiment_compound'] = df_eng['name'].apply(compound_score)
```

```
CPU times: user 6.82 s, sys: 19.9 ms, total: 6.84 s
Wall time: 6.84 s
```

[46]:
```
df_eng['sentiment_neg'].sort_values(ascending=False).
↪head(),df_eng['sentiment_pos'].sort_values(ascending=False).
↪head(),df_eng['sentiment_neu'].sort_values(ascending=False).
↪head(),df_eng['sentiment_compound'].sort_values(ascending=True).head()
```

[46]:
```
(2112    0.705
 1466    0.626
 4922    0.573
 5138    0.565
 4484    0.532
 Name: sentiment_neg, dtype: float64,
 1946    1.0
 877     1.0
 3602    1.0
 3705    1.0
 3707    1.0
 Name: sentiment_pos, dtype: float64,
 2503    1.0
 2513    1.0
```

```
2532    1.0
5905    1.0
2556    1.0
Name: sentiment_neu, dtype: float64,
2417   -0.9897
750    -0.9770
1011   -0.9536
4234   -0.9450
42     -0.9382
Name: sentiment_compound, dtype: float64)
```

[47]:
```
df = df_eng
df.head(2)
```

[47]:

	name	language	sentiment_neg	sentiment_neu	sentiment_pos	sentiment_compound
0	Mavis is such a great host - very precise dire…	en	0.0	0.581	0.419	0.9939
1	Mary is the best and competent landlord I ever…	en	0.0	0.519	0.481	0.7579

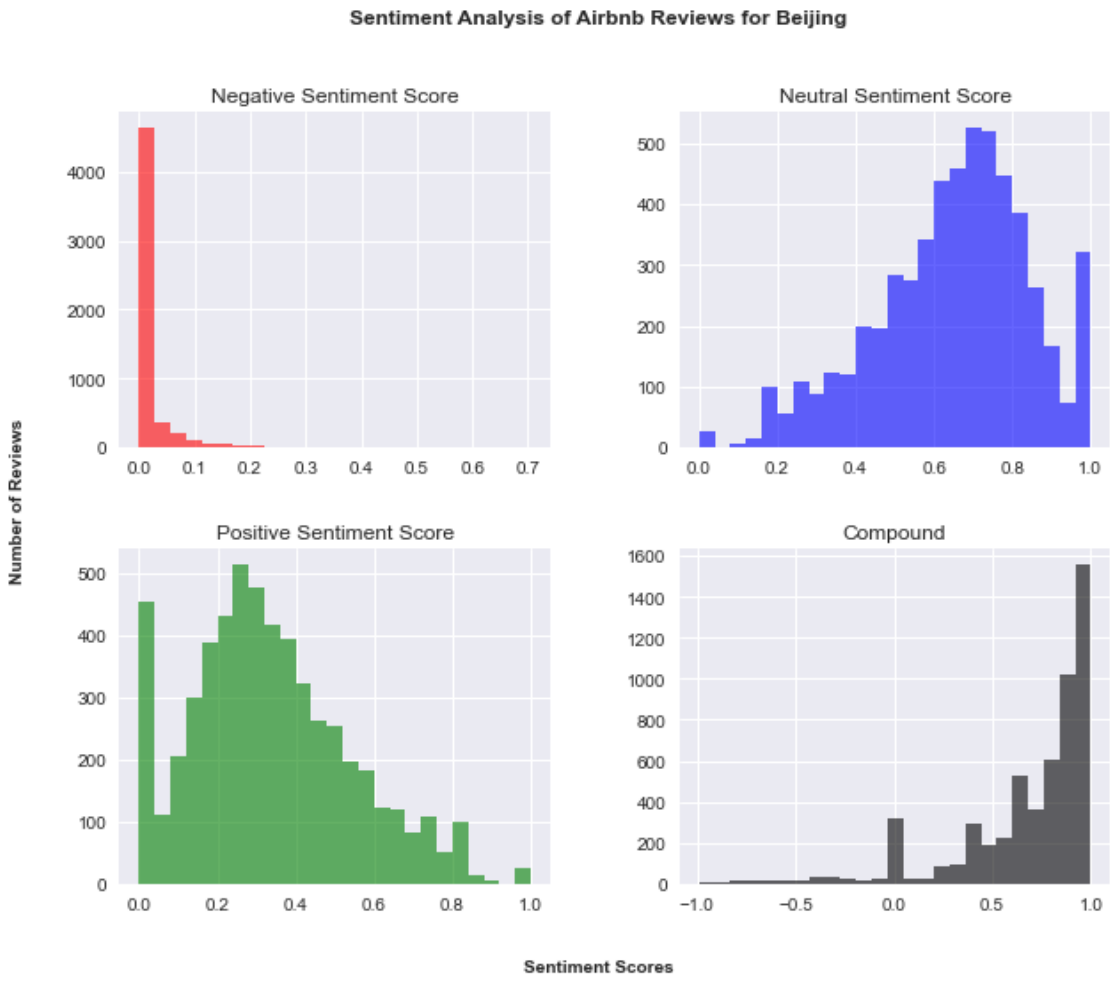
Let's investigate the distribution of all scores:

[48]:
```
# all scores in 4 histograms
fig, axes = plt.subplots(2, 2, figsize=(10,8))

# plot all 4 histograms
df.hist('sentiment_neg', bins=25, ax=axes[0,0], color='red', alpha=0.6)
axes[0,0].set_title('Negative Sentiment Score')
df.hist('sentiment_neu', bins=25, ax=axes[0,1], color='blue', alpha=0.6)
axes[0,1].set_title('Neutral Sentiment Score')
df.hist('sentiment_pos', bins=25, ax=axes[1,0], color='green', alpha=0.6)
axes[1,0].set_title('Positive Sentiment Score')
df.hist('sentiment_compound', bins=25, ax=axes[1,1], color='black', alpha=0.6)
axes[1,1].set_title('Compound')

# plot common x- and y-label
fig.text(0.5, 0.04, 'Sentiment Scores',  fontweight='bold', ha='center')
fig.text(0.04, 0.5, 'Number of Reviews', fontweight='bold', va='center',␣
 ↪rotation='vertical')

# plot title
plt.suptitle('Sentiment Analysis of Airbnb Reviews for Beijing\n\n',␣
 ↪fontsize=12, fontweight='bold');
```



[51]:
```
percentiles = df.sentiment_compound.describe(percentiles=[.05, .1, .2, .3, .4, .
↪5, .6, .7, .8, .9])
percentiles
```

[51]:
```
count    5534.000000
mean        0.684697
std         0.345017
min        -0.989700
5%          0.000000
10%         0.078520
20%         0.474000
30%         0.624900
40%         0.734600
50%         0.816900
60%         0.874800
70%         0.915110
```

```
80%          0.942200
90%          0.967870
max          0.999300
Name: sentiment_compound, dtype: float64
```

[ ]:

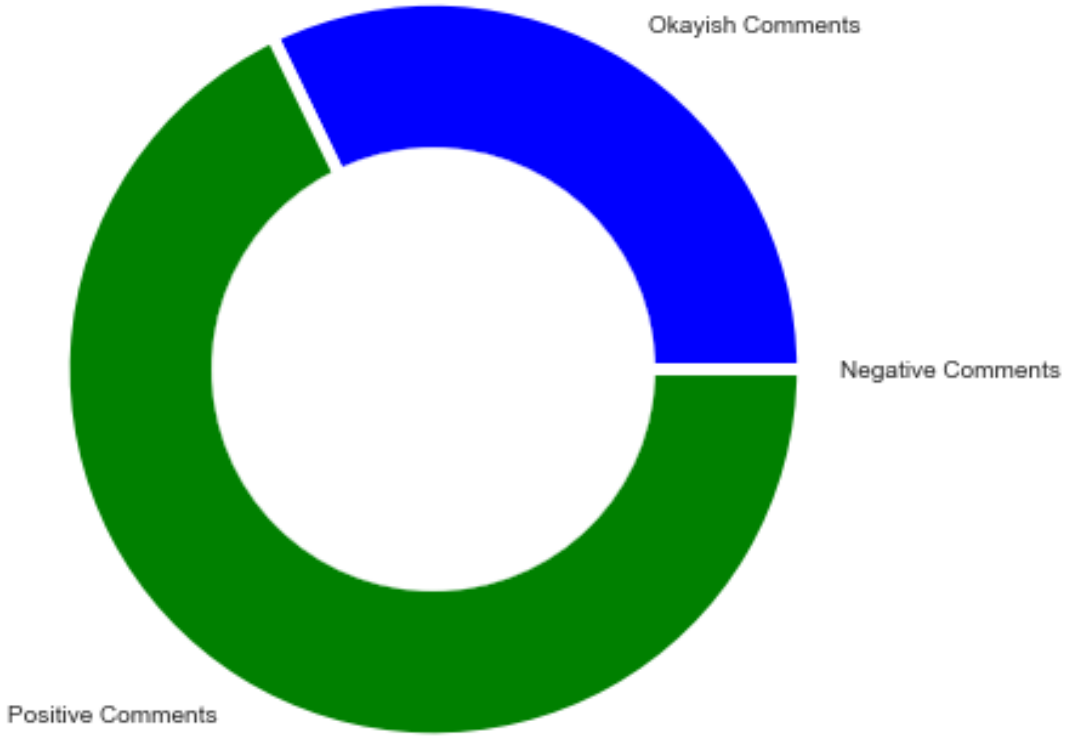
Comparing Negative and Positive Comments

[54]:
```
# assign the data
neg = percentiles['5%']
mid = percentiles['20%']
pos = percentiles['max']
names = ['Negative Comments', 'Okayish Comments','Positive Comments']
size = [neg, mid, pos]

# call a pie chart
plt.pie(size, labels=names, colors=['red', 'blue', 'green'], pctdistance=0.8,
        wedgeprops={'linewidth':5, 'edgecolor':'white' })

# create circle for the center of the plot to make the pie look like a donut
my_circle = plt.Circle((0,0), 0.6, color='white')

# plot the donut chart
fig = plt.gcf()
fig.set_size_inches(7,7)
fig.gca().add_artist(my_circle)
plt.show()
```



Clearly, the bulk of the reviews are tremendously positive. lets see what the negative and positive comments are about

[55]:
```
((df.sentiment_compound)>=0.95).value_counts()
```

[55]:
```
False    4585
True      949
Name: sentiment_compound, dtype: int64
```

[56]:
```
# full dataframe with POSITIVE comments
df_pos = df.loc[df.sentiment_compound >= 0.95]
print(len(df_pos))
# only corpus of POSITIVE comments
pos_comments = df_pos['name'].tolist()
```

```
949
```

[57]:
```
((df.sentiment_compound)<0.0).value_counts()
```

[57]:
```
False    5318
True      216
Name: sentiment_compound, dtype: int64
```

[59]:
```
#full dataframe with NEGATIVE comments
df_neg = df.loc[df.sentiment_compound < 0.0]
print(len(df_neg))
# only corpus of NEGATIVE comments
neg_comments = df_neg['name'].tolist()
```

```
216
```

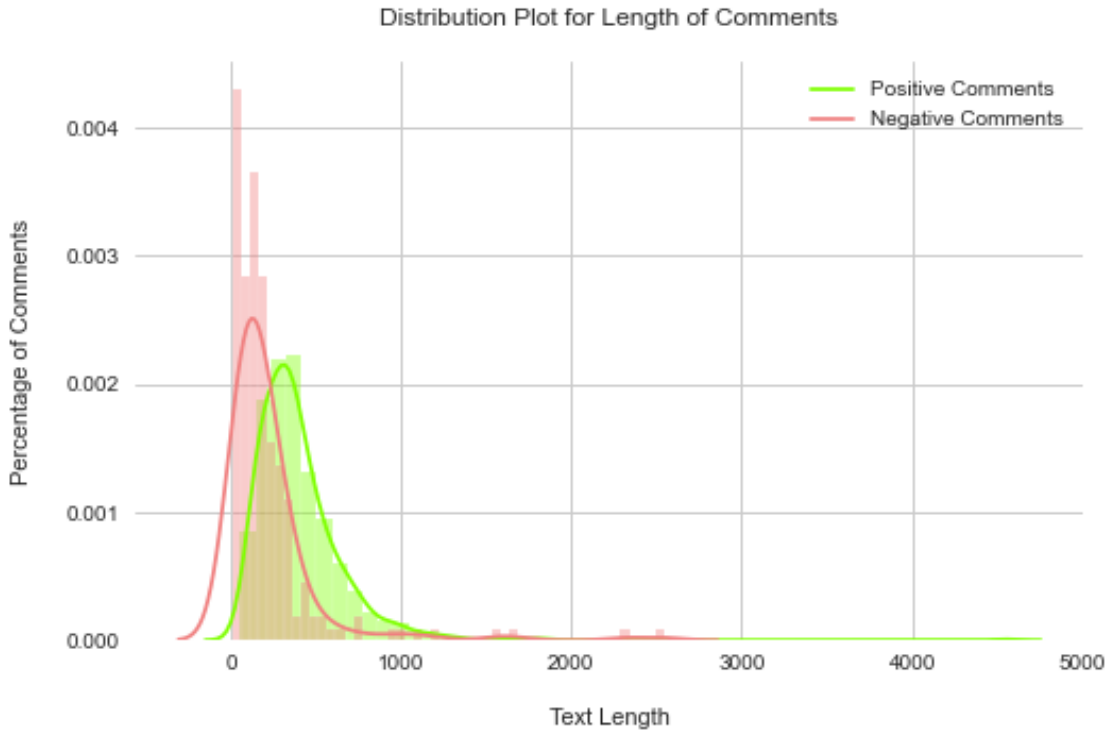
Let's compare the length of both positive and negative comments:

[60]:
```
df_pos['text_length'] = df_pos['name'].apply(len)
df_neg['text_length'] = df_neg['name'].apply(len)
```

[61]:
```
sns.set_style("whitegrid")
plt.figure(figsize=(8,5))

sns.distplot(df_pos['text_length'], kde=True, bins=50, color='chartreuse')
sns.distplot(df_neg['text_length'], kde=True, bins=50, color='lightcoral')

plt.title('\nDistribution Plot for Length of Comments\n')
plt.legend(['Positive Comments', 'Negative Comments'])
plt.xlabel('\nText Length')
plt.ylabel('Percentage of Comments\n');
```



The mode for the text length of positive comments can be found more to the right than for the negative comments, which means most of the positive comments are longer than most of the negative comments. But the tail for negative comments is thicker.

Frequency Distribution

Another method for visually exploring text is with frequency distributions. In the context of a text corpus, such a distribution tells us the prevalence of certain words. Here we use the Yellowbrick library.

[66]:
```
pip install yellowbrick
```

```
WARNING: Value for scheme.headers does not match. Please report this to
<https://github.com/pypa/pip/issues/9617>
distutils: /Users/utility/opt/anaconda3/include/python3.8/UNKNOWN
sysconfig: /Users/utility/opt/anaconda3/include/python3.8
WARNING: Additional context:
user = False
home = None
root = None
prefix = None
```

```
Collecting yellowbrick
  Downloading yellowbrick-1.3.post1-py3-none-any.whl (271 kB)
     |                  | 271 kB 8.2 MB/s eta 0:00:01
Collecting numpy<1.20,>=1.16.0
  Using cached numpy-1.19.5-cp38-cp38-macosx_10_9_x86_64.whl (15.6 MB)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(3.3.2)
Requirement already satisfied: scipy>=1.0.0 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(1.5.2)
Requirement already satisfied: cycler>=0.10.0 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(0.10.0)
Requirement already satisfied: scikit-learn>=0.20 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(0.23.2)
Requirement already satisfied: six in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from
cycler>=0.10.0->yellowbrick) (1.15.0)
Requirement already satisfied: certifi>=2020.06.20 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2020.12.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (8.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=0.20->yellowbrick) (2.1.0)
Requirement already satisfied: joblib>=0.11 in
/Users/utility/opt/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=0.20->yellowbrick) (1.0.0)
Installing collected packages: numpy, yellowbrick
  Attempting uninstall: numpy
    Found existing installation: numpy 1.20.2
    Uninstalling numpy-1.20.2:
      Successfully uninstalled numpy-1.20.2
```

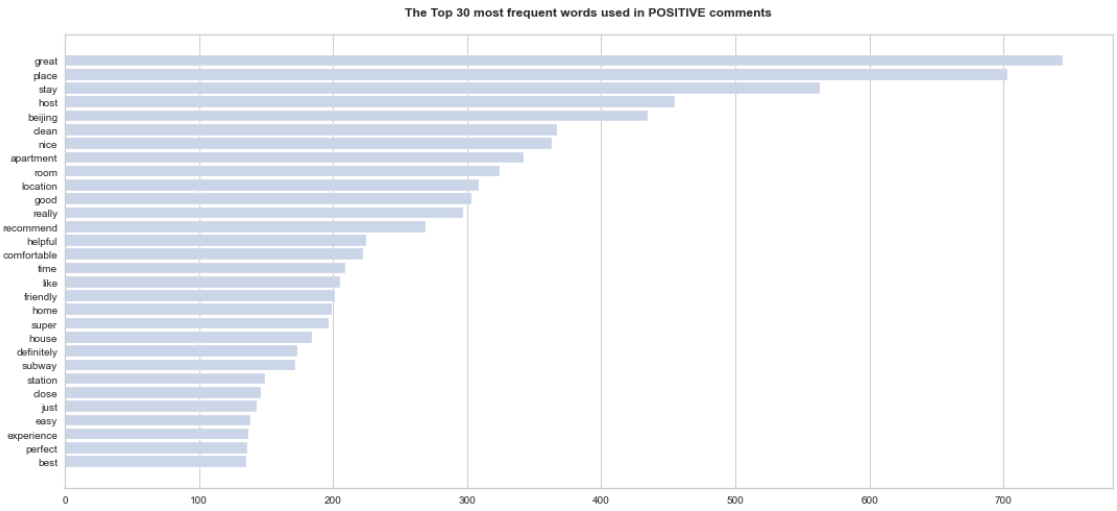
```
WARNING: Value for scheme.headers does not match. Please report this to
<https://github.com/pypa/pip/issues/9617>
distutils: /Users/utility/opt/anaconda3/include/python3.8/UNKNOWN
sysconfig: /Users/utility/opt/anaconda3/include/python3.8
WARNING: Additional context:
user = False
home = None
root = None
prefix = None
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
tensorflow 2.4.1 requires grpcio~=1.32.0, but you have grpcio 1.37.1 which is
incompatible.
pyldavis 3.3.1 requires numpy>=1.20.0, but you have numpy 1.19.5 which is
incompatible.
Successfully installed numpy-1.19.5 yellowbrick-1.3.post1
WARNING: You are using pip version 21.1; however, version 21.1.1 is
available.
You should consider upgrading via the '/Users/utility/opt/anaconda3/bin/python
-m pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[67]:
```
# importing libraries
from sklearn.feature_extraction.text import CountVectorizer
from yellowbrick.text.freqdist import FreqDistVisualizer
from yellowbrick.style import set_palette
```

[68]:
```
# vectorizing text
vectorizer = CountVectorizer(stop_words='english')
docs = vectorizer.fit_transform(pos_comments)
features = vectorizer.get_feature_names()

set_palette('pastel')
plt.figure(figsize=(18,8))
plt.title('The Top 30 most frequent words used in POSITIVE comments\n',␣
 ↪fontweight='bold')
visualizer = FreqDistVisualizer(features=features, n=30)
visualizer.fit(docs)
```

```
visualizer.poof;
```



5 Topic Modelling

[69]:
```
# importing libraries
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import string
```

[70]:
```
# prepare the preprocessing
stop = set(stopwords.words('english'))
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
```

[72]:
```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /Users/utility/nltk_data…
[nltk_data]   Unzipping corpora/wordnet.zip.
```

[72]: True

[73]:
```
# removing stopwords, punctuations and normalizing the corpus
def clean(doc):
    stop_free = " ".join([word for word in doc.lower().split() if word not in␣
 ↪stop])
    punc_free = "".join(token for token in stop_free if token not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized
```

```
doc_clean = [clean(comment).split() for comment in pos_comments]
```

First, we create a Gensim dictionary from the normalized data, then we convert this to a bag-of-words corpus, and save both dictionary and corpus for future use.

[74]:
```
from gensim import corpora
dictionary = corpora.Dictionary(doc_clean)
corpus = [dictionary.doc2bow(text) for text in doc_clean]
```

[75]:
```
import gensim

# let LDA find 3 topics
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=3,␣
 ↪id2word=dictionary, passes=15)
```

[76]:
```
ldamodel.show_topics()
```

[76]:
```
[(0,
  '0.015*"place" + 0.013*"great" + 0.010*"host" + 0.010*"stay" + 0.007*"beijing"
+ 0.007*"time" + 0.006*"apartment" + 0.006*"house" + 0.006*"friendly" +
0.005*"well"'),
 (1,
  '0.012*"apartment" + 0.012*"room" + 0.011*"place" + 0.010*"really" +
0.010*"great" + 0.010*"clean" + 0.009*"good" + 0.009*"also" + 0.008*"host" +
0.008*"u"'),
 (2,
  '0.024*"great" + 0.023*"place" + 0.020*"stay" + 0.017*"host" + 0.014*"beijing"
+ 0.012*"nice" + 0.011*"location" + 0.010*"clean" + 0.009*"u" +
0.008*"recommend"')]
```

[81]:
```
# pip install pyLDAvis
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
pyLDAvis.enable_notebook()

# feed the LDA model into the pyLDAvis instance
lda_viz = gensimvis.prepare(ldamodel, corpus, dictionary)
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[83]:
```
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
pyLDAvis.enable_notebook()

# feed the LDA model into the pyLDAvis instance
lda_viz = gensimvis.prepare(ldamodel, corpus, dictionary)
ldamodel_5 = gensim.models.ldamodel.LdaModel(corpus, num_topics=5,␣
 ↪id2word=dictionary, passes=15)

# uncomment the code if working locally
#ldamodel.save('../input/sentimentData/model5.gensim')

ldamodel_5.show_topics()
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[83]:
```
[(0,
  '0.008*"place" + 0.008*"like" + 0.007*"stay" + 0.007*"nice" + 0.006*"really" +
0.006*"family" + 0.006*"china" + 0.005*"feel" + 0.005*"apartment" +
0.004*"time"'),
 (1,
  '0.025*"place" + 0.018*"great" + 0.018*"stay" + 0.016*"host" + 0.013*"beijing"
+ 0.011*"clean" + 0.011*"nice" + 0.009*"location" + 0.009*"good" +
0.009*"recommend"'),
 (2,
  '0.016*"place" + 0.014*"apartment" + 0.014*"great" + 0.014*"clean" +
0.013*"room" + 0.012*"nice" + 0.009*"host" + 0.008*"really" + 0.008*"u" +
0.008*"stay"'),
 (3,
  '0.026*"great" + 0.018*"place" + 0.016*"stay" + 0.015*"host" + 0.012*"beijing"
+ 0.012*"u" + 0.010*"apartment" + 0.009*"room" + 0.009*"location" +
0.008*"clean"'),
 (4,
  '0.015*"u" + 0.011*"time" + 0.009*"good" + 0.009*"host" + 0.007*"would" +
0.006*"really" + 0.006*"place" + 0.005*"made" + 0.005*"helped" + 0.005*"need"')]
```

5.0.1 * The first topic includes words like place, nice, recommend , host. This sounds like the topic related to the place in whole which is nice and a good host and they would recommend that pace.

5.0.2 * The secound topic include words like restaurant, location, berlin , area ,close . This sounds like a topic related to convenient distances from the accommodation to wherever something interesting was to go to.

5.0.3 * The third topic includes words like station, minute, and walk, and bus, train. This sounds like the topic related to the transport facility from the place they stay. roughly "one munite wlak to bus or train station".

5.0.4 * The fourth topic seems like topic about describing about the place. Like how many bedrooms and the number of bathrooms and kitchen, etc.

5.0.5 * Fifth topic seems like it is telling about the tidiness of the place and also the "great stay" they had and "clean location or neighbourhood" and "great host

[85]:
```
# pyLDAvis.enable_notebook()
# pyLDAvis.gensim.prepare(ldamodel_5, corpus, dictionary)
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-85-505bb8ec58ab> in <module>
      1 pyLDAvis.enable_notebook()
----> 2 pyLDAvis.gensim.prepare(ldamodel_5, corpus, dictionary)

AttributeError: module 'pyLDAvis' has no attribute 'gensim'
```

5.0.6 We observed above 2nd, 3rd, 4th, 5th topics are distinctly different from each other and first topic is the union of 2 nd and 3rd with some other distinct topics.

[86]:
```
ldamodel_8 = gensim.models.ldamodel.LdaModel(corpus, num_topics=8,␣
↪id2word=dictionary, passes=15)


ldamodel_8.show_topics()
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
```

```
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[86]:
```
[(0,
  '0.017*"great" + 0.016*"place" + 0.013*"stay" + 0.013*"u" + 0.012*"beijing" +
0.007*"host" + 0.007*"would" + 0.006*"room" + 0.006*"airport" + 0.006*"also"'),
 (1,
  '0.012*"place" + 0.012*"nice" + 0.011*"room" + 0.010*"host" + 0.010*"also" +
0.008*"beijing" + 0.008*"great" + 0.008*"home" + 0.008*"stay" +
0.007*"airbnb"'),
 (2,
  '0.026*"place" + 0.017*"great" + 0.016*"host" + 0.015*"apartment" +
0.014*"clean" + 0.013*"stay" + 0.011*"recommend" + 0.010*"u" + 0.010*"beijing" +
0.009*"room"'),
 (3,
  '0.031*"great" + 0.020*"place" + 0.018*"stay" + 0.015*"host" + 0.013*"beijing"
+ 0.012*"location" + 0.011*"good" + 0.011*"apartment" + 0.011*"really" +
0.011*"room"'),
 (4,
  '0.019*"place" + 0.018*"great" + 0.016*"host" + 0.013*"stay" + 0.012*"room" +
0.012*"clean" + 0.010*"beijing" + 0.010*"nice" + 0.009*"good" + 0.009*"well"'),
 (5,
  '0.020*"stay" + 0.018*"great" + 0.011*"place" + 0.011*"host" + 0.010*"beijing"
+ 0.009*"nice" + 0.009*"subway" + 0.008*"location" + 0.007*"bed" +
0.007*"time"'),
 (6,
  '0.016*"u" + 0.011*"stay" + 0.009*"nice" + 0.008*"really" + 0.007*"amazing" +
0.006*"time" + 0.006*"great" + 0.006*"need" + 0.006*"host" + 0.006*"place"'),
 (7,
  '0.014*"home" + 0.012*"good" + 0.009*"place" + 0.009*"beijing" + 0.009*"nice"
+ 0.008*"like" + 0.008*"apartment" + 0.008*"stay" + 0.007*"also" +
0.007*"host"')]
```

Considering 8 topics did not give any better results and we can see from the above results the topics are overlapped .

5.0.7 * But revealed one important topic about the check-in and booking processes which should be easy to access.

5.1 Putting it all together - the WordCloud, the Frequency Distribution and the Topic Modelling - it is often the following criteria that make someone rate an apartment positively:

5.1.1 1. The apartment is clean, the bathroom is clean, the bed is comfortable.

5.1.2 2. The area is centrally located with short walking distances, good public transport connections, and has cafes and restaurants nearby.

5.1.3 3. The check- in formalities and the process of booking should be easy.

6 Investigating Negative Comments

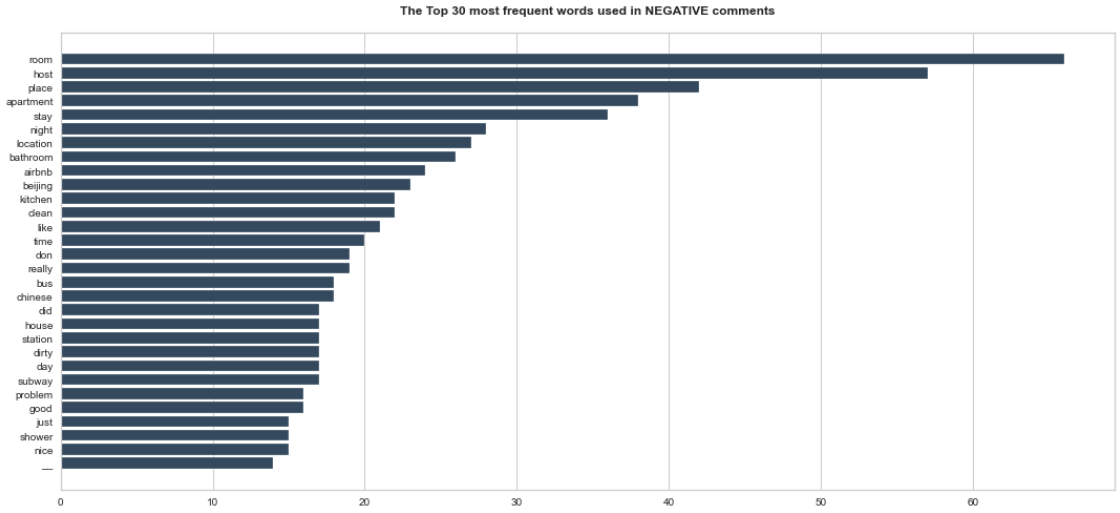
Frequency Distribution

[88]:
```
# vectorizing text
vectorizer = CountVectorizer(stop_words='english')
docs = vectorizer.fit_transform(neg_comments)
features = vectorizer.get_feature_names()

# preparing the plot
set_palette('flatui')
plt.figure(figsize=(18,8))
plt.title('The Top 30 most frequent words used in NEGATIVE comments\n',␣
 ↪fontweight='bold')

# instantiating and fitting the FreqDistVisualizer, plotting the top 30 most␣
 ↪frequent terms
visualizer = FreqDistVisualizer(features=features, n=30)
visualizer.fit(docs)
visualizer.poof;
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```



topic modeling for negetive reviews

[89]:
```
# calling the cleaning function we defined earlier
doc_clean = [clean(comment).split() for comment in neg_comments]
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/Users/utility/opt/anaconda3/lib/python3.8/asyncio/events.py:81:
DeprecationWarning: `run_cell_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  self._context.run(self._callback, *self._args)
```

[90]:
```
# create a dictionary from the normalized data, convert this to a bag-of-words␣
↪corpus
dictionary = corpora.Dictionary(doc_clean)
corpus = [dictionary.doc2bow(text) for text in doc_clean]
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/Users/utility/opt/anaconda3/lib/python3.8/asyncio/events.py:81:
DeprecationWarning: `run_cell_async` will not call `transform_cell`
```

```
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  self._context.run(self._callback, *self._args)
```

[91]:
```
# let LDA find 3 topics
ldamodel_neg = gensim.models.ldamodel.LdaModel(corpus, num_topics=3,␣
 ↪id2word=dictionary, passes=15)
ldamodel_neg.show_topics()
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[91]:
```
[(0,
  '0.012*"host" + 0.011*"apartment" + 0.010*"stay" + 0.008*"beijing" +
0.007*"room" + 0.007*"find" + 0.006*"really" + 0.006*"it" + 0.005*"guest" +
0.005*"problem"'),
 (1,
  '0.015*"room" + 0.006*"airbnb" + 0.006*"place" + 0.005*"also" + 0.005*"go" +
0.005*"house" + 0.005*"time" + 0.005*"one" + 0.005*"night" + 0.004*"minute"'),
 (2,
  '0.011*"room" + 0.011*"host" + 0.010*"place" + 0.010*"get" + 0.007*"u" +
0.007*"clean" + 0.007*"bathroom" + 0.006*"night" + 0.006*"one" +
0.006*"location"')]
```

[92]:
```
# pyLDAvis.enable_notebook()
# pyLDAvis.gensim.prepare(ldamodel_neg, corpus, dictionary)
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-92-f5218a75210e> in <module>
      1 pyLDAvis.enable_notebook()
----> 2 pyLDAvis.gensim.prepare(ldamodel_neg, corpus, dictionary)

AttributeError: module 'pyLDAvis' has no attribute 'gensim'
```

[93]:
```
# let LDA find 5 topics
ldamodel_5_neg = gensim.models.ldamodel.LdaModel(corpus, num_topics=5,␣
↪id2word=dictionary, passes=15)
ldamodel_5_neg.show_topics()
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[93]:
```
[(0,
  '0.014*"host" + 0.011*"night" + 0.011*"room" + 0.009*"apartment" + 0.007*"one"
+ 0.007*"location" + 0.007*"stay" + 0.007*"bathroom" + 0.007*"also" +
0.007*"place"'),
 (1,
  '0.015*"room" + 0.011*"bathroom" + 0.009*"kitchen" + 0.007*"host" +
0.007*"place" + 0.006*"get" + 0.006*"2" + 0.006*"really" + 0.006*"u" +
0.005*"one"'),
 (2,
  '0.018*"room" + 0.008*"time" + 0.008*"host" + 0.006*"hard" + 0.005*"arrived" +
0.005*"house" + 0.005*"nice" + 0.005*"it" + 0.005*"night" + 0.005*"another"'),
 (3,
  '0.014*"place" + 0.010*"find" + 0.008*"airbnb" + 0.007*"stay" + 0.007*"get" +
0.006*"host" + 0.006*"chinese" + 0.006*"hotel" + 0.006*"apartment" +
0.006*"location"'),
 (4,
  '0.011*"host" + 0.011*"bus" + 0.011*"room" + 0.010*"get" + 0.009*"subway" +
0.008*"apartment" + 0.007*"stop" + 0.007*"airbnb" + 0.007*"stay" +
0.006*"also"')]
```

[160]:
```
# pyLDAvis.enable_notebook()
# pyLDAvis.gensim.prepare(ldamodel_5_neg, corpus, dictionary)
```

[160]:
```
PreparedData(topic_coordinates=              x         y  topics  cluster
Freq
topic
1     -0.138280 -0.105563       1        1  43.854332
4     -0.161849  0.005672       2        1  35.905392
2     -0.088154  0.112653       3        1  12.842824
3      0.188662  0.109567       4        1   3.871375
0      0.199622 -0.122330       5        1   3.526079, topic_info=      Category
Freq         Term        Total  loglift  logprob
term
38     Default  1141.000000         host  1141.000000  30.0000  30.0000
19     Default   735.000000          day   735.000000  29.0000  29.0000
```

119	Default	1888.000000	apartment	1888.000000	28.0000	28.0000
91	Default	244.000000	arrival	244.000000	27.0000	27.0000
265	Default	1355.000000	place	1355.000000	26.0000	26.0000
358	Default	385.000000	close	385.000000	25.0000	25.0000
95	Default	581.000000	berlin	581.000000	24.0000	24.0000
203	Default	760.000000	location	760.000000	23.0000	23.0000
1618	Default	107.000000	war	107.000000	22.0000	22.0000
287	Default	633.000000	problem	633.000000	21.0000	21.0000
308	Default	616.000000	airbnb	616.000000	20.0000	20.0000
211	Default	659.000000	nice	659.000000	19.0000	19.0000
357	Default	207.000000	walk	207.000000	18.0000	18.0000
67	Default	1266.000000	room	1266.000000	17.0000	17.0000
729	Default	261.000000	station	261.000000	16.0000	16.0000
254	Default	455.000000	dirty	455.000000	15.0000	15.0000
93	Default	93.000000	reservation	93.000000	14.0000	14.0000
354	Default	165.000000	stop	165.000000	13.0000	13.0000
284	Default	292.000000	minute	292.000000	12.0000	12.0000
13	Default	671.000000	clean	671.000000	11.0000	11.0000
714	Default	75.000000	und	75.000000	10.0000	10.0000
450	Default	170.000000	restaurant	170.000000	9.0000	9.0000
500	Default	220.000000	city	220.000000	8.0000	8.0000
728	Default	160.000000	near	160.000000	7.0000	7.0000
44	Default	459.000000	key	459.000000	6.0000	6.0000
678	Default	69.000000	die	69.000000	5.0000	5.0000
123	Default	208.000000	away	208.000000	4.0000	4.0000
344	Default	123.000000	bus	123.000000	3.0000	3.0000
45	Default	503.000000	kitchen	503.000000	2.0000	2.0000
300	Default	313.000000	everything	313.000000	1.0000	1.0000
…	…	…	…	…	…	…
5321	Topic5	5.703131	pero	6.426805	3.2255	-6.7310
6414	Topic5	5.486557	tatjana	6.210721	3.2210	-6.7697
3933	Topic5	5.115761	leakage	5.862386	3.2088	-6.8397
8763	Topic5	4.730936	tenía	5.454475	3.2027	-6.9179
2013	Topic5	4.655991	bien	5.382220	3.2000	-6.9338
2812	Topic5	4.648293	management	5.378987	3.1990	-6.9355
5313	Topic5	54.051361	la	62.639908	3.1975	-4.4821
6405	Topic5	8.697236	se	10.086078	3.1968	-6.3090
1842	Topic5	4.563251	fabian	5.304588	3.1944	-6.9540
10891	Topic5	4.311907	yehudi	5.038024	3.1893	-7.0106
10000	Topic5	4.310890	tobi	5.037627	3.1892	-7.0108
3462	Topic5	24.559483	el	29.371111	3.1661	-5.2709
3257	Topic5	10.028067	al	12.952691	3.0891	-6.1666
5096	Topic5	7.142548	maria	8.988964	3.1151	-6.5059
93	Topic5	48.685040	reservation	93.407288	2.6934	-4.5866
5181	Topic5	17.340948	un	27.381426	2.8882	-5.6189
681	Topic5	28.561113	e	65.495956	2.5150	-5.1199
91	Topic5	42.420750	arrival	244.635773	1.5929	-4.7244

```
3758    Topic5    12.509000    cancelled    33.282963   2.3664  -5.9455
38      Topic5    62.667362         host  1141.646729   0.4426  -4.3341
19      Topic5    50.328419          day   735.109375   0.6635  -4.5534
674     Topic5    11.539460          con    31.896770   2.3283  -6.0262
2010    Topic5    13.603740       cancel    51.479401   2.0141  -5.8616
113     Topic5    19.329201         time   708.853271  -0.2570  -5.5104
1365    Topic5     9.482661        frank    39.609852   1.9154  -6.2225
308     Topic5    13.983053       airbnb   616.972534  -0.4420  -5.8341
95      Topic5    13.568082       berlin   581.166016  -0.4123  -5.8643
351     Topic5     9.163607           no    52.590614   1.5977  -6.2567
589     Topic5     9.879372         wifi   411.541382  -0.3845  -6.1815
473     Topic5     9.883283         stay   961.195251  -1.2323  -6.1811

[376 rows x 6 columns], token_table=       Topic      Freq            Term
term
4979       3  0.880453            1015
4398       3  0.823737            2030
90         5  0.963363              86
343        2  0.952039      accessible
368        1  0.987435      additional
985        1  0.976996         address
985        2  0.018788         address
2043       1  0.965995      aggressive
308        1  0.896312          airbnb
308        2  0.069695          airbnb
308        3  0.008104          airbnb
308        4  0.003242          airbnb
308        5  0.022691          airbnb
1124       1  0.964626         airbnbs
324        1  0.332167         airport
324        3  0.664334         airport
3257       4  0.154408              al
3257       5  0.772040              al
1439       1  0.019449  alexanderplatz
1439       2  0.213936  alexanderplatz
1439       3  0.758501  alexanderplatz
1439       5  0.019449  alexanderplatz
667        4  0.989813           alles
245        1  0.390822            also
245        2  0.479983            also
245        3  0.106993            also
245        4  0.020804            also
245        5  0.002972            also
1564       1  0.979603           angry
6297       5  0.858999     apartamento
…      …       …              …
1618       4  0.990276             war
```

```
322        3  0.889584        washroom
432        1  0.207136            well
432        2  0.615573            well
432        3  0.140036            well
432        4  0.037926            well
5107       4  0.968167             wie
589        1  0.272148            wifi
589        2  0.622052            wifi
589        3  0.075327            wifi
589        4  0.004860            wifi
589        5  0.024299            wifi
1115       1  0.188830          window
1115       2  0.662755          window
1115       3  0.144399          window
1115       4  0.003703          window
1115       5  0.003703          window
3202       4  0.987576             wir
716        4  0.964850         wohnung
88         1  0.671880           would
88         2  0.203767           would
88         3  0.110144           would
88         4  0.008261           would
88         5  0.005507           would
89         1  0.978243           wrote
10891      5  0.793962          yehudi
3357       3  0.903549         younger
1778       4  0.895497          zimmer
1072       3  0.916209             zoo
719        4  0.909343              zu

[728 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1',
'ylab': 'PC2'}, topic_order=[2, 5, 3, 4, 1])
```

[94]:
```
# let LDA find 8 topics
ldamodel_8_neg = gensim.models.ldamodel.LdaModel(corpus, num_topics=8,␣
↪id2word=dictionary, passes=15)
ldamodel_8_neg.show_topics()
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[94]:
```
[(0,
  '0.020*"room" + 0.010*"host" + 0.008*"airbnb" + 0.008*"area" + 0.008*"stay" +
```

```
0.007*"beijing" + 0.006*"all" + 0.006*"university" + 0.006*"around" +
0.006*"look"'),
 (1,
  '0.013*"apartment" + 0.012*"host" + 0.010*"u" + 0.008*"shower" +
0.007*"always" + 0.007*"bathroom" + 0.006*"dirty" + 0.006*"beijing" +
0.005*"floor" + 0.005*"well"'),
 (2,
  '0.012*"place" + 0.010*"stay" + 0.009*" " + 0.009*"host" + 0.007*"  " +
0.007*"problem" + 0.007*"kitchen" + 0.005*"chinese" + 0.005*"apartment" +
0.005*"clean"'),
 (3,
  '0.015*"place" + 0.012*"station" + 0.012*"room" + 0.011*"location" +
0.010*"like" + 0.009*"nice" + 0.009*"apartment" + 0.009*"dirty" + 0.008*"subway"
+ 0.007*"time"'),
 (4,
  '0.013*"bus" + 0.013*"get" + 0.013*"room" + 0.012*"place" + 0.010*"bathroom" +
0.009*"find" + 0.008*"go" + 0.007*"one" + 0.006*"could" + 0.006*"take"'),
 (5,
  '0.022*"host" + 0.013*"stay" + 0.011*"room" + 0.010*"apartment" +
0.010*"night" + 0.009*"get" + 0.008*"also" + 0.007*"airbnb" + 0.006*"guest" +
0.006*"house"'),
 (6,
  '0.014*"room" + 0.010*"location" + 0.010*"u" + 0.010*"avoid" + 0.008*"night" +
0.008*"place" + 0.008*"soap" + 0.008*"find" + 0.006*"host" + 0.006*"one"'),
 (7,
  '0.022*"room" + 0.009*"night" + 0.009*"day" + 0.007*"chinese" + 0.007*"leave"
+ 0.007*"want" + 0.007*"bed" + 0.006*"airport" + 0.006*"speak" + 0.006*"one"')]
```

[95]:
```
pyLDAvis.enable_notebook()
pyLDAvis.gensim.prepare(ldamodel_8_neg, corpus, dictionary)
```

```
/Users/utility/opt/anaconda3/lib/python3.8/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-95-65f73a837819> in <module>
      1 pyLDAvis.enable_notebook()
----> 2 pyLDAvis.gensim.prepare(ldamodel_8_neg, corpus, dictionary)

AttributeError: module 'pyLDAvis' has no attribute 'gensim'
```

6.0.1 We can see lot of positive words here which can be caused because of not taking the Bi-grams.

6.0.2 Ex: Not Good place is vecorized into Not , good , place. So good and place appeared many times in the topics

6.1 Some of the topics we observed

6.1.1 1. Dirty Bathrooms, Bad towels and bad kitchen.

6.1.2 2. The Location or the appartment is too noisy.

6.1.3 3. Not enough space and the damaged items, not functioning ammenites in the room.

6.1.4 4. Provided only one key for the flat.

6.1.5 5. The area is centrally not located with short walking distances, good public transport connections.

7 CONCLUSION

7.0.1 Putting it all together - the WordCloud, the Frequency Distribution and the Topic Modelling for both the positve and negetive reviews. A new host who is planning to rent his place to Airbnb need to make sure he provide all the topics we extracted from TOPIC MODELING.

Check out this https://github.com/adityajill/Predicting-the-Price-for-An-Airbnb-Host-in-Berlin-

For the new host who is going to rent his place to AIRBNB, and what price should he be expect from the facilites and amminites he has.

[ ]: