[1]

```
import pandas as pd
```

First read the data into dataframe. Encoding needs to be paid attention to.

[2]

```
df = pd.read_csv('Seattle_Hotels.csv', encoding='latin-1')
df.head()
```

	name	address	desc
0	Hilton Garden Seattle Downtown	1821 Boren Avenue, Seattle Washington 98101 USA	Located on the southern tip of Lake Union, the...
1	Sheraton Grand Seattle	1400 6th Avenue, Seattle, Washington 98101 USA	Located in the city's vibrant core, the Sherat...
2	Crowne Plaza Seattle Downtown	1113 6th Ave, Seattle, WA 98101	Located in the heart of downtown Seattle, the ...
3	Kimpton Hotel Monaco Seattle	1101 4th Ave, Seattle, WA98101	What?s near our hotel downtown Seattle locatio...
4	The Westin Seattle	1900 5th Avenue, Seattle, Washington 98101 USA	Situated amid incredible shopping and iconic a...

[19]

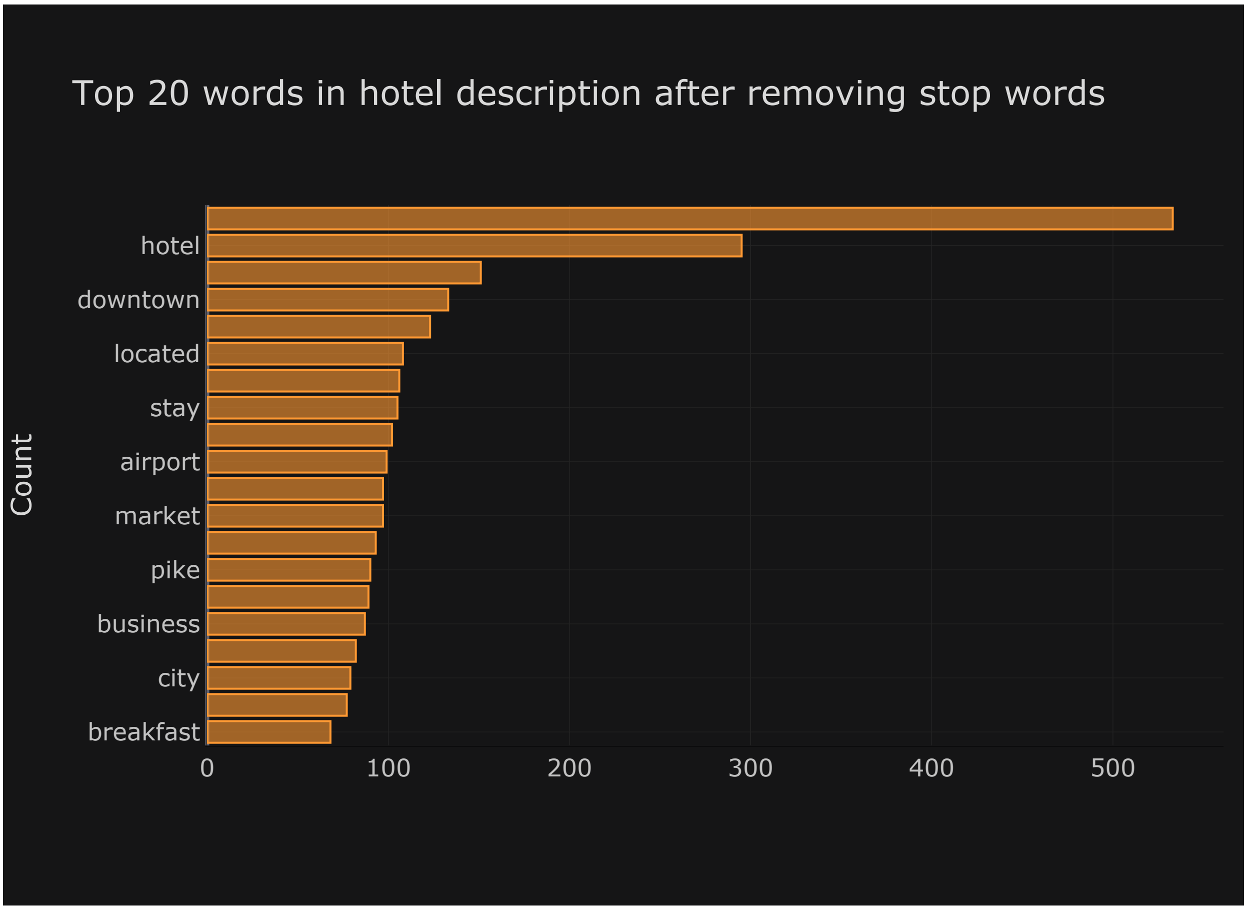
```
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from nltk.corpus import stopwords
from IPython.core.interactiveshell import InteractiveShell
import plotly.figure_factory as ff
InteractiveShell.ast_node_interactivity = 'all'
from plotly.offline import iplot
import cufflinks
```

```
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='solar')
```

Token frequency distribution after removing stop words

[20]

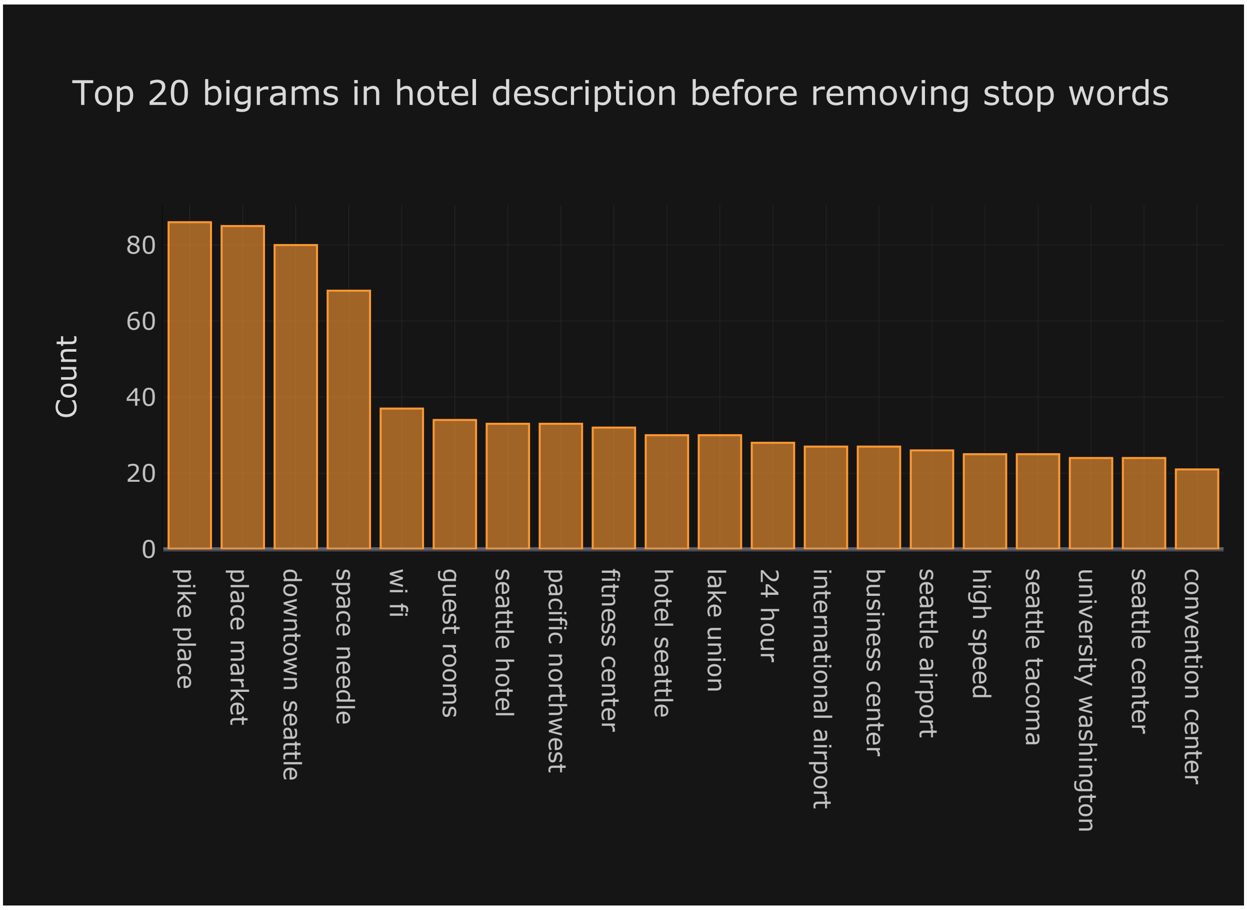
```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key=lambda x: x[1],
reverse=True)
    return words_freq[:n]

common_words = get_top_n_words(df['desc'], 20)
df1 = pd.DataFrame(common_words, columns=['desc', 'count'])
df1.groupby('desc').sum()
['count'].sort_values().iplot(kind='barh', yTitle='Count',
linecolor='black', title='Top 20 words in hotel description after
removing stop words')
```



Bigram frequency distribution after removing stop words

[22]

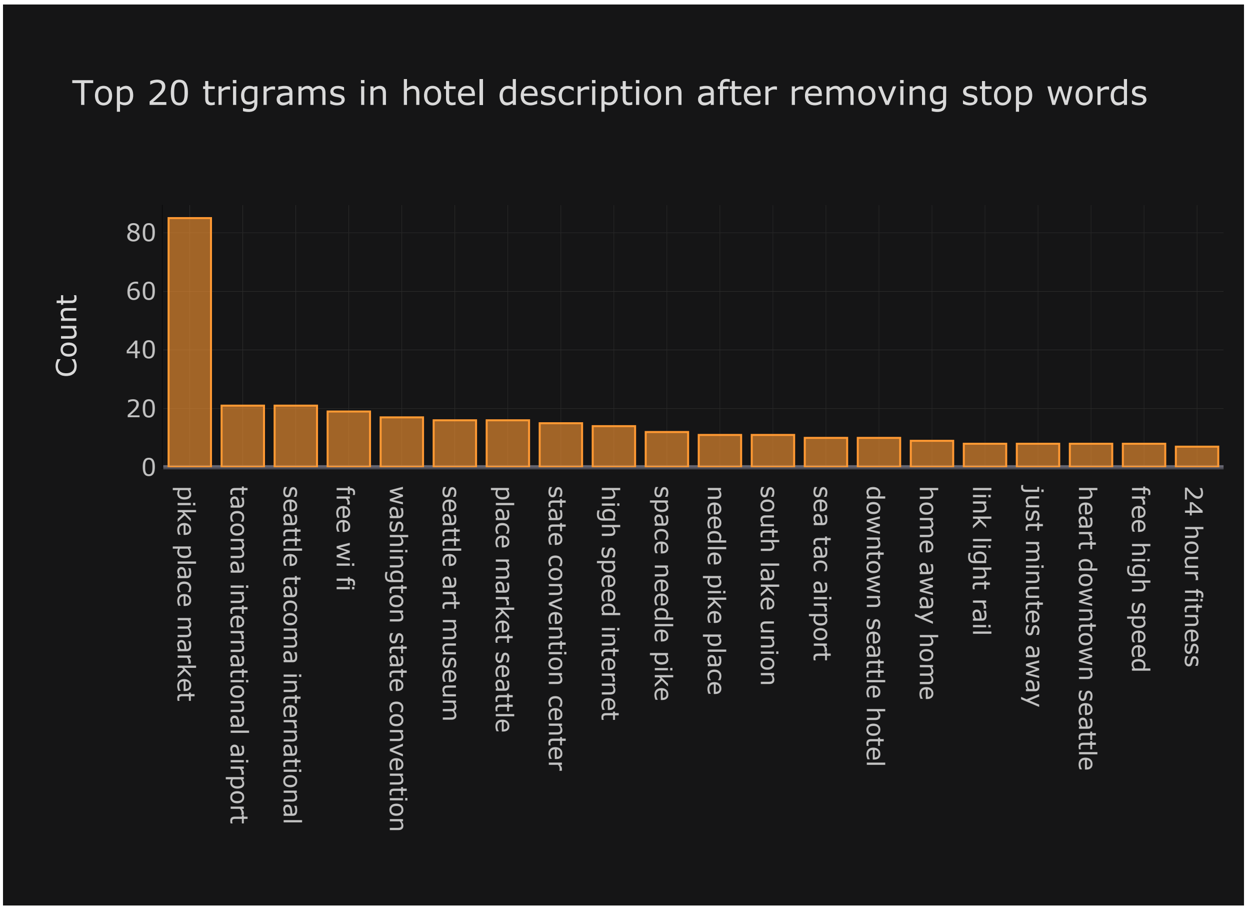
```python
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
common_words = get_top_n_bigram(df['desc'], 20)
df2 = pd.DataFrame(common_words, columns = ['desc' , 'count'])
df2.groupby('desc').sum()['count'].sort_values(ascending=False).iplot(kind='bar', yTitle='Count', linecolor='black', title='Top 20 bigrams in hotel description before removing stop words')
```



Trigrams frequency distribution after removing stop words

[23]

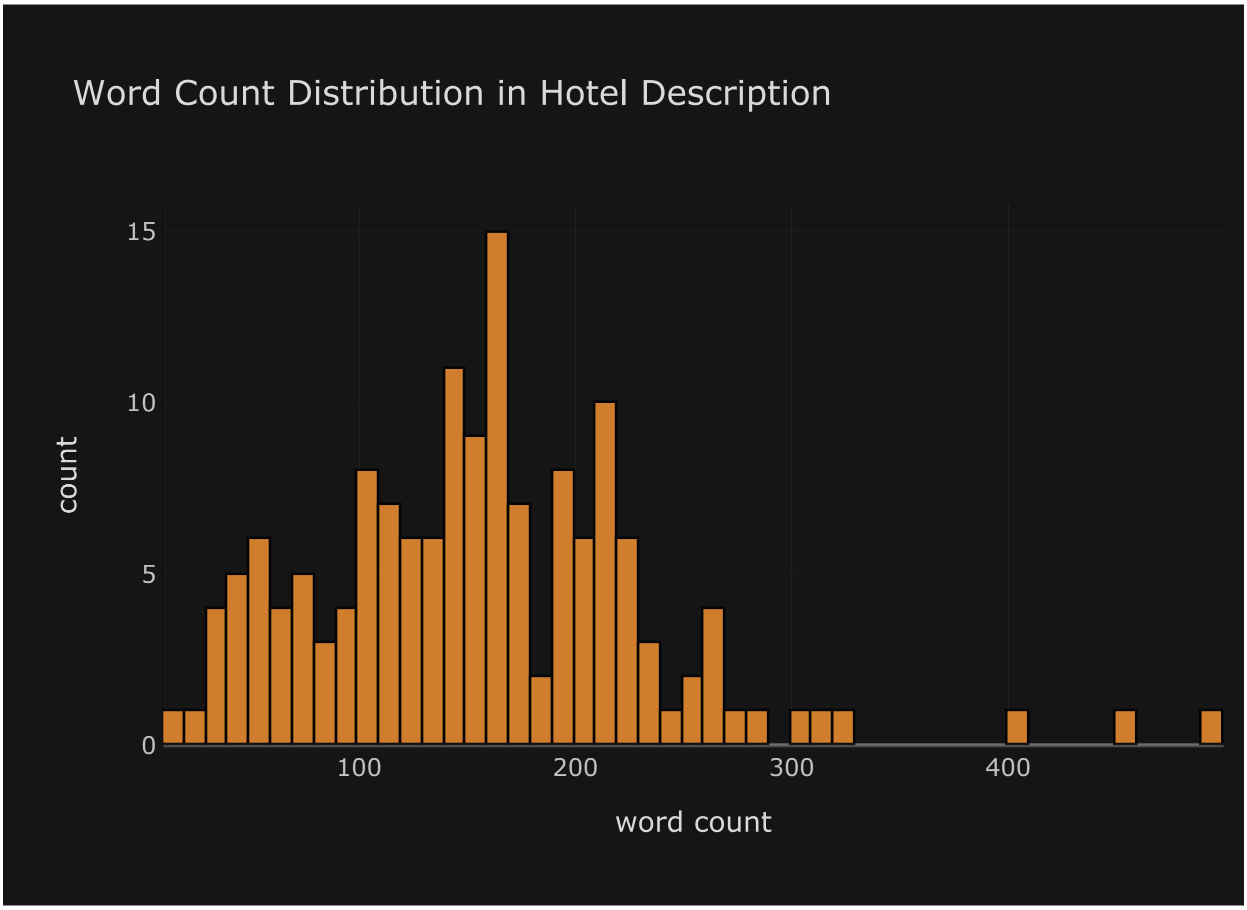
```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3),
stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],
reverse=True)
    return words_freq[:n]
common_words = get_top_n_trigram(df['desc'], 20)
df3 = pd.DataFrame(common_words, columns = ['desc' , 'count'])
df3.groupby('desc').sum()
['count'].sort_values(ascending=False).iplot(kind='bar',
yTitle='Count', linecolor='black', title='Top 20 trigrams in
hotel description after removing stop words')
```



Hotel description word count distribution

[24]

```
df['word_count'] = df['desc'].apply(lambda x:
len(str(x).split()))
df['word_count'].iplot(
    kind='hist',
    bins = 50,
    linecolor='black',
    xTitle='word count',
    yTitle='count',
    title='Word Count Distribution in Hotel Description')
```



Text Preprocessing

[28]

```
import re

REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))

def clean_text(text):
    """
        text: a string

        return: modified initial string
    """
    # lowercase text
    text = text.lower()
    # replace REPLACE_BY_SPACE_RE symbols by space in text.
substitute the matched string in REPLACE_BY_SPACE_RE with space.
    text = REPLACE_BY_SPACE_RE.sub(' ', text)
    # remove symbols which are in BAD_SYMBOLS_RE from text.
substitute the matched string in BAD_SYMBOLS_RE with nothing.
    text = BAD_SYMBOLS_RE.sub('', text)
    # remove stopwors from text
    text = ' '.join(word for word in text.split() if word not in
STOPWORDS)
    return text

df['desc_clean'] = df['desc'].apply(clean_text)
```

Modeling

- Create a TF-IDF matrix of unigrams, bigrams, and trigrams for each hotel.
- Compute similarity between all hotels using sklearn’s linear_kernel (equivalent to cosine similarity in our case).
- Define a function that takes in hotel name as input and returns the top 10 recommended hotels.

[29]

```
from sklearn.metrics.pairwise import linear_kernel

df.set_index('name', inplace = True)
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3),
min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(df['desc_clean'])
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

indices = pd.Series(df.index)

def recommendations(name, cosine_similarities =
cosine_similarities):

    recommended_hotels = []

    # gettin the index of the hotel that matches the name
    idx = indices[indices == name].index[0]

    # creating a Series with the similarity scores in descending
order
    score_series =
pd.Series(cosine_similarities[idx]).sort_values(ascending =
False)
```

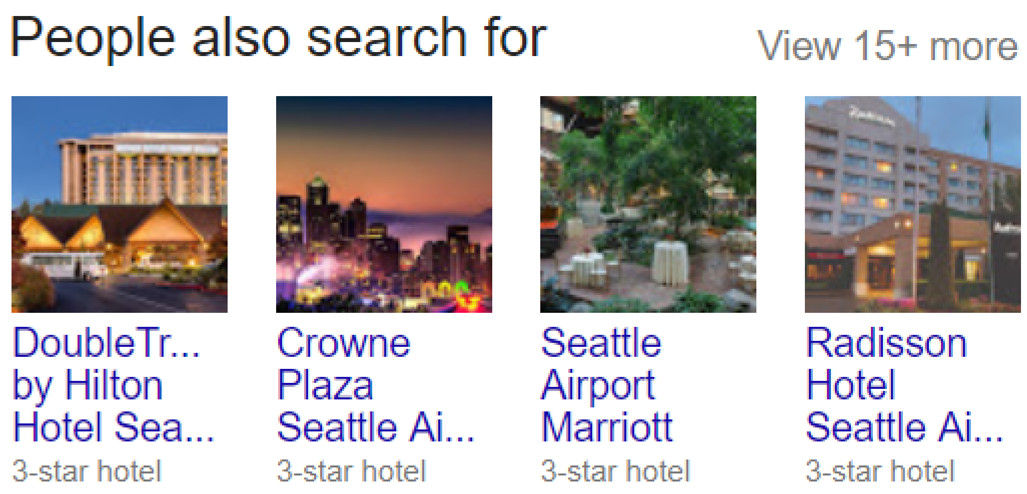
```
    # getting the indexes of the 10 most similar hotels except itself
    top_10_indexes = list(score_series.iloc[1:11].index)

    # populating the list with the names of the top 10 matching hotels
    for i in top_10_indexes:
        recommended_hotels.append(list(df.index)[i])

    return recommended_hotels
```

Recommendations

```
[30]  recommendations('Hilton Seattle Airport & Conference Center')
```

```
['Embassy Suites by Hilton Seattle Tacoma International Airport',
 'DoubleTree by Hilton Hotel Seattle Airport',
 'Seattle Airport Marriott',
 'Motel 6 Seattle Sea-Tac Airport South',
 'Econo Lodge SeaTac Airport North',
 'Four Points by Sheraton Downtown Seattle Center',
 'Knights Inn Tukwila',
 'Econo Lodge Renton-Bellevue',
 'Hampton Inn Seattle/Southcenter',
 'Radisson Hotel Seattle Airport']
```

Trip Advisor Recommendation Results



People also search for

View 15+ more

DoubleTr... by Hilton Hotel Sea...
3-star hotel

Crowne Plaza Seattle Ai...
3-star hotel

Seattle Airport Marriott
3-star hotel

Radisson Hotel Seattle Ai...
3-star hotel