01_EDA_Yelp_Asian_Restaurants

November 17, 2024

Exploratory Data Analysis of Yelp Reviews for Asian Restaurants in the United States

This notebook performs an exploratory data analysis (EDA) of Yelp reviews for Asian restaurants across the United States. The goal is to understand customer perceptions of food quality and authenticity.

1. Introduction

In this notebook, we perform an exploratory data analysis of Yelp reviews to uncover insights about customer perceptions of Asian restaurants in major United States. We focus on:

- Loading and preprocessing the data
- Filtering for Asian restaurants in United States
- Analyzing ratings, review lengths, and sentiments
- Identifying common words and themes
- Exploring geographical patterns

2. Import Libraries

We start by importing the necessary libraries for data manipulation, visualization, and natural language processing.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from nltk.corpus import stopwords
from nltk import word_tokenize, download
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from collections import Counter
from wordcloud import WordCloud

# For NLP preprocessing
#download('stopwords')
#download('punkt')
#download('vader_lexicon')
```

```
# Set visualization style
sns.set(style='whitegrid')
```

3. Load and Preprocess Business Data

3.1 Load the Business Data

We load the business dataset from the Yelp dataset.

[2]:
```
# Load the business data
business = pd.read_json('yelp_academic_dataset_business.json', lines=True)

# Display the first few rows
print("Business Data:")
print(business.head())
```

```
Business Data:
              business_id                      name  \
0  Pns2l4eNsfO8kk83dixA6A  Abby Rappoport, LAC, CMQ
1  mpf3x-BjTdTEA3yCZrAYPw             The UPS Store
2  tUFrWirKiKi_TAnsVWINQQ                    Target
3  MTSW4McQd7CbVtyjqoe9mw        St Honore Pastries
4  mWMc6_wTdE0EUBKIGXDVfA  Perkiomen Valley Brewery

                          address           city state postal_code  \
0           1616 Chapala St, Ste 2  Santa Barbara    CA       93101
1  87 Grasso Plaza Shopping Center         Affton    MO       63123
2             5255 E Broadway Blvd         Tucson    AZ       85711
3                     935 Race St   Philadelphia    PA       19107
4                   101 Walnut St     Green Lane    PA       18054

    latitude   longitude  stars  review_count  is_open  \
0  34.426679 -119.711197    5.0             7        0
1  38.551126  -90.335695    3.0            15        1
2  32.223236 -110.880452    3.5            22        0
3  39.955505  -75.155564    4.0            80        1
4  40.338183  -75.471659    4.5            13        1

                                      attributes  \
0                    {'ByAppointmentOnly': 'True'}
1            {'BusinessAcceptsCreditCards': 'True'}
2  {'BikeParking': 'True', 'BusinessAcceptsCredit…
3  {'RestaurantsDelivery': 'False', 'OutdoorSeati…
4  {'BusinessAcceptsCreditCards': 'True', 'Wheelc…

                                      categories  \
0  Doctors, Traditional Chinese Medicine, Naturop…
1  Shipping Centers, Local Services, Notaries, Ma…
2  Department Stores, Shopping, Fashion, Home & G…
```

```
3  Restaurants, Food, Bubble Tea, Coffee & Tea, B…
4                          Brewpubs, Breweries, Food

                                                hours
0                                                None
1  {'Monday': '0:0-0:0', 'Tuesday': '8:0-18:30', …
2  {'Monday': '8:0-22:0', 'Tuesday': '8:0-22:0', …
3  {'Monday': '7:0-20:0', 'Tuesday': '7:0-20:0', …
4  {'Wednesday': '14:0-22:0', 'Thursday': '16:0-2…
```

3.2 Explore Business Data

Let's check the columns and the number of businesses.

[3]:
```
print("\nBusiness Data Columns:")
print(business.columns)

print("\nNumber of Businesses:")
print(len(business))
```

```
Business Data Columns:
Index(['business_id', 'name', 'address', 'city', 'state', 'postal_code',
       'latitude', 'longitude', 'stars', 'review_count', 'is_open',
       'attributes', 'categories', 'hours'],
      dtype='object')

Number of Businesses:
150346
```

4. Filter for Asian Restaurants in Major US Cities

4.1 Filter Businesses in the United States

We filter the businesses to include only those located in United States.

[4]:
```
# List of US states (abbreviations)
us_states = [
    'AL', 'AK', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DE', 'FL', 'GA', 'HI', 'ID',␣
 ↪'IL', 'IN', 'IA',
    'KS', 'KY', 'LA', 'ME', 'MD', 'MA', 'MI', 'MN', 'MS', 'MO', 'MT', 'NE',␣
 ↪'NV', 'NH', 'NJ',
    'NM', 'NY', 'NC', 'ND', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN',␣
 ↪'TX', 'UT', 'VT',
    'VA', 'WA', 'WV', 'WI', 'WY'
]

# Filter businesses in the US
business_us = business[business['state'].isin(us_states)]
```

```
# Filter for open businesses
business_us_open = business_us[business_us['is_open'] == 1]

# Filter for businesses with at least 5 reviews
business_us_open_reviews = business_us_open[business_us_open['review_count'] >=␣
 ↪5]

print(f"Number of open businesses in the US with at least 5 reviews:␣
 ↪{len(business_us_open_reviews)}")


print(f"\nNumber of businesses in the US: {len(business_us)}")
```

```
Number of open businesses in the US with at least 5 reviews: 115351

Number of businesses in the US: 144771
```

4.2 Filter for Asian Restaurants

We identify Asian restaurants by checking if their categories include Asian cuisine keywords.

[5]:
```
# Define a function to check if a category is Asian cuisine
def is_asian_cuisine(categories):
    if categories:
        categories_lower = categories.lower()
        asian_keywords = [
            'asian', 'chinese', 'japanese', 'korean', 'thai', 'vietnamese',␣
 ↪'indian',
            'malaysian', 'filipino', 'indonesian', 'sushi', 'ramen', 'dim sum',
            'pakistani', 'bangladeshi', 'singaporean', 'mongolian', 'cambodian',
            'laotian', 'himalayan', 'nepalese', 'burmese', 'tibetan'
        ]
        return any(keyword in categories_lower for keyword in asian_keywords)
    else:
        return False

# Apply the function to filter Asian restaurants
business_us_open_reviews.loc[:,'is_asian'] =␣
 ↪business_us_open_reviews['categories'].apply(is_asian_cuisine)

asian_restaurants =␣
 ↪business_us_open_reviews[business_us_open_reviews['is_asian'] == True]

print(f"Number of Asian restaurants in the US: {len(asian_restaurants)}")
```

```
Number of Asian restaurants in the US: 4914
```

4.3 Get List of Relevant Business IDs

We extract the unique business IDs for the Asian restaurants.

[6]:
```
# Get unique business IDs for Asian restaurants
asian_business_ids = asian_restaurants['business_id'].unique()
```

5. Load and Filter Review Data in Chunks

Due to the large size of the review dataset, we read it in chunks and filter reviews related to the Asian restaurants we've identified.

[7]:
```
# Initialize an empty list to store filtered reviews
filtered_reviews_list = []

# Specify the chunk size (adjust as necessary)
chunk_size = 100000  # 100,000 rows at a time

# Read the review data in chunks
reviews_chunks = pd.read_json('yelp_academic_dataset_review.json', lines=True,␣
 ↪chunksize=chunk_size)

# Counter to keep track of chunks processed
chunk_count = 0

# Loop through the chunks and filter for relevant business IDs
for chunk in reviews_chunks:
    chunk_count += 1
    #print(f"Processing chunk {chunk_count}...")
    # Filter reviews for the selected business IDs
    filtered_chunk = chunk[chunk['business_id'].isin(asian_business_ids)]
    # Append filtered chunk to the list
    filtered_reviews_list.append(filtered_chunk)

# Combine all filtered review chunks into a single DataFrame
reviews_asian = pd.concat(filtered_reviews_list, ignore_index=True)

print(f"\nNumber of reviews for Asian restaurants: {len(reviews_asian)}")
```

```
Number of reviews for Asian restaurants: 591773
```

6. Merge Business and Review Data

We merge the filtered reviews with the business data to have all relevant information in one DataFrame.

[8]:
```
```

```
# Merge business data with reviews
df = pd.merge(reviews_asian, asian_restaurants, on='business_id', how='left',␣
 ↪suffixes=('_review', '_business'))

# Display the merged DataFrame
print("\nMerged DataFrame:")
print(df.head())
```

```
Merged DataFrame:
                review_id                 user_id             business_id  \
0  AqPFM1eE6RsU23_auESxiA  _7bHUi9Uuf5__HHc_Q8guQ  kxX2SOes4o-D3ZQBkiMRfA
1  pUycOfUwM8vqX7KjRRhUEA  59MxRhNVhU9MYndMkz0wtw  gebiRewfieSdtt17PTW6Zg
2  eCiWBf1CJ0Zdv1uVarEhhw  OhECKhQEexFypOMY6kypRw  vC2qm1y3Au5czBtbhc-DNw
3  G_5UczbCBJriUAbxz3J7Tw  clWLI5OZP2ad25ugMVI8gg  x4XdNhp0Xn8lOivzc77J-g
4  r2IBPY_E8AE5_GpsqlONyg  IKbjLnfBQtEyVzEu8CuOLg  VJEzpfLs_Jnzgqh5A_FVTg

   stars_review  useful  funny  cool  \
0             5       1      0     1
1             3       0      0     0
2             4       0      0     0
3             5       0      0     0
4             4       0      0     0

                                                text                 date  \
0  Wow!  Yummy, different,  delicious.   Our favo… 2015-01-04 00:01:03
1  Had a party of 6 here for hibachi. Our waitres… 2016-07-25 07:31:06
2  Yes, this is the only sushi place in town. How… 2013-09-04 03:48:20
3  Best thai food in the area.  Everything was au… 2013-08-15 15:27:51
4  It was my fiance's birthday and he decided he … 2014-04-01 13:05:18

                           name  … postal_code   latitude   longitude  \
0                         Zaika  …       19114  40.079848  -75.025080
1  Hibachi Steak House & Sushi Bar  …       93101  34.416984 -119.695556
2                     Sushi Teri  …       93013  34.398527 -119.518475
3           Thai Place Restaurant  …       19460  40.132635  -75.533226
4      Jasmine Rice - Rittenhouse  …       19102  39.947084  -75.168205

  stars_business  review_count  is_open  \
0            4.0           181        1
1            3.5           488        1
2            3.0           167        1
3            4.5           222        1
4            3.5           307        1

                                        attributes  \
0  {'Caters': 'True', 'Ambience': '{'romantic': F…
1  {'Corkage': 'False', 'RestaurantsTakeOut': 'Tr…
```

```
2  {'RestaurantsReservations': 'True', 'NoiseLeve…
3  {'OutdoorSeating': 'False', 'RestaurantsDelive…
4  {'RestaurantsPriceRange2': '2', 'RestaurantsAt…

                                       categories  \
0           Halal, Pakistani, Restaurants, Indian
1  Steakhouses, Sushi Bars, Restaurants, Japanese
2                          Restaurants, Sushi Bars
3                               Thai, Restaurants
4                  Soup, Thai, Restaurants, Salad

                                              hours is_asian
0  {'Tuesday': '11:0-21:0', 'Wednesday': '11:0-21…     True
1                              {'Monday': '0:0-0:0'}     True
2  {'Monday': '17:0-22:0', 'Tuesday': '17:0-22:0'…     True
3  {'Tuesday': '17:0-21:30', 'Wednesday': '17:0-2…     True
4  {'Monday': '13:30-22:30', 'Tuesday': '13:30-22…     True

[5 rows x 23 columns]
```

7. Exploratory Data Analysis (EDA)

Now, we proceed with the EDA using the merged DataFrame.

7.1 Descriptive Statistics

7.1.1 Basic Information

We examine the basic structure and summary statistics of the DataFrame.

[9]:
```
print("\nDataFrame Information:")
df.info()

print("\nSummary Statistics:")
print(df.describe())
```

```
DataFrame Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 591773 entries, 0 to 591772
Data columns (total 23 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   review_id       591773 non-null  object
 1   user_id         591773 non-null  object
 2   business_id     591773 non-null  object
 3   stars_review    591773 non-null  int64
 4   useful          591773 non-null  int64
 5   funny           591773 non-null  int64
```

```
 6   cool            591773 non-null  int64
 7   text            591773 non-null  object
 8   date            591773 non-null  datetime64[ns]
 9   name            591773 non-null  object
 10  address         591773 non-null  object
 11  city            591773 non-null  object
 12  state           591773 non-null  object
 13  postal_code     591773 non-null  object
 14  latitude        591773 non-null  float64
 15  longitude       591773 non-null  float64
 16  stars_business  591773 non-null  float64
 17  review_count    591773 non-null  int64
 18  is_open         591773 non-null  int64
 19  attributes      591272 non-null  object
 20  categories      591773 non-null  object
 21  hours           578899 non-null  object
 22  is_asian        591773 non-null  bool
dtypes: bool(1), datetime64[ns](1), float64(3), int64(6), object(12)
memory usage: 99.9+ MB

Summary Statistics:
        stars_review         useful          funny           cool  \
count  591773.000000  591773.000000  591773.000000  591773.000000
mean        3.904783       1.031360       0.291076       0.503386
min         1.000000       0.000000       0.000000       0.000000
25%         3.000000       0.000000       0.000000       0.000000
50%         4.000000       0.000000       0.000000       0.000000
75%         5.000000       1.000000       0.000000       0.000000
max         5.000000     191.000000     129.000000     195.000000
std         1.341699       2.550442       1.296022       1.956285

                             date       latitude      longitude  \
count                      591773  591773.000000  591773.000000
mean   2017-05-09 19:17:12.719816704      36.294415     -88.484950
min              2005-03-01 19:33:35      27.675727    -119.916058
25%              2015-06-15 20:03:08      32.222806     -90.271490
50%              2017-10-17 22:46:17      39.487483     -82.751421
75%              2019-08-30 20:13:48      39.951577     -75.201654
max              2022-01-19 19:48:13      43.725446     -74.685404
std                              NaN       4.861742      15.254925

       stars_business   review_count   is_open
count   591773.000000  591773.000000  591773.0
mean         3.898092     413.889273       1.0
min          1.000000       5.000000       1.0
25%          3.500000     120.000000       1.0
50%          4.000000     231.000000       1.0
75%          4.500000     437.000000       1.0
```

```
max          5.000000    5721.000000       1.0
std          0.519095     684.196594       0.0
```

7.1.2 Number of Restaurants and Reviews

[10]:
```
num_reviews = df.shape[0]
num_restaurants = df['business_id'].nunique()

print(f"\nTotal number of reviews: {num_reviews}")
print(f"Total number of Asian restaurants in the US: {num_restaurants}")
```

```
Total number of reviews: 591773
Total number of Asian restaurants in the US: 4914
```

7.2 Ratings Distribution

7.2.1 Calculate Rating Statistics

We calculate the average, median, and standard deviation of the star ratings.

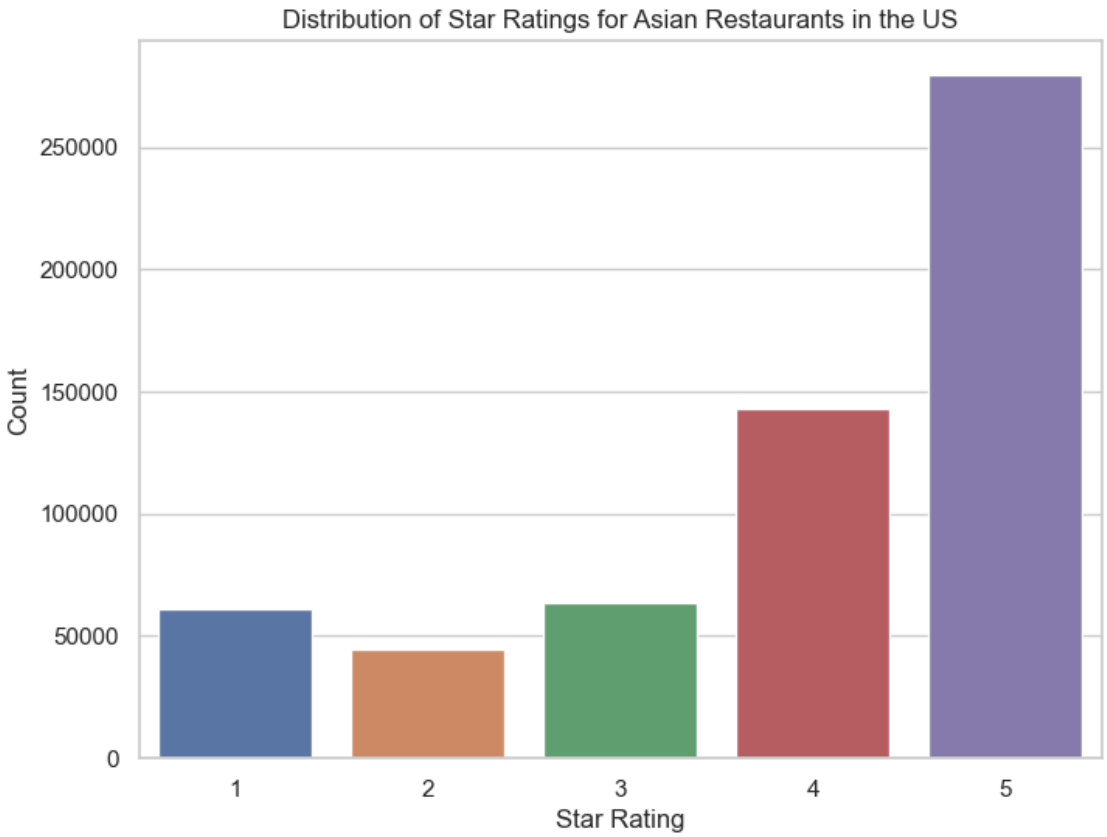
[11]:
```
average_rating = df['stars_review'].mean()
median_rating = df['stars_review'].median()
rating_std = df['stars_review'].std()

print(f"\nAverage Rating: {average_rating:.2f}")
print(f"Median Rating: {median_rating}")
print(f"Standard Deviation of Ratings: {rating_std:.2f}")
```

```
Average Rating: 3.90
Median Rating: 4.0
Standard Deviation of Ratings: 1.34
```

7.2.2 Plot Ratings Histogram

We visualize the distribution of star ratings.

[12]:
```
plt.figure(figsize=(8,6))
sns.countplot(x='stars_review', data=df, order=sorted(df['stars_review'].
  ↪unique()))
plt.title('Distribution of Star Ratings for Asian Restaurants in the US')
plt.xlabel('Star Rating')
plt.ylabel('Count')
plt.show()
```



7.3 Review Length Analysis

7.3.1 Calculate Review Lengths

We compute the length of each review in words.

[13]:
```
# Calculate the length of each review
df['review_length'] = df['text'].apply(lambda x: len(x.split()))

# Summary statistics of review lengths
average_length = df['review_length'].mean()
median_length = df['review_length'].median()
std_length = df['review_length'].std()

print(f"\nAverage Review Length (in words): {average_length:.2f}")
print(f"Median Review Length (in words): {median_length}")
print(f"Standard Deviation of Review Length: {std_length:.2f}")
```

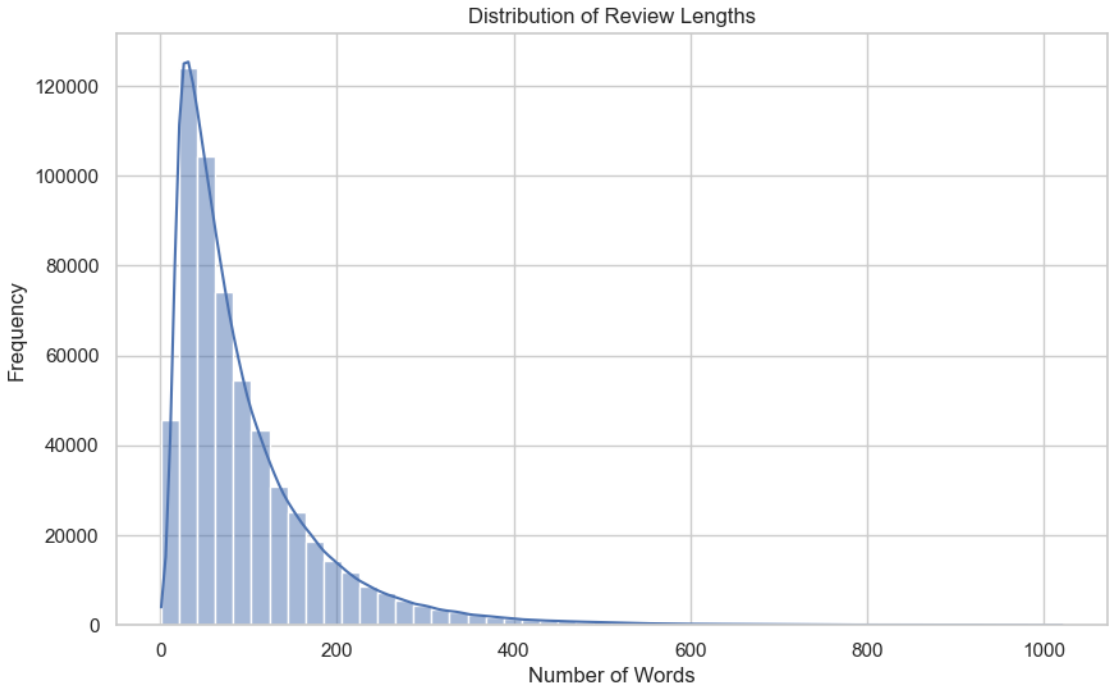
```
Average Review Length (in words): 95.65
Median Review Length (in words): 68.0
Standard Deviation of Review Length: 88.68
```

7.3.2 Plot Review Length Distribution

We plot the distribution of review lengths.

[14]:
```
plt.figure(figsize=(10,6))
sns.histplot(df['review_length'], bins=50, kde=True)
plt.title('Distribution of Review Lengths')
plt.xlabel('Number of Words')
plt.ylabel('Frequency')
plt.show()
```

```
/Users/esteffrivero/anaconda3/lib/python3.11/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



7.4 Most Common Words and Bigrams

7.4.1 Preprocess Text

We preprocess the review text by lowercasing, removing punctuation, and stopwords.

[15]:
```
# Define standard stop words
stop_words = set(stopwords.words('english'))
```

```
# Add restaurant-specific stop words
restaurant_stop_words = [
    'restaurant', 'food', 'place', 'menu', 'order', 'ordered', 'eat', 'ate',
    'dish', 'dishes', 'meal', 'meals', 'server', 'servers', 'waiter',␣
 ↪'waitress',
    'staff', 'table', 'tables', 'seat', 'seated', 'drinks', 'drink',␣
 ↪'appetizer',
    'appetizers', 'main', 'course', 'courses', 'dessert', 'desserts', 'lunch',
    'dinner', 'breakfast', 'brunch', 'location', 'area', 'experience',␣
 ↪'experiences',
    'customer', 'customers', 'service', 'services', 'takeout', 'take', 'out',␣
 ↪'time',
    'times', 'visit', 'visits', 'visited', 'day', 'days', 'night', 'nights',␣
 ↪'people',
    'person', 'group', 'groups', 'friend', 'friends', 'family'
]

# Combine standard and custom stop words
stop_words.update(restaurant_stop_words)

# Function to preprocess text
def preprocess_text(text):
    # Lowercase
    text = text.lower()
    # Remove punctuation and numbers
    text = re.sub(r'[^a-z\s]', '', text)
    # Tokenize
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    return tokens

# Apply preprocessing
df['tokens'] = df['text'].apply(preprocess_text)
```

7.4.2 Most Common Words

We identify the top 20 most frequent words in the reviews.

[16]:
```
# Flatten list of tokens
all_words = [word for tokens in df['tokens'] for word in tokens]
word_counts = Counter(all_words)
common_words = word_counts.most_common(20)

print("\nTop 20 Most Common Words After Removing Custom Stop Words:")
for word, count in common_words:
    print(f"{word}: {count}")
```

```
Top 20 Most Common Words After Removing Custom Stop Words:
good: 344458
great: 242690
sushi: 208503
like: 186643
chicken: 176135
one: 156422
back: 152845
get: 152598
really: 145011
go: 144126
would: 137053
also: 136851
delicious: 125883
rice: 121904
best: 120383
got: 111625
ive: 110160
try: 104911
us: 103053
always: 101142
```

7.4.3 Most Common Bigrams

We identify the top 20 most frequent bigrams (two-word phrases).

[17]:
```python
from nltk.util import ngrams

# Function to get bigrams
def get_bigrams(tokens_list):
    bigrams_list = []
    for tokens in tokens_list:
        bigrams_list.extend(list(ngrams(tokens, 2)))
    return bigrams_list

# Get bigrams
all_bigrams = get_bigrams(df['tokens'])
bigram_counts = Counter(all_bigrams)
common_bigrams = bigram_counts.most_common(20)

print("\nTop 20 Most Common Bigrams After Removing Custom Stop Words:")
for bigram, count in common_bigrams:
    print(f"{' '.join(bigram)}: {count}")
```

```
Top 20 Most Common Bigrams After Removing Custom Stop Words:
fried rice: 36900
```

```
pad thai: 26317
go back: 24783
really good: 24099
come back: 23133
highly recommend: 22304
spring rolls: 19297
pretty good: 19270
ive ever: 17056
happy hour: 13266
dim sum: 12870
coming back: 12168
one best: 11255
cant wait: 10843
would recommend: 10791
ice cream: 10753
egg rolls: 10598
definitely back: 10589
dont know: 10450
even though: 10313
```

7.5 Sentiment Analysis with VADER

We use VADER to compute sentiment scores for the reviews.

7.5.1 Initialize VADER Sentiment Analyzer

[18]:
```
# Initialize VADER
sid = SentimentIntensityAnalyzer()
```

7.5.2 Calculate Sentiment Scores

We calculate the compound sentiment score for each review.

[19]:
```
# Function to get sentiment scores
def get_sentiment_score(text):
    scores = sid.polarity_scores(text)
    return scores['compound']

# Apply to DataFrame
df['sentiment_score'] = df['text'].apply(get_sentiment_score)
```

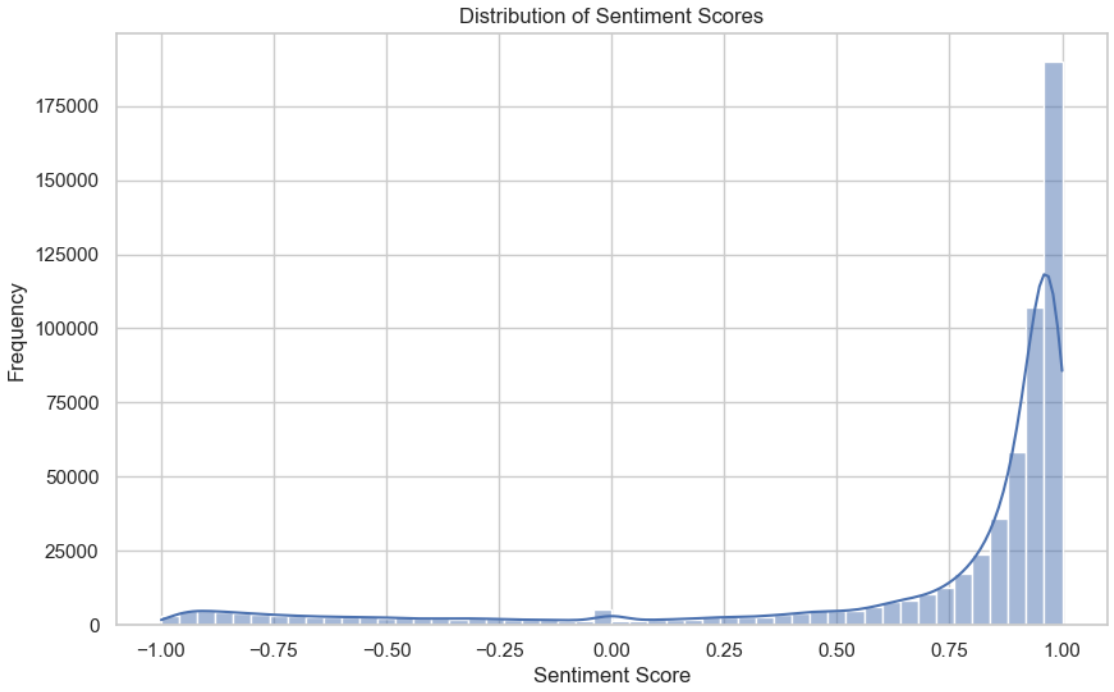
7.5.3 Plot Sentiment Score Distribution

We visualize the distribution of sentiment scores.

[20]:
```
plt.figure(figsize=(10,6))
sns.histplot(df['sentiment_score'], bins=50, kde=True)
plt.title('Distribution of Sentiment Scores')
plt.xlabel('Sentiment Score')
```

```
plt.ylabel('Frequency')
plt.show()
```

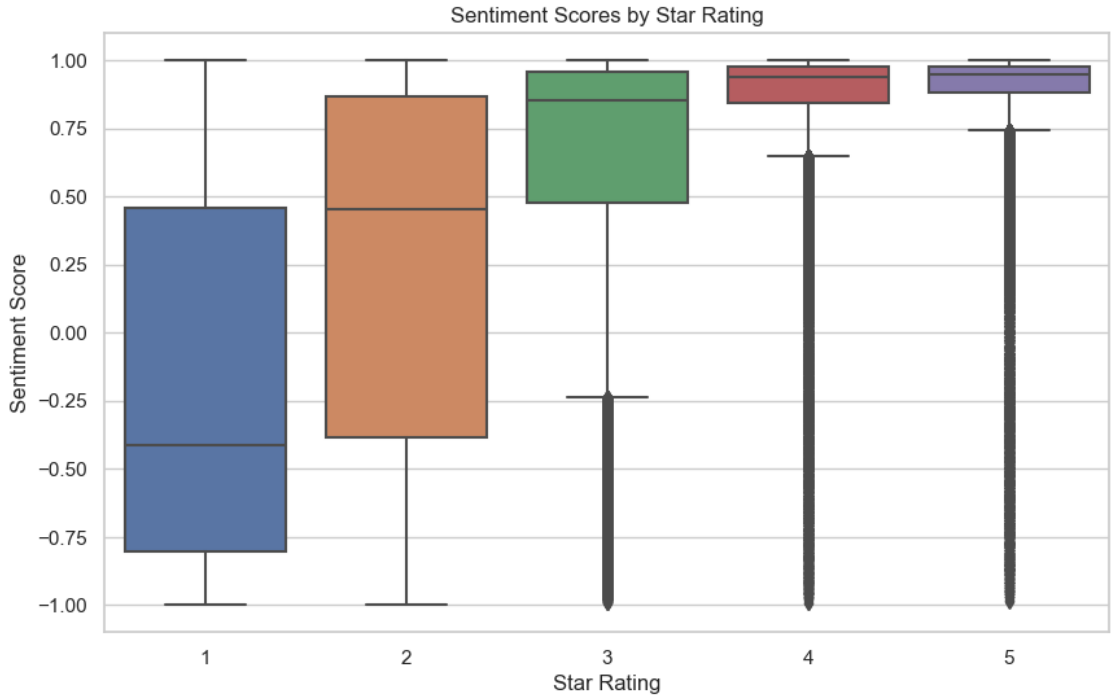
```
/Users/esteffrivero/anaconda3/lib/python3.11/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



7.5.4 Analyze Sentiment by Star Rating

We examine how sentiment scores vary across different star ratings.

[21]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='stars_review', y='sentiment_score', data=df)
plt.title('Sentiment Scores by Star Rating')
plt.xlabel('Star Rating')
plt.ylabel('Sentiment Score')
plt.show()
```



7.6 Geographical Analysis

7.6.1 Average Ratings by State

We can analyze the average ratings by state to see regional differences.

[22]:
```
state_ratings = df.groupby('state')['stars_review'].mean().reset_index()
state_ratings = state_ratings.sort_values(by='stars_review', ascending=False)

print("\nAverage Ratings by State:")
display(state_ratings)
```

```
Average Ratings by State:
```

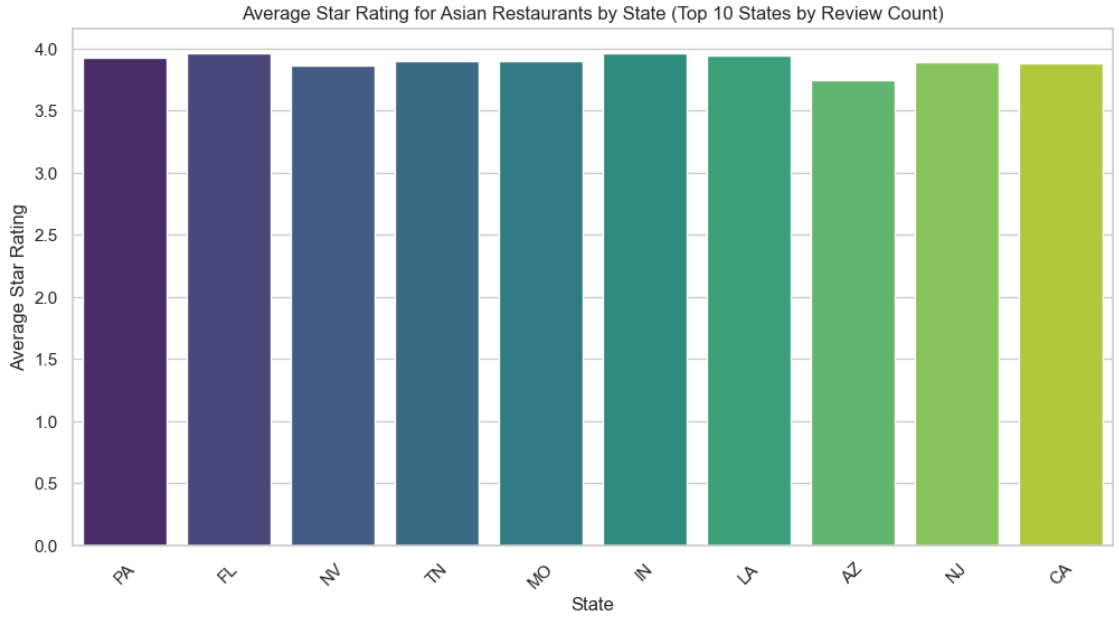
	state	stars_review
6	IN	3.962005
3	FL	3.960375
7	LA	3.946596
11	PA	3.922307
12	TN	3.899695
8	MO	3.896109
9	NJ	3.887785
1	CA	3.881727
10	NV	3.858819
4	ID	3.797508

```
0     AZ      3.747618
2     DE      3.746036
5     IL      3.714320
```

7.6.2 Plot Average Ratings by State

We visualize the top 10 states by average star rating.

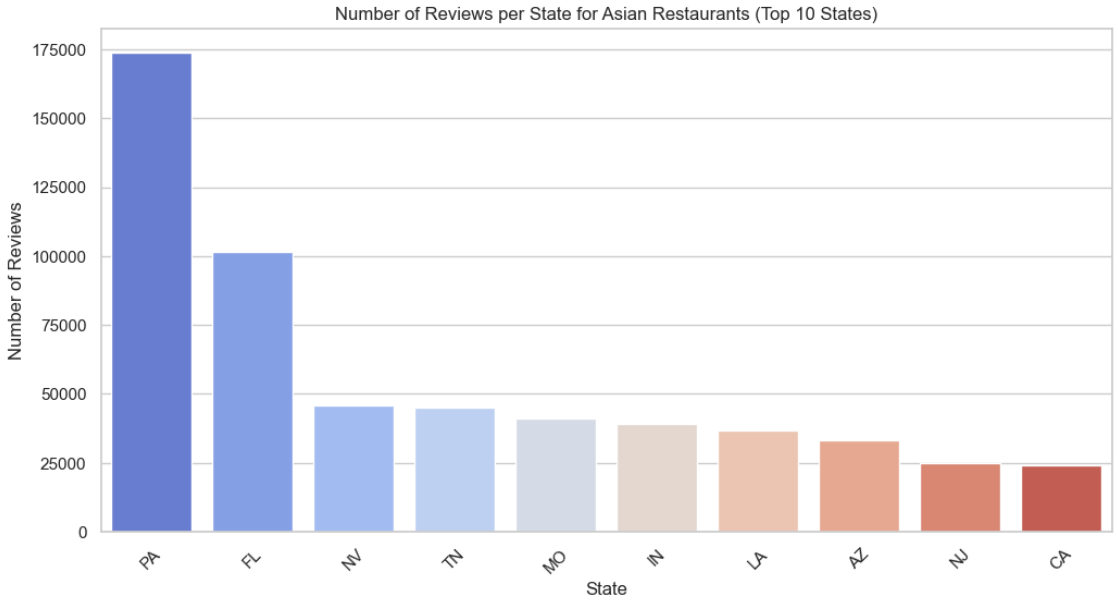
[23]:
```
# Limit to top 10 states with most reviews
state_review_counts = df['state'].value_counts().reset_index()
state_review_counts.columns = ['state', 'review_count']

# Merge with average ratings
state_data = pd.merge(state_ratings, state_review_counts, on='state')
top_states = state_data.sort_values(by='review_count', ascending=False).head(10)

plt.figure(figsize=(12,6))
sns.barplot(x='state', y='stars_review', data=top_states, palette='viridis')
plt.title('Average Star Rating for Asian Restaurants by State (Top 10 States by␣
 ↪Review Count)')
plt.xlabel('State')
plt.ylabel('Average Star Rating')
plt.xticks(rotation=45)
plt.show()
```



7.6.3 Number of Reviews per State

Understanding the number of reviews per state provides context to the average ratings.

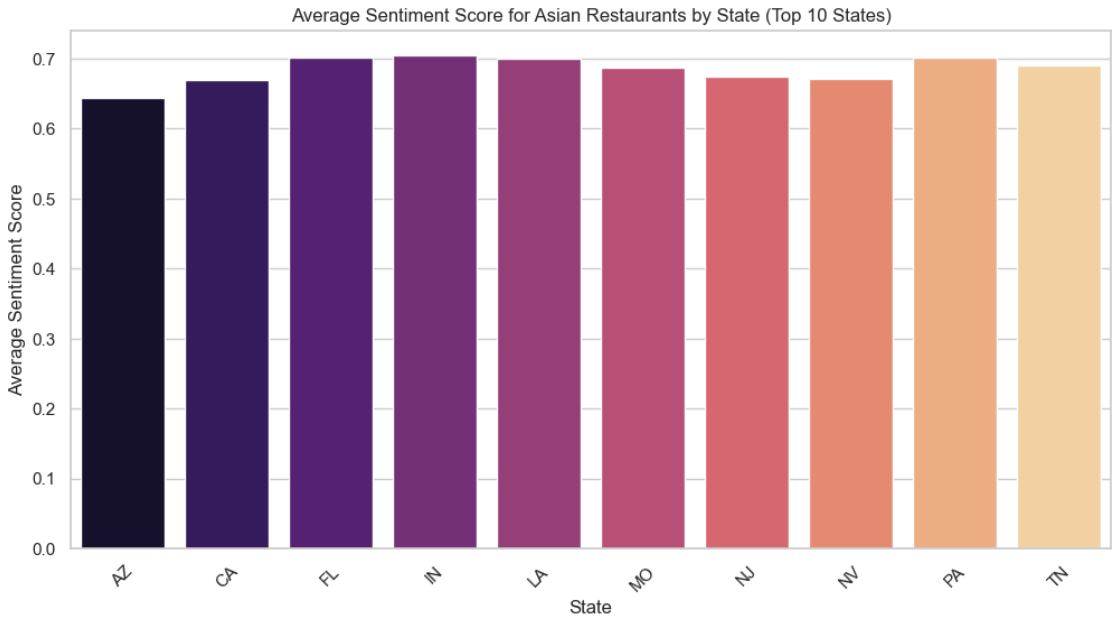
[24]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='state', y='review_count', data=top_states, palette='coolwarm')
plt.title('Number of Reviews per State for Asian Restaurants (Top 10 States)')
plt.xlabel('State')
plt.ylabel('Number of Reviews')
plt.xticks(rotation=45)
plt.show()
```



7.6.4 Average Sentiment Scores by State

We analyze the average sentiment scores by state.

[25]:
```
state_sentiments = df.groupby('state')['sentiment_score'].mean().reset_index()
state_sentiments = pd.merge(state_sentiments, state_review_counts, on='state')
top_states_sentiments = state_sentiments[state_sentiments['state'].
 ↪isin(top_states['state'])]

plt.figure(figsize=(12,6))
sns.barplot(x='state', y='sentiment_score', data=top_states_sentiments,␣
 ↪palette='magma')
plt.title('Average Sentiment Score for Asian Restaurants by State (Top 10␣
 ↪States)')
plt.xlabel('State')
plt.ylabel('Average Sentiment Score')
plt.xticks(rotation=45)
plt.show()
```



7.7 Correlation Analysis

7.7.1 Correlation Between Review Length and Star Rating

We calculate the correlation coefficient.

[26]:
```
correlation = df['review_length'].corr(df['stars_review'])
print(f"\nCorrelation between Review Length and Star Rating: {correlation:.2f}")
```

```
Correlation between Review Length and Star Rating: -0.17
```

7.7.2 Box Plot

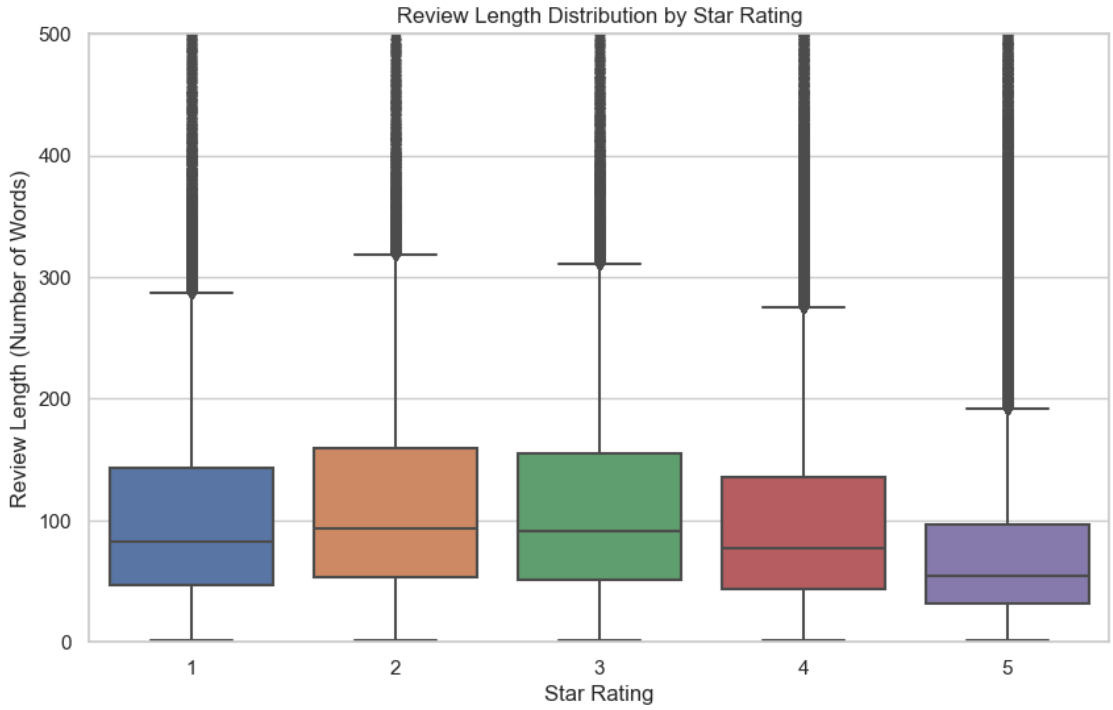
We plot the relationship between review length and star rating.

[27]:
```
# Calculate quantiles
quantiles = df['review_length'].quantile([0.25, 0.5, 0.75, 0.90, 0.95, 0.99, 1.
  ↪0])
print("\nQuantiles of Review Lengths:")
print(quantiles)
```

```
Quantiles of Review Lengths:
0.25      38.0
0.50      68.0
0.75     122.0
0.90     200.0
0.95     265.0
```

```
0.99     440.0
1.00    1021.0
Name: review_length, dtype: float64
```

[28]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='stars_review', y='review_length', data=df)
plt.title('Review Length Distribution by Star Rating')
plt.xlabel('Star Rating')
plt.ylabel('Review Length (Number of Words)')
plt.ylim(0, 500)  # Adjust the limit based on your data
plt.show()
```



7.8 Word Clouds

We create word clouds to visualize the most frequent words in the reviews.

7.8.1 Generate Word Cloud for All Reviews

[29]:
```
# Combine all tokens into one text
all_text = ' '.join([' '.join(tokens) for tokens in df['tokens']])

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
  ↪generate(all_text)
```

```
# Display the generated image
plt.figure(figsize=(15,7.5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of All Reviews (After Removing Custom Stop Words)')
plt.show()
```

7.8.2 Word Cloud for Positive and Negative Reviews

[30]:
```
# Positive reviews (stars_review >= 4)
positive_reviews = df[df['stars_review'] >= 4]
positive_text = ' '.join([' '.join(tokens) for tokens in␣
 ↪positive_reviews['tokens']])
wordcloud_positive = WordCloud(width=800, height=400, background_color='white').
 ↪generate(positive_text)

# Negative reviews (stars_review <= 2)
negative_reviews = df[df['stars_review'] <= 2]
negative_text = ' '.join([' '.join(tokens) for tokens in␣
 ↪negative_reviews['tokens']])
wordcloud_negative = WordCloud(width=800, height=400, background_color='white').
 ↪generate(negative_text)

# Display word clouds
fig, axes = plt.subplots(1, 2, figsize=(20,10))
axes[0].imshow(wordcloud_positive, interpolation='bilinear')
```

```
axes[0].axis('off')
axes[0].set_title('Positive Reviews Word Cloud (After Removing Custom Stop␣
↪Words)')

axes[1].imshow(wordcloud_negative, interpolation='bilinear')
axes[1].axis('off')
axes[1].set_title('Negative Reviews Word Cloud (After Removing Custom Stop␣
↪Words)')

plt.show()
```

8. Save Preprocessed Data

We save the preprocessed DataFrame for future use.

[31]:
```
# Save the DataFrame with new features
df.to_csv('asian_restaurants_reviews_us_preprocessed.csv', index=False)
```

02_Advanced_Analysis_Yelp_Asian_Restaurants

November 17, 2024

Advanced Analysis of Yelp Reviews for Asian Restaurants in the United States

This notebook continues the exploratory data analysis by diving deeper into the Yelp reviews dataset for Asian restaurants in the United States. We will perform:

- Topic Modeling using Latent Dirichlet Allocation (LDA)
- Aspect-Based Sentiment Analysis
- Temporal Analysis (if time permits)
- Further insights and visualizations

1. Introduction

In this notebook, we build upon the previous exploratory data analysis to perform advanced analyses on Yelp reviews of Asian restaurants in the United States. The focus is on uncovering hidden themes in the reviews through topic modeling and analyzing customer sentiments towards specific aspects of their dining experience.

2. Import Libraries

We start by importing the necessary libraries for data manipulation, natural language processing, and visualization.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re

# NLP libraries
import gensim
from gensim import corpora
from gensim.models.ldamodel import LdaModel
from gensim.models.coherencemodel import CoherenceModel
from nltk.corpus import stopwords
from nltk import word_tokenize, download
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Visualization libraries
```

```
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

# For handling warnings
import warnings
warnings.filterwarnings('ignore')

# For NLP preprocessing
#download('stopwords')
#download('punkt')
#download('vader_lexicon')

# Set visualization style
sns.set(style='whitegrid')
```

3. Load Preprocessed Data

We load the preprocessed DataFrame saved from the previous analysis.

[2]:
```
# Load the preprocessed data
df = pd.read_csv('asian_restaurants_reviews_us_preprocessed.csv')

# Display the first few rows
print("Preprocessed Data:")
display(df.head())
```

```
Preprocessed Data:
```

	review_id	user_id	business_id	stars_review	useful	funny	cool	text	date
0	AqPFM1eE6RsU23_auESxiA	_7bHUi9Uuf5__HHc_Q8guQ	kxX2SOes4o-D3ZQBkiMRfA	5	1	0	1	Wow! Yummy, different, delicious. Our favo…	2015-01-04 00:01:03
1	pUycOfUwM8vqX7KjRRhUEA	59MxRhNVhU9MYndMkz0wtw	gebiRewfieSdtt17PTW6Zg	3	0	0	0	Had a party of 6 here for hibachi. Our waitres…	2016-07-25 07:31:06
2	eCiWBf1CJ0Zdv1uVarEhhw	OhECKhQEexFypOMY6kypRw	vC2qm1y3Au5czBtbhc-DNw	4	0	0	0	Yes, this is the only sushi place in town. How…	2013-09-04 03:48:20
3	G_5UczbCBJriUAbxz3J7Tw	clWLI5OZP2ad25ugMVI8gg	x4XdNhp0Xn8lOivzc77J-g	5	0	0	0	Best thai food in the area. Everything was au…	2013-08-15 15:27:51
4	r2IBPY_E8AE5_GpsqlONyg	IKbjLnfBQtEyVzEu8CuOLg	VJEzpfLs_Jnzgqh5A_FVTg	4	0	0	0	It was my fiance's birthday and he decided he …	2014-04-01 13:05:18

```
                            name  … stars_business review_count is_open  \
0                          Zaika  …            4.0          181       1
1  Hibachi Steak House & Sushi Bar  …            3.5          488       1
2                     Sushi Teri  …            3.0          167       1
3           Thai Place Restaurant  …            4.5          222       1
4       Jasmine Rice - Rittenhouse  …            3.5          307       1

                                        attributes  \
0  {'Caters': 'True', 'Ambience': "{'romantic': F…
1  {'Corkage': 'False', 'RestaurantsTakeOut': 'Tr…
2  {'RestaurantsReservations': 'True', 'NoiseLeve…
3  {'OutdoorSeating': 'False', 'RestaurantsDelive…
4  {'RestaurantsPriceRange2': '2', 'RestaurantsAt…

                                    categories  \
0          Halal, Pakistani, Restaurants, Indian
1  Steakhouses, Sushi Bars, Restaurants, Japanese
2                      Restaurants, Sushi Bars
3                           Thai, Restaurants
4                Soup, Thai, Restaurants, Salad

                                          hours  is_asian  review_length  \
0  {'Tuesday': '11:0-21:0', 'Wednesday': '11:0-21…      True             40
1                            {'Monday': '0:0-0:0'}      True             97
2  {'Monday': '17:0-22:0', 'Tuesday': '17:0-22:0'…      True             58
3  {'Tuesday': '17:0-21:30', 'Wednesday': '17:0-2…      True             18
4  {'Monday': '13:30-22:30', 'Tuesday': '13:30-22…      True            214

                                        tokens sentiment_score
0  ['wow', 'yummy', 'different', 'delicious', 'fa…          0.9588
1  ['party', 'hibachi', 'brought', 'separate', 's…          0.9782
2  ['yes', 'sushi', 'town', 'however', 'great', '…          0.9622
3  ['best', 'thai', 'everything', 'authentic', 'd…          0.8910
4  ['fiances', 'birthday', 'decided', 'wanted', '…          0.9854

[5 rows x 26 columns]
```

4. Topic Modeling with LDA

We perform topic modeling to uncover hidden themes in the reviews.

4.1 Prepare Data for LDA

We need to prepare the data by creating a dictionary and corpus suitable for LDA.

4.1.1 Preprocessing Text Data If the tokenized text is not available, we need to preprocess the text again. However, since we have preprocessed tokens, we can proceed.

[3]:
```
# Check if 'tokens' column exists
if 'tokens' in df.columns:
    # Convert string representation of lists to actual lists
    df['tokens'] = df['tokens'].apply(eval)
else:
    # Preprocess text if 'tokens' column is not available
    stop_words = set(stopwords.words('english'))

    # Add restaurant-specific stop words as before
    restaurant_stop_words = [
        'restaurant', 'food', 'place', 'menu', 'order', 'ordered', 'eat', 'ate',
        'dish', 'dishes', 'meal', 'meals', 'server', 'servers', 'waiter',␣
 ↪'waitress',
        'staff', 'table', 'tables', 'seat', 'seated', 'drinks', 'drink',␣
 ↪'appetizer',
        'appetizers', 'main', 'course', 'courses', 'dessert', 'desserts',␣
 ↪'lunch',
        'dinner', 'breakfast', 'brunch', 'location', 'area', 'experience',␣
 ↪'experiences',
        'customer', 'customers', 'service', 'services', 'takeout', 'take',␣
 ↪'out', 'time',
        'times', 'visit', 'visits', 'visited', 'day', 'days', 'night',␣
 ↪'nights', 'people',
        'person', 'group', 'groups', 'friend', 'friends', 'family'
    ]
    stop_words.update(restaurant_stop_words)

    # Preprocess text
    def preprocess_text(text):
        text = text.lower()
        text = re.sub(r'[^a-z\s]', '', text)
        tokens = word_tokenize(text)
        tokens = [word for word in tokens if word not in stop_words]
        return tokens

    df['tokens'] = df['text'].apply(preprocess_text)
```

4.1.2 Creating Dictionary and Corpus

[4]:
```
# Create a dictionary representation of the documents
dictionary = corpora.Dictionary(df['tokens'])

# Filter out extremes to remove very rare and very common words
dictionary.filter_extremes(no_below=15, no_above=0.5)

# Create the Bag-of-Words corpus
corpus = [dictionary.doc2bow(text) for text in df['tokens']]
```

```
# Print basic information
print(f'Number of unique tokens: {len(dictionary)}')
print(f'Number of documents: {len(corpus)}')
```

```
Number of unique tokens: 23923
Number of documents: 591773
```

4.2 Build LDA Model

We build the LDA model to discover topics in the reviews.

4.2.1 Determining Optimal Number of Topics We test different numbers of topics and calculate coherence scores to find the optimal number.

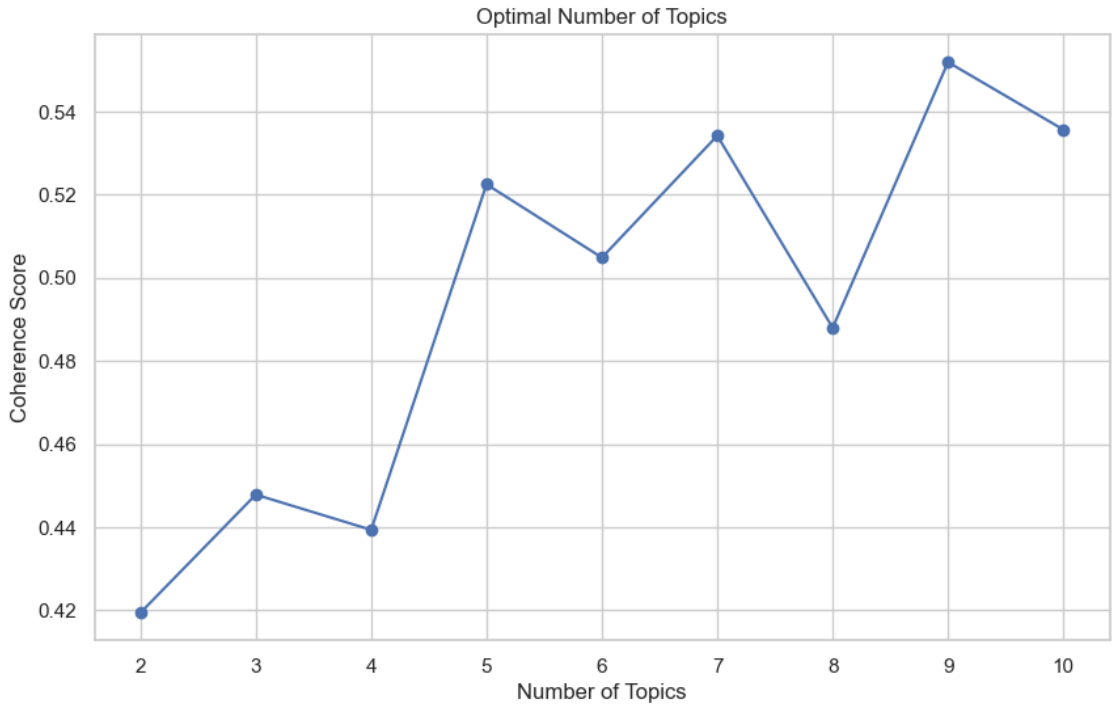
[5]:
```
coherence_scores = []
model_list = []
topic_range = range(2, 11)

for num_topics in topic_range:
    lda_model = LdaModel(corpus=corpus,
                         id2word=dictionary,
                         num_topics=num_topics,
                         random_state=42,
                         chunksize=1000,
                         passes=10,
                         alpha='auto',
                         per_word_topics=True)
    model_list.append(lda_model)
    coherencemodel = CoherenceModel(model=lda_model, texts=df['tokens'],␣
 ↪dictionary=dictionary, coherence='c_v')
    coherence_scores.append(coherencemodel.get_coherence())
    print(f'Number of Topics: {num_topics}, Coherence Score: {coherencemodel.
 ↪get_coherence():.4f}')
```

```
Number of Topics: 2, Coherence Score: 0.4194
Number of Topics: 3, Coherence Score: 0.4478
Number of Topics: 4, Coherence Score: 0.4394
Number of Topics: 5, Coherence Score: 0.5225
Number of Topics: 6, Coherence Score: 0.5049
Number of Topics: 7, Coherence Score: 0.5342
Number of Topics: 8, Coherence Score: 0.4880
Number of Topics: 9, Coherence Score: 0.5519
Number of Topics: 10, Coherence Score: 0.5357
```

4.2.2 Plotting Coherence Scores

[6]:
```
plt.figure(figsize=(10,6))
plt.plot(topic_range, coherence_scores, marker='o')
```

```
plt.xlabel('Number of Topics')
plt.ylabel('Coherence Score')
plt.title('Optimal Number of Topics')
plt.show()
```



4.3 Explore and Interpret Topics

[11]:
```
optimal_num_topics = 9  # Replace with the optimal number determined

# Train the final LDA model
lda_model = LdaModel(corpus=corpus,
                     id2word=dictionary,
                     num_topics=optimal_num_topics,
                     random_state=42,
                     chunksize=1000,
                     passes=10,
                     alpha='auto',
                     per_word_topics=True)
```

4.3.1 Display Topics

[12]:
```
from pprint import pprint

# Print the keywords in the topics
```

```
pprint(lda_model.print_topics(num_words=10))
```

```
[(0,
  '0.061*"ramen" + 0.053*"pho" + 0.049*"tea" + 0.035*"pork" + 0.032*"broth" + '
  '0.021*"boba" + 0.021*"vietnamese" + 0.020*"bowl" + 0.017*"terminal" + '
  '0.016*"mi"'),
 (1,
  '0.051*"great" + 0.027*"delicious" + 0.025*"best" + 0.022*"love" + '
  '0.021*"always" + 0.021*"definitely" + 0.021*"try" + 0.018*"back" + '
  '0.018*"fresh" + 0.018*"amazing"'),
 (2,
  '0.066*"good" + 0.034*"like" + 0.029*"really" + 0.024*"get" + 0.016*"im" + '
  '0.015*"dont" + 0.014*"chinese" + 0.014*"much" + 0.014*"pretty" + '
  '0.013*"little"'),
 (3,
  '0.041*"chicken" + 0.031*"rice" + 0.026*"thai" + 0.023*"sauce" + '
  '0.022*"fried" + 0.021*"spicy" + 0.019*"soup" + 0.016*"flavor" + '
  '0.015*"shrimp" + 0.015*"hot"'),
 (4,
  '0.016*"us" + 0.012*"got" + 0.012*"came" + 0.012*"didnt" + 0.011*"even" + '
  '0.011*"would" + 0.011*"back" + 0.010*"one" + 0.009*"could" + 0.009*"went"'),
 (5,
  '0.085*"indian" + 0.047*"chicken" + 0.046*"buffet" + 0.041*"naan" + '
  '0.033*"masala" + 0.026*"lamb" + 0.023*"tikka" + 0.021*"butter" + '
  '0.017*"paneer" + 0.015*"garlic"'),
 (6,
  '0.016*"options" + 0.013*"bar" + 0.012*"market" + 0.011*"inside" + '
  '0.010*"dining" + 0.010*"parking" + 0.010*"large" + 0.009*"variety" + '
  '0.009*"items" + 0.008*"fun"'),
 (7,
  '0.042*"cheese" + 0.031*"cream" + 0.029*"ice" + 0.019*"datz" + 0.017*"pizza" '
  '+ 0.015*"sandwich" + 0.014*"fries" + 0.013*"burger" + 0.012*"mac" + '
  '0.012*"steak"'),
 (8,
  '0.169*"sushi" + 0.078*"roll" + 0.074*"rolls" + 0.042*"fish" + 0.021*"tuna" '
  '+ 0.020*"salmon" + 0.018*"salad" + 0.014*"japanese" + 0.014*"fresh" + '
  '0.014*"tempura"')]
```

4.4 Visualize Topics

We use pyLDAvis to visualize the topics interactively.

[13]:
```
# Prepare the visualization
pyLDAvis.enable_notebook()
lda_display = gensimvis.prepare(lda_model, corpus, dictionary,␣
 ↪sort_topics=False)

# Display the visualization
```

```
pyLDAvis.display(lda_display)
```

[13]: <IPython.core.display.HTML object>

4.5 Assign Topics to Reviews

We assign the dominant topic to each review.

[15]:
```
def format_topics_sentences(ldamodel, corpus, texts):
    # Initialize an empty list to hold the results
    sent_topics_list = []

    for i, row_list in enumerate(ldamodel[corpus]):
        if ldamodel.per_word_topics:
            row = row_list[0]
        else:
            row = row_list

        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the dominant topic, percentage contribution, and keywords for␣
 ↪each review
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # Dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_list.append([int(topic_num), round(prop_topic, 4),␣
 ↪topic_keywords])
            else:
                break

    # Create a DataFrame from the list
    sent_topics_df = pd.DataFrame(sent_topics_list, columns=['Dominant_Topic',␣
 ↪'Perc_Contribution', 'Topic_Keywords'])

    # Add original text to the end of the output
    contents = texts.reset_index(drop=True)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return sent_topics_df

# Call the function
df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model,␣
 ↪corpus=corpus, texts=df['text'])

# Format the DataFrame
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic',␣
 ↪'Topic_Perc_Contrib', 'Keywords', 'Text']
```

```
# Display the top 5 rows
df_dominant_topic.head()
```

[15]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	1	0.2854	great, delicious, best, love, always, definite…	Wow! Yummy, different, delicious. Our favo…
1	1	4	0.3834	us, got, came, didnt, even, would, back, one, …	Had a party of 6 here for hibachi. Our waitres…
2	2	1	0.3023	great, delicious, best, love, always, definite…	Yes, this is the only sushi place in town. How…
3	3	1	0.4392	great, delicious, best, love, always, definite…	Best thai food in the area. Everything was au…
4	4	3	0.2959	chicken, rice, thai, sauce, fried, spicy, soup…	It was my fiance's birthday and he decided he …

5. Aspect-Based Sentiment Analysis

We analyze sentiments associated with each identified topic.

5.1 Sentiment Analysis per Topic

5.1.1 Merge Dominant Topics with Original Data

[17]:
```
# Merge the dominant topic back into the main DataFrame
df_aspect_sentiment = pd.merge(df, df_dominant_topic[['Document_No',␣
↪'Dominant_Topic']], left_index=True, right_on='Document_No', how='left')

# Display the first few rows
df_aspect_sentiment.head()
```

[17]:

	review_id	user_id	business_id	stars_review	useful	funny	cool
0	AqPFMleE6RsU23_auESxiA	_7bHUi9Uuf5__HHc_Q8guQ	kxX2SOes4o-D3ZQBkiMRfA	5	1	0	1
1	pUycOfUwM8vqX7KjRRhUEA	59MxRhNVhU9MYndMkz0wtw	gebiRewfieSdtt17PTW6Zg				
2	eCiWBf1CJ0Zdv1uVarEhhw	OhECKhQEexFypOMY6kypRw	vC2qm1y3Au5czBtbhc-DNw				
3	G_5UczbCBJriUAbxz3J7Tw	clWLI5OZP2ad25ugMVI8gg	x4XdNhp0Xn8l0ivzc77J-g				
4	r2IBPY_E8AE5_GpsqlONyg	IKbjLnfBQtEyVzEu8CuOLg	VJEzpfLs_Jnzgqh5A_FVTg				

```
1             3       0      0     0
2             4       0      0     0
3             5       0      0     0
4             4       0      0     0

                                                text                 date  \
0  Wow!  Yummy, different,  delicious.   Our favo…  2015-01-04 00:01:03
1  Had a party of 6 here for hibachi. Our waitres…  2016-07-25 07:31:06
2  Yes, this is the only sushi place in town. How…  2013-09-04 03:48:20
3  Best thai food in the area.  Everything was au…  2013-08-15 15:27:51
4  It was my fiance's birthday and he decided he …  2014-04-01 13:05:18

                            name  … is_open  \
0                         Zaika  …       1
1  Hibachi Steak House & Sushi Bar  …       1
2                    Sushi Teri  …       1
3           Thai Place Restaurant  …       1
4      Jasmine Rice - Rittenhouse  …       1

                                         attributes  \
0  {'Caters': 'True', 'Ambience': "{'romantic': F…
1  {'Corkage': 'False', 'RestaurantsTakeOut': 'Tr…
2  {'RestaurantsReservations': 'True', 'NoiseLeve…
3  {'OutdoorSeating': 'False', 'RestaurantsDelive…
4  {'RestaurantsPriceRange2': '2', 'RestaurantsAt…

                                    categories  \
0          Halal, Pakistani, Restaurants, Indian
1  Steakhouses, Sushi Bars, Restaurants, Japanese
2                      Restaurants, Sushi Bars
3                           Thai, Restaurants
4                Soup, Thai, Restaurants, Salad

                                           hours  is_asian  review_length  \
0  {'Tuesday': '11:0-21:0', 'Wednesday': '11:0-21…      True             40
1                            {'Monday': '0:0-0:0'}      True             97
2  {'Monday': '17:0-22:0', 'Tuesday': '17:0-22:0'…      True             58
3  {'Tuesday': '17:0-21:30', 'Wednesday': '17:0-2…      True             18
4  {'Monday': '13:30-22:30', 'Tuesday': '13:30-22…      True            214

                                          tokens  sentiment_score  \
0  [wow, yummy, different, delicious, favorite, l…           0.9588
1  [party, hibachi, brought, separate, sushi, ord…           0.9782
2  [yes, sushi, town, however, great, youre, crav…           0.9622
3  [best, thai, everything, authentic, delicious,…           0.8910
4  [fiances, birthday, decided, wanted, good, lai…           0.9854
```

```
   Document_No Dominant_Topic
0            0               1
1            1               4
2            2               1
3            3               1
4            4               3

[5 rows x 28 columns]
```

5.1.2 Calculate Average Sentiment Score per Topic

[18]:
```
# Calculate average sentiment score per topic
topic_sentiments = df_aspect_sentiment.
 ↪groupby('Dominant_Topic')['sentiment_score'].mean().reset_index()

# Map topic numbers to keywords for better interpretation
topic_keywords = df_dominant_topic[['Dominant_Topic', 'Keywords']].
 ↪drop_duplicates()
topic_sentiments = pd.merge(topic_sentiments, topic_keywords,␣
 ↪on='Dominant_Topic')

# Rename columns
topic_sentiments.columns = ['Dominant_Topic', 'Average_Sentiment_Score',␣
 ↪'Topic_Keywords']

# Display the results
topic_sentiments.sort_values(by='Average_Sentiment_Score', ascending=False,␣
 ↪inplace=True)
display(topic_sentiments)
```

```
   Dominant_Topic  Average_Sentiment_Score  \
1               1                 0.889338
6               6                 0.849861
8               8                 0.814052
5               5                 0.754287
3               3                 0.741322
0               0                 0.738163
2               2                 0.726697
7               7                 0.717528
4               4                 0.368172

                                     Topic_Keywords
1  great, delicious, best, love, always, definite…
6  options, bar, market, inside, dining, parking,…
8  sushi, roll, rolls, fish, tuna, salmon, salad,…
5  indian, chicken, buffet, naan, masala, lamb, t…
3  chicken, rice, thai, sauce, fried, spicy, soup…
0  ramen, pho, tea, pork, broth, boba, vietnamese…
```

```
2  good, like, really, get, im, dont, chinese, mu…
7  cheese, cream, ice, datz, pizza, sandwich, fri…
4  us, got, came, didnt, even, would, back, one, …
```

5.2 Visualize Sentiments per Topic

[19]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='Dominant_Topic', y='Average_Sentiment_Score',␣
 ↪data=topic_sentiments, palette='coolwarm')
plt.title('Average Sentiment Score per Topic')
plt.xlabel('Topic Number')
plt.ylabel('Average Sentiment Score')
plt.xticks(rotation=45)
plt.show()
```