Restaurants receive thousands of customer reviews, but star ratings alone fail to explain why ratings increase or decline. Reviews often contain rich information about food quality, service, pricing, ambience, and operational factors, yet this information remains unstructured and difficult to analyze at scale. This limits restaurants' ability to identify the drivers of customer satisfaction and dissatisfaction

Notebook 3: Insight Analysis

This notebook performs exploratory and diagnostic analysis on a previously constructed review-level dataset (aspect_df) to explain why restaurant ratings vary.

By analyzing sentiment distributions across review topics (aspects) and linking them to restaurant operational attributes, this notebook identifies the key drivers of positive and negative customer sentiment. The goal is to translate unstructured review text into actionable insights that help explain changes in star ratings beyond the ratings themselves.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Load Prepared Analytical Dataset

```
import pandas as pd
import pickle

PATH = "/content/drive/MyDrive/restaurant_sentiment_model/aspect_df.pkl"

#load
aspect_df = pd.read_pickle(PATH)
```

```
aspect_df.head()
```

	business_id	review	topic	sentiment	AcceptsInsurance	AgesAllowed	Alcohol	Ambience	BYO
0	YtSqYv1Q_pOltsVPSx54SA	Tremendous service (Big shout out to Douglas) ...	-1	2	Unknown	Unknown	'full_bar'	{'romantic': False, 'intimate': False, 'classy...	Unknow
1	aY_n9RSaD2Yw09jSFFePew	We visited once and were very disappointed in ...	3	0	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknow
2	18eWJFJbXyR9j_5xfcRLYA	This is the first time I tried this place and ...	6	2	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknow
3	jOOOrH5n2ijnsZKxzPSAiw	This is one of the busiest Chick fil A's I've ...	-1	2	Unknown	Unknown	u'none'	{'romantic': False, 'intimate': False, 'touris...	Unknow
4	1QVB0_-piu0GXes87BXeGw	Love this place...best hot dogs and chili dogs...	-1	2	Unknown	Unknown	u'none'	{'touristy': False, 'hipster': False, 'romanti...	Unknow

5 rows × 43 columns

```
aspect_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 43 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   business_id                 10000 non-null  object
 1   review                      10000 non-null  object
 2   topic                       10000 non-null  int64
 3   sentiment                   10000 non-null  int64
 4   AcceptsInsurance           10000 non-null  object
 5   AgesAllowed                10000 non-null  object
 6   Alcohol                    10000 non-null  object
 7   Ambience                   10000 non-null  object
 8   BYOB                       10000 non-null  object
 9   BYOBCorkage                10000 non-null  object
 10  BestNights                 10000 non-null  object
 11  BikeParking                10000 non-null  object
 12  BusinessAcceptsBitcoin     10000 non-null  object
 13  BusinessAcceptsCreditCards  10000 non-null  object
 14  BusinessParking            10000 non-null  object
 15  ByAppointmentOnly          10000 non-null  object
 16  Caters                     10000 non-null  object
 17  CoatCheck                  10000 non-null  object
 18  Corkage                    10000 non-null  object
 19  DietaryRestrictions        10000 non-null  object
 20  DogsAllowed                10000 non-null  object
 21  DriveThru                  10000 non-null  object
 22  GoodForDancing             10000 non-null  object
 23  GoodForKids                10000 non-null  object
 24  GoodForMeal                10000 non-null  object
 25  HairSpecializesIn          10000 non-null  object
 26  HappyHour                  10000 non-null  object
 27  HasTV                      10000 non-null  object
 28  Music                      10000 non-null  object
 29  NoiseLevel                 10000 non-null  object
 30  Open24Hours                10000 non-null  object
 31  OutdoorSeating             10000 non-null  object
 32  RestaurantsAttire          10000 non-null  object
 33  RestaurantsCounterService  10000 non-null  object
 34  RestaurantsDelivery        10000 non-null  object
 35  RestaurantsGoodForGroups   10000 non-null  object
 36  RestaurantsPriceRange2     10000 non-null  object
 37  RestaurantsReservations    10000 non-null  object
 38  RestaurantsTableService    10000 non-null  object
 39  RestaurantsTakeOut         10000 non-null  object
 40  Smoking                    10000 non-null  object
 41  WheelchairAccessible       10000 non-null  object
 42  WiFi                       10000 non-null  object
dtypes: int64(2), object(41)
memory usage: 3.3+ MB
```

```
pip install -U bertopic
```

```
Requirement already satisfied: scikit-learn>=1.0 in /usr/local/lib/python3.12/dist-packages (from bertopic) (1.6.1)
Requirement already satisfied: sentence-transformers>=0.4.1 in /usr/local/lib/python3.12/dist-packages (from bertopic) (5.
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.12/dist-packages (from bertopic) (4.67.1)
Requirement already satisfied: llvmlite>0.36.0 in /usr/local/lib/python3.12/dist-packages (from bertopic) (0.43.0)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.12/dist-packages (from hdbscan>=0.8.29->bertopic) (1.1
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.12/dist-packages (from hdbscan>=0.8.29->bertopic) (1.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.1.5->bert
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.1.5->bertopic) (202
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.1.5->bertopic) (2
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly>=4.7.0->bertopic) (
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from plotly>=4.7.0->bertopic) (25.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=1.0->be
Requirement already satisfied: transformers<6.0.0,>=4.41.0 in /usr/local/lib/python3.12/dist-packages (from sentence-trans
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from sentence-transformers>=0.4.1
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.12/dist-packages (from sentence-transform
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/local/lib/python3.12/dist-packages (from sentence-transfor
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.12/dist-packages (from umap-learn>=0.5.0->bertopic)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.12/dist-packages (from umap-learn>=0.5.0->bertop
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sente
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas>=1
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-transfo
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-tran
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-tr
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-transformer
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.1
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch>=1.1
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0-
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers<6
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers<6.0.0,>=4.
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch>=1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch>=1.11.0->sen
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->hugging
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface-hub>=0.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface-h
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface-h
Downloading bertopic-0.17.4-py3-none-any.whl (154 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.7/154.7 kB 8.2 MB/s eta 0:00:00
Installing collected packages: bertopic
Successfully installed bertopic-0.17.4
```

```
from bertopic import BERTopic
from sentence_transformers import SentenceTransformer

embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
MODEL_PATH = "/content/drive/MyDrive/restaurant_sentiment_model/aspect"
topic_model = BERTopic.load(
    MODEL_PATH,
    embedding_model=embedding_model
)
```

```
/usr/local/lib/python3.12/dist-packages/hdbscan/robust_single_linkage_.py:175: SyntaxWarning: invalid escape sequence '\{'
  $max \{ core_k(a), core_k(b), 1/\alpha d(a,b) \}$.
```

```
modules.json:  100%                                    349/349  [00:00<00:00,  36.0kB/s]

config_sentence_transformers.json:  100%               116/116  [00:00<00:00,  13.8kB/s]

README.md:      10.5k/?  [00:00<00:00,  1.09MB/s]

sentence_bert_config.json:  100%                       53.0/53.0  [00:00<00:00,  6.39kB/s]

config.json:  100%                                     612/612  [00:00<00:00,  63.0kB/s]

model.safetensors:  100%                               90.9M/90.9M  [00:00<00:00,  163MB/s]

tokenizer_config.json:  100%                           350/350  [00:00<00:00,  37.5kB/s]

vocab.txt:      232k/?  [00:00<00:00,  12.4MB/s]

tokenizer.json:      466k/?  [00:00<00:00,  29.1MB/s]

special_tokens_map.json:  100%                         112/112  [00:00<00:00,  13.3kB/s]

config.json:  100%                                     190/190  [00:00<00:00,  23.7kB/s]
```

Topic and Sentiment Label Mapping

```
import matplotlib.pyplot as plt
import pandas as pd

#Map sentiment labels
sentiment_mapping = {0: "Negative", 1: "Neutral", 2: "Positive"}
aspect_df['sentiment_label'] = aspect_df['sentiment'].map(sentiment_mapping)

#Map topic IDs to descriptive labels
topic_info = topic_model.get_topic_info()

topic_mapping = {}
for topic_id in topic_info['Topic']:
    if topic_id == -1:
        topic_mapping[topic_id] = "Other / Outliers"
    else:
        words = [word for word, _ in topic_model.get_topic(topic_id)[:3]]  # top 3 words
        topic_mapping[topic_id] = " ".join(words)

aspect_df['topic_label'] = aspect_df['topic'].map(topic_mapping)
```

```
aspect_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 45 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   business_id                 10000 non-null  object
 1   review                      10000 non-null  object
 2   topic                       10000 non-null  int64
 3   sentiment                   10000 non-null  int64
 4   AcceptsInsurance            10000 non-null  object
 5   AgesAllowed                 10000 non-null  object
 6   Alcohol                     10000 non-null  object
 7   Ambience                    10000 non-null  object
 8   BYOB                        10000 non-null  object
 9   BYOBCorkage                 10000 non-null  object
 10  BestNights                  10000 non-null  object
 11  BikeParking                 10000 non-null  object
 12  BusinessAcceptsBitcoin      10000 non-null  object
 13  BusinessAcceptsCreditCards  10000 non-null  object
 14  BusinessParking             10000 non-null  object
 15  ByAppointmentOnly           10000 non-null  object
 16  Caters                      10000 non-null  object
 17  CoatCheck                   10000 non-null  object
 18  Corkage                     10000 non-null  object
 19  DietaryRestrictions         10000 non-null  object
 20  DogsAllowed                 10000 non-null  object
 21  DriveThru                   10000 non-null  object
 22  GoodForDancing              10000 non-null  object
 23  GoodForKids                 10000 non-null  object
 24  GoodForMeal                 10000 non-null  object
 25  HairSpecializesIn           10000 non-null  object
 26  HappyHour                   10000 non-null  object
 27  HasTV                       10000 non-null  object
 28  Music                       10000 non-null  object
 29  NoiseLevel                  10000 non-null  object
 30  Open24Hours                 10000 non-null  object
 31  OutdoorSeating              10000 non-null  object
 32  RestaurantsAttire           10000 non-null  object
 33  RestaurantsCounterService   10000 non-null  object
 34  RestaurantsDelivery         10000 non-null  object
 35  RestaurantsGoodForGroups    10000 non-null  object
 36  RestaurantsPriceRange2      10000 non-null  object
 37  RestaurantsReservations     10000 non-null  object
 38  RestaurantsTableService     10000 non-null  object
 39  RestaurantsTakeOut          10000 non-null  object
 40  Smoking                     10000 non-null  object
 41  WheelchairAccessible        10000 non-null  object
 42  WiFi                        10000 non-null  object
 43  sentiment_label             10000 non-null  object
 44  topic_label                 10000 non-null  object
dtypes: int64(2), object(43)
memory usage: 3.4+ MB
```

```
aspect_df.head()
```

	business_id	review	topic	sentiment	AcceptsInsurance	AgesAllowed	Alcohol	Ambience	BYO
0	YtSqYv1Q_pOltsVPSx54SA	Tremendous service (Big shout out to Douglas) ...	-1	2	Unknown	Unknown	'full_bar'	{'romantic': False, 'intimate': False, 'classy...	Unknow
1	aY_n9RSaD2Yw09jSFFePew	We visited once and were very disappointed in ...	3	0	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknow
2	18eWJFJbXyR9j_5xfcRLYA	This is the first time I tried this place and ...	6	2	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknow
3	jOOOrH5n2ijnsZKxzPSAiw	This is one of the busiest Chick fil A's I've ...	-1	2	Unknown	Unknown	u'none'	{'romantic': False, 'intimate': False, 'touris...	Unknow
4	1QVB0_-piu0GXes87BXeGw	Love this place...best hot dogs and chili dogs...	-1	2	Unknown	Unknown	u'none'	{'touristy': False, 'hipster': False, 'romanti...	Unknow

5 rows × 45 columns

```
filtered_df = aspect_df[aspect_df["topic"] != -1]
```

```
filtered_df.head()
```

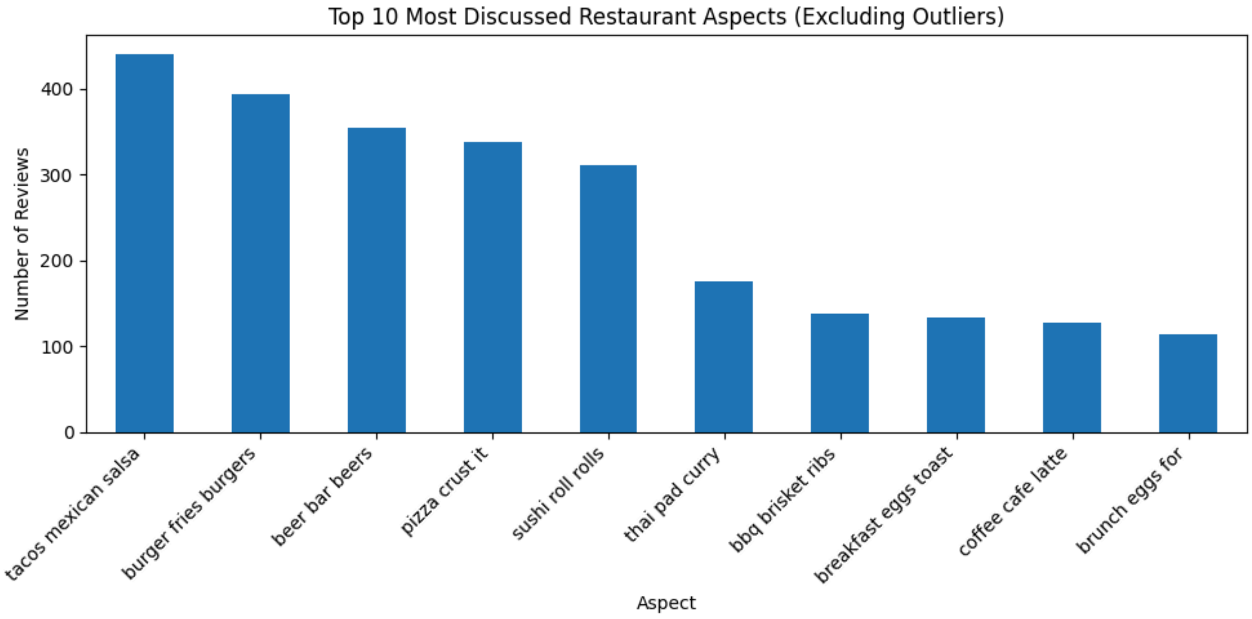
	business_id	review	topic	sentiment	AcceptsInsurance	AgesAllowed	Alcohol	Ambience	BY(
1	aY_n9RSaD2Yw09jSFFePew	We visited once and were very disappointed in ...	3	0	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknov
2	18eWJFJbXyR9j_5xfcRLYA	This is the first time I tried this place and ...	6	2	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknov
7	9OG5YkX1g2GReZM0AskizA	Great bar Happy Hour 4-7 every day. Wine & Dra...	1	2	Unknown	Unknown	'full_bar'	{u'divey': False, u'hipster': False, u'casual'...	Unknov
8	LnQRfj3pPz0369stRnwUWw	Very good sushi. The peanut avocado roll is a ...	4	2	Unknown	Unknown	u'beer_and_wine'	{u'divey': False, u'hipster': False, u'casual'...	Unknov
12	Mha77MN_BYGB-w7DONZ5YA	Our group of 6 decided to try something differ...	17	2	Unknown	Unknown	u'none'	{'romantic': False, 'intimate': False, 'touris...	Unknov

5 rows × 45 columns

```
import pickle

PATH = "/content/drive/MyDrive/restaurant_sentiment_model/filtered_df.pkl"

#Save
filtered_df.to_pickle(PATH)

#Later, load
filtered_df = pd.read_pickle(PATH)
```

Most Discussed Restaurant Aspects

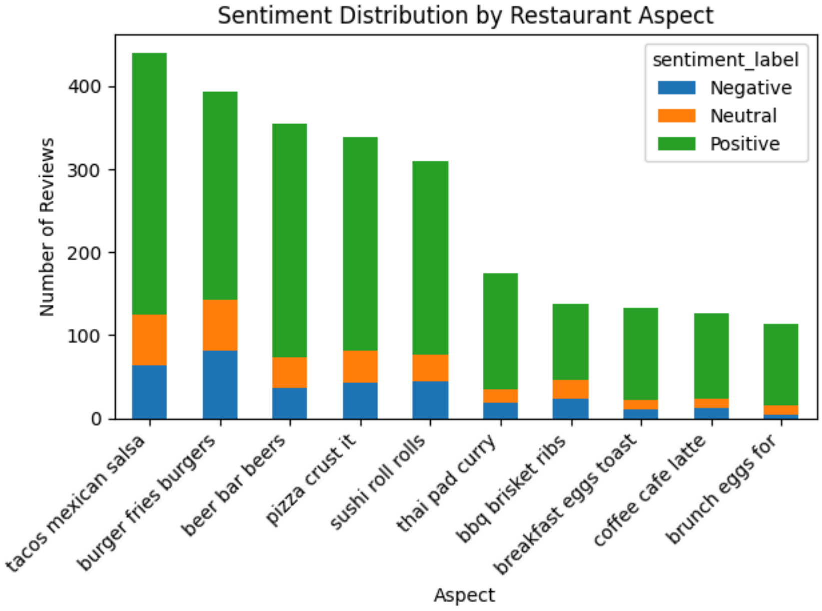
```
import matplotlib.pyplot as plt

topic_counts = (
    filtered_df["topic_label"]
    .value_counts()
    .head(10)
)

plt.figure(figsize=(10,5))
topic_counts.plot(kind="bar")
plt.title("Top 10 Most Discussed Restaurant Aspects (Excluding Outliers)")
plt.xlabel("Aspect")
plt.ylabel("Number of Reviews")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```



Sentiment Distribution by Restaurant Aspect

```
sentiment_topic = (
    aspect_df
    .groupby(["topic_label", "sentiment_label"])
    .size()
    .unstack(fill_value=0)
    .loc[topic_counts.index]
)

plt.figure(figsize=(10,5))
sentiment_topic.plot(kind="bar", stacked=True)
plt.title("Sentiment Distribution by Restaurant Aspect")
plt.xlabel("Aspect")
plt.ylabel("Number of Reviews")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```

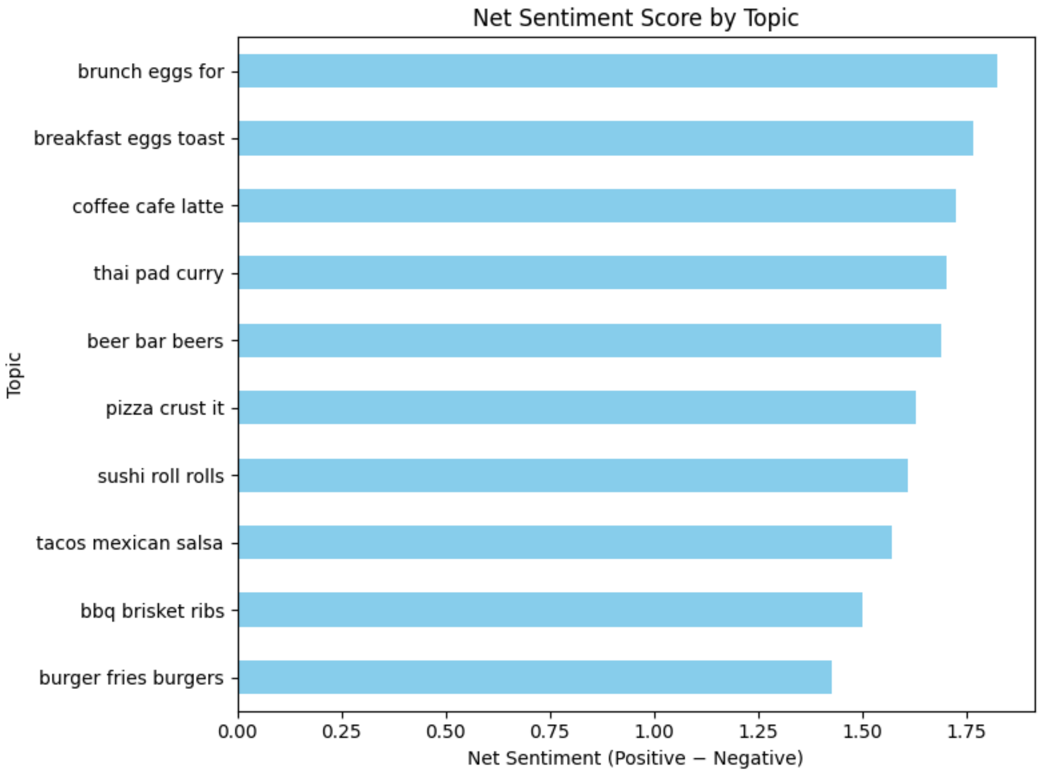
```
<Figure size 1000x500 with 0 Axes>
```



Net Sentiment of Restaurent topics

```
import matplotlib.pyplot as plt

#Get top 10 topics
```

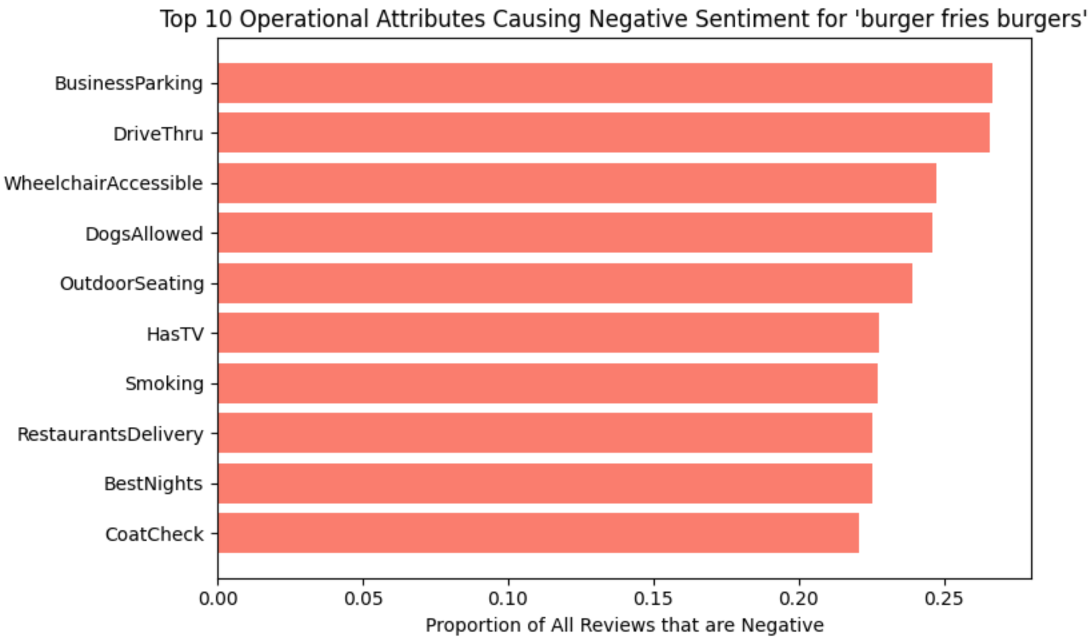
```
top_topics = filtered_df['topic_label'].value_counts().head(10).index

#Filter dataframe to only include top topics
filtered_top_df = filtered_df[filtered_df['topic_label'].isin(top_topics)]

#Calculate mean net sentiment by topic
net_sentiment = (
    filtered_top_df
    .groupby("topic_label")["sentiment"]
    .mean()
    .sort_values()
)

#Plot horizontal bar chart
plt.figure(figsize=(8,6))
net_sentiment.plot(kind="barh", color="skyblue")
plt.title("Net Sentiment Score by Topic")
plt.xlabel("Net Sentiment (Positive − Negative)")
plt.ylabel("Topic")
plt.tight_layout()
plt.show()
```



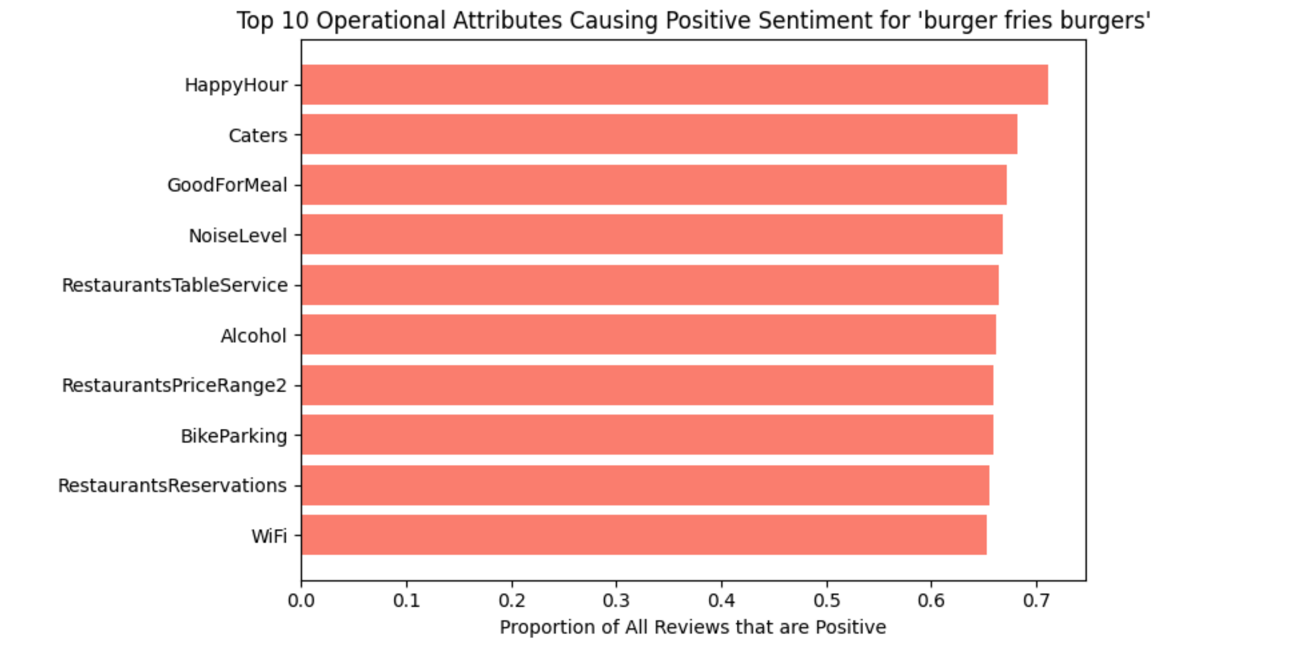
Deep-Dive Analysis: Aspect-Specific Drivers

```
import matplotlib.pyplot as plt

# Define topic
topic = "burger fries burgers"

# Filter for topic
topic_df = filtered_df[filtered_df["topic_label"] == topic]

# Automatically detect operational attributes
exclude_cols = ["business_id", "review", "topic", "topic_label", "sentiment", "sentiment_label"]
operational_attributes = [col for col in topic_df.columns if col not in exclude_cols]

# Compute proportion of negative reviews for each attribute relative to all reviews
neg_attr_scores = {}

for attr in operational_attributes:
    if topic_df[attr].dtype == 'object':
        # Get all reviews for this attribute value
        counts = topic_df.groupby(attr)["sentiment_label"].value_counts(normalize=True)
        # Take the proportion of negative sentiment for the most common attribute value
        most_common_value = topic_df[attr].mode()[0]
        neg_score = counts.get((most_common_value, "Negative"), 0)
        neg_attr_scores[attr] = neg_score
```

```
# Sort descending and take top 5
top5_neg_attrs = sorted(neg_attr_scores.items(), key=lambda x: x[1], reverse=True)[:10]

# Plot
attrs, scores = zip(*top5_neg_attrs)

plt.figure(figsize=(8,5))
plt.barh(attrs, scores, color='salmon')
plt.xlabel("Proportion of All Reviews that are Negative")
plt.title(f"Top 10 Operational Attributes Causing Negative Sentiment for '{topic}'")
plt.gca().invert_yaxis()  # highest at top
plt.tight_layout()
plt.show()
```



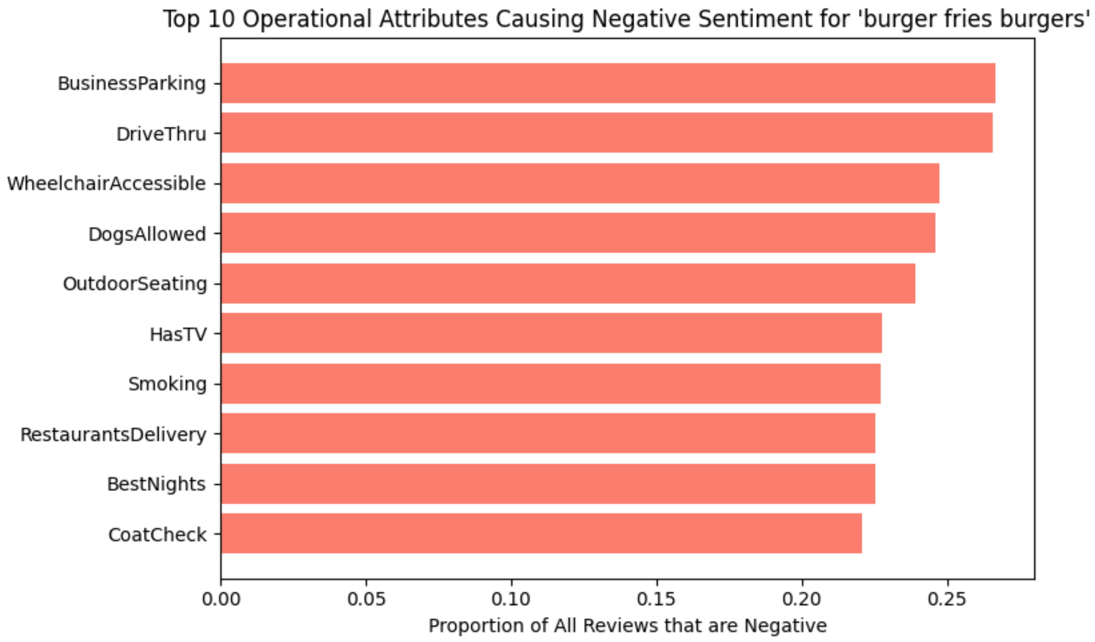
```
import matplotlib.pyplot as plt

# Define topic
topic = "burger fries burgers"

# Filter for topic
topic_df = filtered_df[filtered_df["topic_label"] == topic]

# Automatically detect operational attributes
exclude_cols = ["business_id", "review", "topic", "topic_label", "sentiment", "sentiment_label"]
operational_attributes = [col for col in topic_df.columns if col not in exclude_cols]

# Compute proportion of positive reviews for each attribute relative to all reviews
pos_attr_scores = {}

for attr in operational_attributes:
    if topic_df[attr].dtype == 'object':
        # Get all reviews for this attribute value
        counts = topic_df.groupby(attr)["sentiment_label"].value_counts(normalize=True)
        # Take the proportion of positive sentiment for the most common attribute value
        most_common_value = topic_df[attr].mode()[0]
        pos_score = counts.get((most_common_value, "Positive"), 0)
        pos_attr_scores[attr] = pos_score

# Sort descending and take top 5
top5_pos_attrs = sorted(pos_attr_scores.items(), key=lambda x: x[1], reverse=True)[:10]

# Plot
attrs, scores = zip(*top5_pos_attrs)

plt.figure(figsize=(8,5))
plt.barh(attrs, scores, color='salmon')
plt.xlabel("Proportion of All Reviews that are Positive")
plt.title(f"Top 10 Operational Attributes Causing Positive Sentiment for '{topic}'")
plt.gca().invert_yaxis()  # highest at top
plt.tight_layout()
plt.show()
```



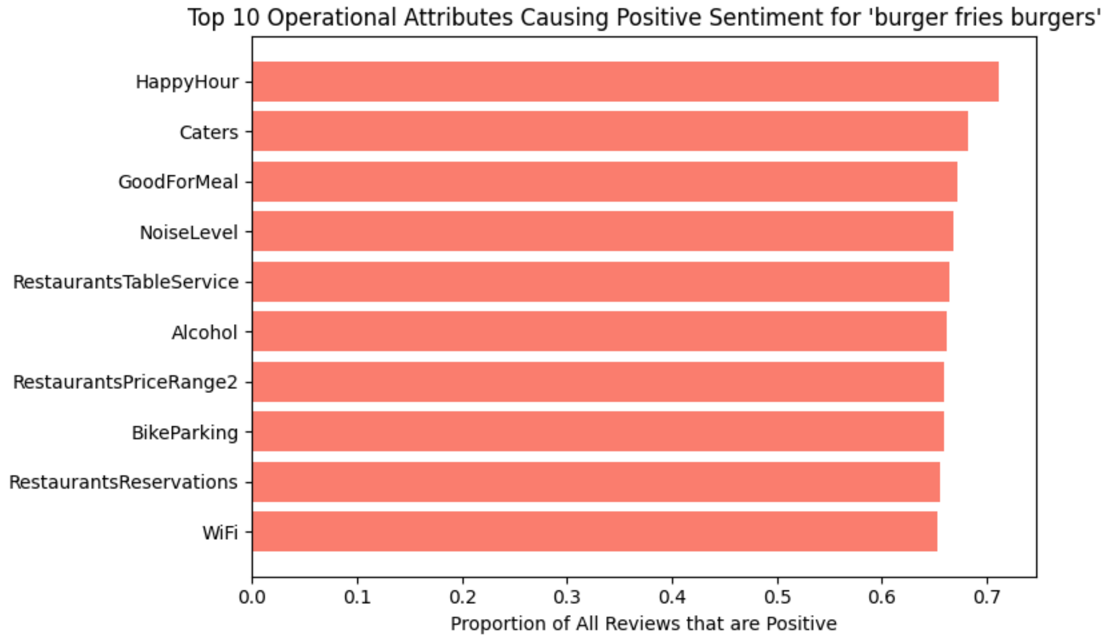
```
import matplotlib.pyplot as plt

def plot_top_negative_attributes(filtered_df, topic):
    # Define topic
    topic = topic

    # Filter for topic
    topic_df = filtered_df[filtered_df["topic_label"] == topic]

    # Automatically detect operational attributes
    exclude_cols = ["business_id", "review", "topic", "topic_label", "sentiment", "sentiment_label"]
    operational_attributes = [col for col in topic_df.columns if col not in exclude_cols]

    # Compute proportion of negative reviews for each attribute relative to all reviews
    neg_attr_scores = {}

    for attr in operational_attributes:
        if topic_df[attr].dtype == 'object':
            # Get all reviews for this attribute value
            counts = topic_df.groupby(attr)["sentiment_label"].value_counts(normalize=True)
            # Take the proportion of negative sentiment for the most common attribute value
            most_common_value = topic_df[attr].mode()[0]
            neg_score = counts.get((most_common_value, "Negative"), 0)
            neg_attr_scores[attr] = neg_score

    # Sort descending and take top 5
    top5_neg_attrs = sorted(neg_attr_scores.items(), key=lambda x: x[1], reverse=True)[:10]

    # Plot
    attrs, scores = zip(*top5_neg_attrs)

    plt.figure(figsize=(8,5))
    plt.barh(attrs, scores, color='salmon')
    plt.xlabel("Proportion of All Reviews that are Negative")
    plt.title(f"Top 10 Operational Attributes Causing Negative Sentiment for '{topic}'")
    plt.gca().invert_yaxis()  # highest at top
    plt.tight_layout()
    plt.show()
```

```
import matplotlib.pyplot as plt

def plot_top_positive_attributes(filtered_df, topic):
    # Define topic
    topic = topic

    # Filter for topic
    topic_df = filtered_df[filtered_df["topic_label"] == topic]

    # Automatically detect operational attributes
    exclude_cols = ["business_id", "review", "topic", "topic_label", "sentiment", "sentiment_label"]
    operational_attributes = [col for col in topic_df.columns if col not in exclude_cols]

    # Compute proportion of positive reviews for each attribute relative to all reviews
    pos_attr_scores = {}

    for attr in operational_attributes:
        if topic_df[attr].dtype == 'object':
            # Get all reviews for this attribute value
            counts = topic_df.groupby(attr)["sentiment_label"].value_counts(normalize=True)
            # Take the proportion of positive sentiment for the most common attribute value
            most_common_value = topic_df[attr].mode()[0]
            pos_score = counts.get((most_common_value, "Positive"), 0)
            pos_attr_scores[attr] = pos_score

    # Sort descending and take top 5
    top5_pos_attrs = sorted(pos_attr_scores.items(), key=lambda x: x[1], reverse=True)[:10]

    # Plot
    attrs, scores = zip(*top5_pos_attrs)

    plt.figure(figsize=(8,5))
    plt.barh(attrs, scores, color='salmon')
    plt.xlabel("Proportion of All Reviews that are Positive")
    plt.title(f"Top 10 Operational Attributes Causing Positive Sentiment for '{topic}'")
    plt.gca().invert_yaxis()  # highest at top
    plt.tight_layout()
    plt.show()
```

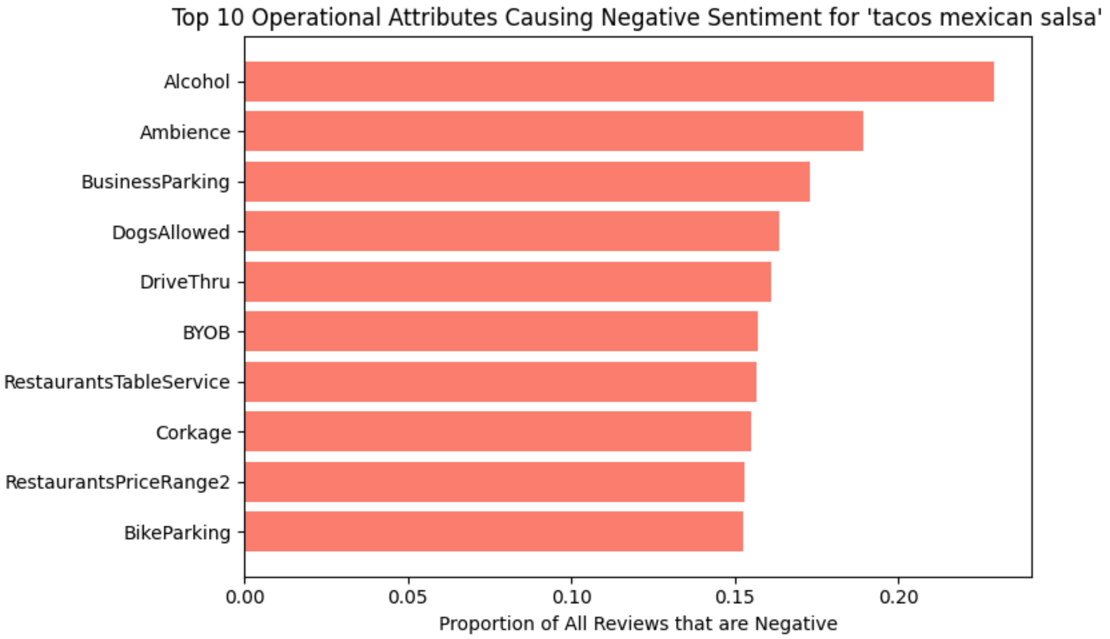
```
plot_top_negative_attributes(
    filtered_df,
    topic="burger fries burgers"
)
```

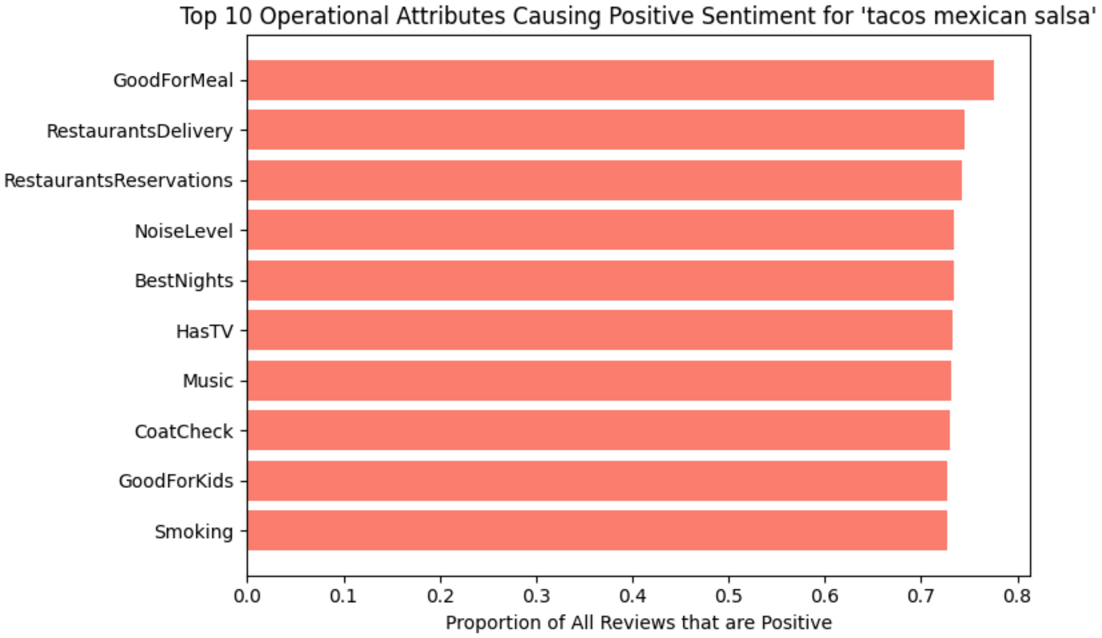


```
plot_top_positive_attributes(
    filtered_df,
    topic="burger fries burgers"
)
```



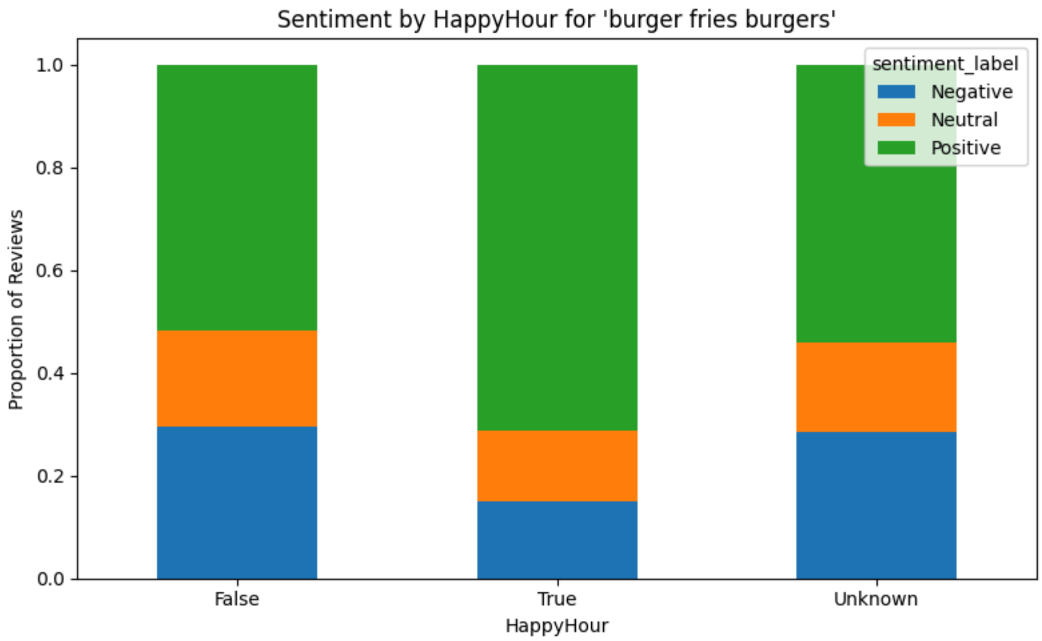
```
plot_top_negative_attributes(
    filtered_df,
    topic = "tacos mexican salsa"
)

plot_top_positive_attributes(
    filtered_df,
    topic = "tacos mexican salsa"
)
```





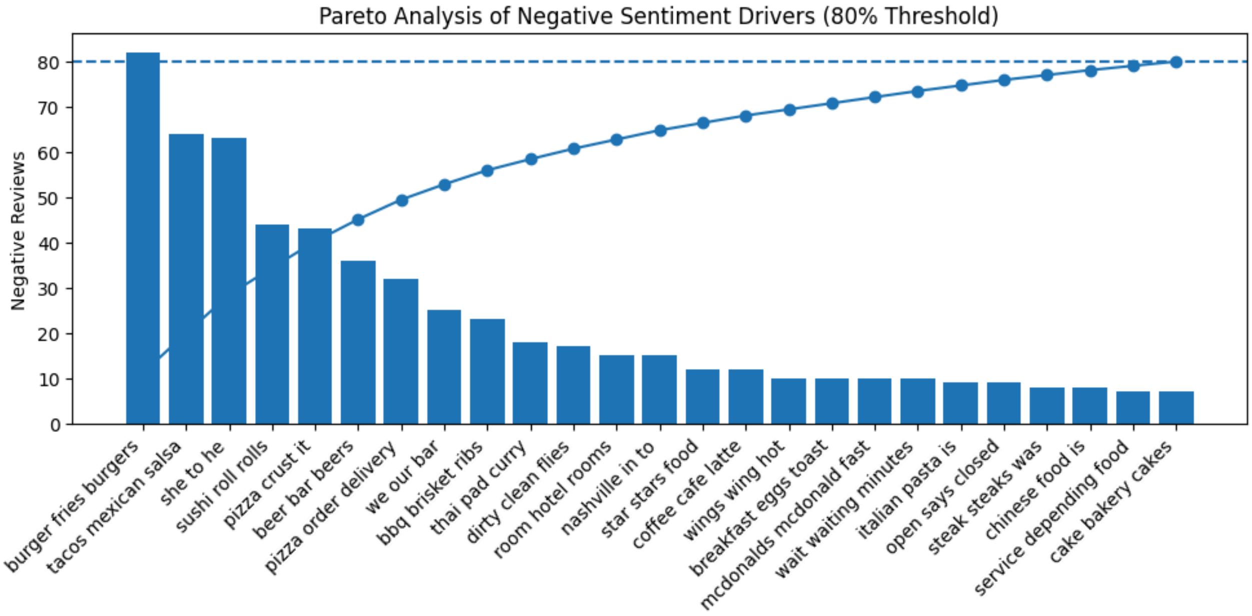
Operational Attribute Impact Analysis

```
topic = "burger fries burgers"
attribute = "HappyHour"

topic_df = filtered_df[filtered_df["topic_label"] == topic]

sentiment_dist = (
    topic_df
    .groupby([attribute, "sentiment_label"])
    .size()
    .unstack(fill_value=0)
)

sentiment_dist = sentiment_dist.div(sentiment_dist.sum(axis=1), axis=0)

sentiment_dist.plot(
    kind="bar",
    stacked=True,
    figsize=(8,5)
)

plt.title(f"Sentiment by {attribute} for '{topic}'")
plt.ylabel("Proportion of Reviews")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



Pareto Analysis of Negative Sentiment Drivers

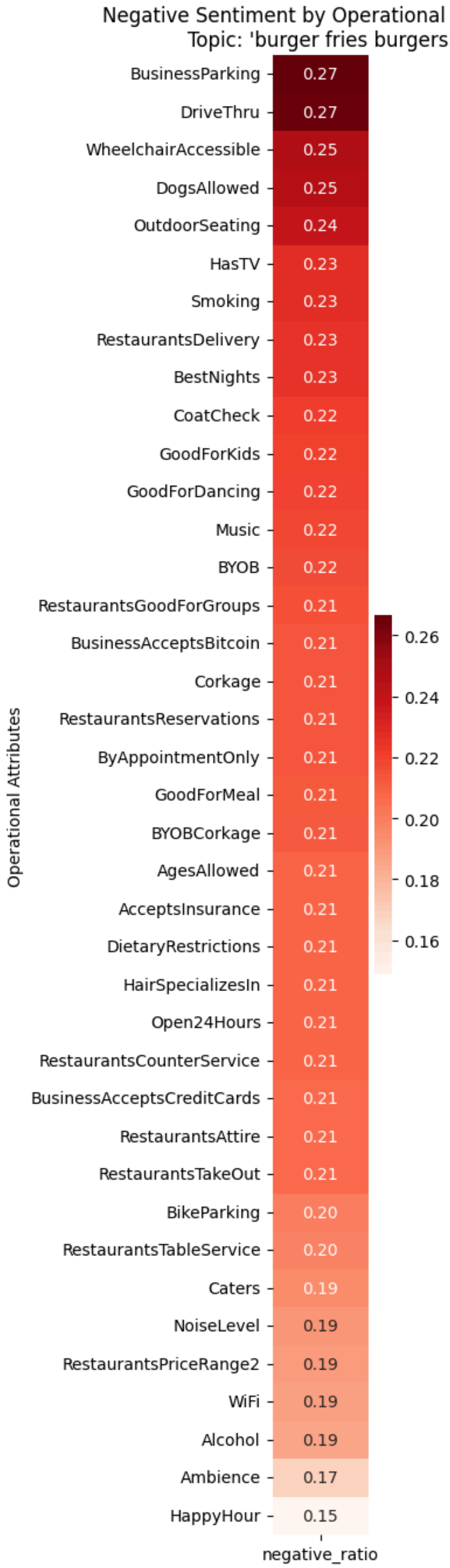
```
import matplotlib.pyplot as plt

#Count negative reviews per topic
neg_counts = (
    filtered_df[filtered_df["sentiment_label"] == "Negative"]
    ["topic_label"]
    .value_counts()
)

#Compute cumulative percentage
cum_pct = neg_counts.cumsum() / neg_counts.sum() * 100

#Keep only topics up to 80% cumulative impact
pareto_df = (
    cum_pct[cum_pct <= 80]
    .index
)

neg_counts_80 = neg_counts.loc[pareto_df]
cum_pct_80 = cum_pct.loc[pareto_df]

# Plot
plt.figure(figsize=(10,5))

plt.bar(neg_counts_80.index, neg_counts_80.values)
plt.plot(
plt.plot(
    neg_counts_80.index,
    cum_pct_80.values,
    marker="o"
)

plt.axhline(80, linestyle="--")
plt.ylabel("Negative Reviews")
plt.title("Pareto Analysis of Negative Sentiment Drivers (80% Threshold)")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```



Heatmap Analysis of Operational Drivers

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

def plot_single_topic_attribute_heatmap(filtered_df, topic):
    #Filter for topic
    topic_df = filtered_df[filtered_df["topic_label"] == topic]

    #Exclude non-operational columns
    exclude_cols = ["business_id", "review", "topic", "topic_label", "sentiment", "sentiment_label"]
    attributes = [c for c in topic_df.columns if c not in exclude_cols]

    heatmap_data = []

    for attr in attributes:
        if topic_df[attr].dtype == "object":
            counts = (
                topic_df
                .groupby(attr)["sentiment_label"]
                .value_counts(normalize=True)
            )
            most_common_value = topic_df[attr].mode()[0]
            neg_score = counts.get((most_common_value, "Negative"), 0)

            heatmap_data.append({
                "attribute": attr,
                "negative_ratio": neg_score
            })

    heatmap_df = (
        pd.DataFrame(heatmap_data)
        .set_index("attribute")
        .sort_values("negative_ratio", ascending=False)
    )

    plt.figure(figsize=(4, max(6, len(heatmap_df) * 0.35)))
    sns.heatmap(
        heatmap_df,
        cmap="Reds",
        annot=True,
        fmt=".2f",
        cbar=True
    )

    plt.title(f"Negative Sentiment by Operational Attribute\nTopic: '{topic}'")
    plt.xlabel("")
    plt.ylabel("Operational Attributes")
    plt.tight_layout()
    plt.show()
```

```
plot_single_topic_attribute_heatmap(
    filtered_df,
    topic="burger fries burgers"
)
```



```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

def plot_single_topic_sentiment_heatmap(filtered_df, topic):
    #Filter for topic
    topic_df = filtered_df[filtered_df["topic_label"] == topic]

    #Exclude non-operational columns
```

```
exclude_cols = ["business_id", "review", "topic", "topic_label", "sentiment", "sentiment_label"]
attributes = [c for c in topic_df.columns if c not in exclude_cols]

heatmap_data = []

for attr in attributes:
    if topic_df[attr].dtype == "object":
        counts = (
            topic_df
            .groupby(attr)["sentiment_label"]
            .value_counts(normalize=True)
        )

        most_common_value = topic_df[attr].mode()[0]

        neg_ratio = counts.get((most_common_value, "Negative"), 0)
        pos_ratio = counts.get((most_common_value, "Positive"), 0)

        #Net sentiment score: positive - negative
        net_score = pos_ratio - neg_ratio

        heatmap_data.append({
            "attribute": attr,
            "net_sentiment": net_score
        })

heatmap_df = (
    pd.DataFrame(heatmap_data)
    .set_index("attribute")
    .sort_values("net_sentiment", ascending=True)
)

plt.figure(figsize=(4, max(6, len(heatmap_df) * 0.35)))
sns.heatmap(
    heatmap_df,
    cmap="RdYlGn",
    center=0,
    annot=True,
    fmt=".2f",
    cbar_kws={"label": "Net Sentiment (Positive – Negative)"}
```