# Sentiment Analysis of Coffee Shop Reviews

**Author:** Benjamin Heindl

**Course:** IST 664

**Date:** November 13, 2023

In [1]:
```
# Import necessary libraries
import pandas as pd
from transformers import pipeline
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from collections import Counter
import nltk
from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.tokenize import word_tokenize
from collections import Counter
from wordcloud import WordCloud, get_single_color_func
import matplotlib.pyplot as plt
```

## Data Preprocessing

Define functions for preprocessing our text data. This includes:

- converting text to lowercase
- removing punctuation
- tokenizing
- removing stopwords
- lemmatizing the words

In [2]:
```
# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Function to map NLTK's POS tags to the format expected by the lemmatizer
def get_wordnet_pos(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

# Preprocessing function
def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(t, get_wordnet_pos(t)) for t in tokens if t not in stopwords.words('english')]
    return tokens
```

## Loading and Preprocessing the Reviews

In [3]:
```
# Load the reviews dataset
file_path = '/Users/benjheindl/Desktop/Fall 2023/IST 664 Natural Language Processing/NLP Presentation/coffee_shop_reviews.csv'
reviews_df = pd.read_csv(file_path)

# Apply preprocessing to your reviews
reviews_df['Processed_Review'] = reviews_df['Review'].apply(lambda x: ' '.join(preprocess_text(x)))
```

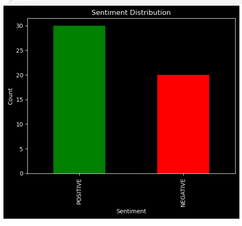
## Sentiment Analysis using BERT

In [4]:
```
# Using Hugging Face Transformers library
# Employing pre-trained BERT model for sentiment analysis
# Model classifies each review as either positive or negative

# Load the sentiment-analysis pipeline
sentiment_pipeline = pipeline('sentiment-analysis')

# Apply sentiment analysis
reviews_df['Sentiment'] = reviews_df['Processed_Review'].apply(lambda x: sentiment_pipeline(x)[0]['label'])
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english and revision af0f99b (https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
```

## Visualizing the Sentiment Distribution

In [5]:
```
# Calculate and visualize sentiment distribution
# Provides overview of the general sentiment in the reviews

# Set the style of matplotlib to 'dark_background'
plt.style.use('dark_background')

# Calculate and visualize sentiment distribution
sentiment_counts = reviews_df['Sentiment'].value_counts()

# Create a bar chart with custom colors for positive and negative sentiments
colors = ['green' if sentiment == 'POSITIVE' else 'red' for sentiment in sentiment_counts.index]
sentiment_counts.plot(kind='bar', color=colors)

# Customize the plot with titles and labels
plt.title('Sentiment Distribution', color='white')
plt.xlabel('Sentiment', color='white')
plt.ylabel('Count', color='white')

# Show the plot
plt.show()
```




## Detailed Sentiment Analysis

In [6]:
```
# Modify the sentiment analysis to include scores
# Provide a measure of how positive or negative a review is

reviews_df['Sentiment_Details'] = reviews_df['Processed_Review'].apply(sentiment_pipeline)
reviews_df['Sentiment'] = reviews_df['Sentiment_Details'].apply(lambda x: x[0]['label'])
reviews_df['Sentiment_Score'] = reviews_df['Sentiment_Details'].apply(lambda x: x[0]['score'])
```

## Displaying Top Reviews

In [7]:
```
# Helps understand the kind of language and content that contributes to
# strong positive or negative sentiments.

# Also using as checkpoint

# Sort and display the top 5 positive reviews
top_positive_reviews = reviews_df[reviews_df['Sentiment'] == 'POSITIVE'].sort_values(by='Sentiment_Score', ascending=False).head(5)
print("Top 5 Positive Reviews:")
print(top_positive_reviews[['Review', 'Sentiment_Score']], "\n")

# Sort and display the top 5 negative reviews
top_negative_reviews = reviews_df[reviews_df['Sentiment'] == 'NEGATIVE'].sort_values(by='Sentiment_Score', ascending=False).head(5)
print("Top 5 Negative Reviews:")
print(top_negative_reviews[['Review', 'Sentiment_Score']])
```

```
Top 5 Positive Reviews:
                                               Review  Sentiment_Score
20  Such a vibrant local spot with amazing coffee ...         0.999824
23  Their loyalty program is fantastic - lots of p...         0.999805
4   Friendly staff and a warm atmosphere make for ...         0.999801
43  The staff seemed preoccupied and not very welc...         0.999782
13  They always remember my order; it's nice to ha...         0.999772

Top 5 Negative Reviews:
                                               Review  Sentiment_Score
29  Service was slow; it seemed like they were und...         0.999678
41  There's a lack of power outlets for laptops, w...         0.999509
25  The coffee was lukewarm, which was a bit disap...         0.999297
33  The coffee shop's interior is a bit outdated a...         0.999201
47  The pastry I had was stale, as if it wasn't fr...         0.999842
```

## Extracting Significant Words

In [8]:
```
sia = SentimentIntensityAnalyzer()

# Function to score words based on sentiment and return top 20 significant ones
def get_top_significant_words(reviews, top_n=20):
    positive_word_scores = Counter()
    negative_word_scores = Counter()

    for review in reviews:
        for word in word_tokenize(review):
            sentiment_score = sia.polarity_scores(word)['compound']

            if sentiment_score > 0:
                positive_word_scores[word] += sentiment_score
            elif sentiment_score < 0:
                negative_word_scores[word] += abs(sentiment_score)

    # Select top 20 positive and negative words
    top_positive_words = {word: score for word, score in positive_word_scores.most_common(top_n)}
    top_negative_words = {word: score for word, score in negative_word_scores.most_common(top_n)}

    return top_positive_words, top_negative_words

# Extracting top 20 significant words for positive and negative reviews
positive_reviews = reviews_df[reviews_df['Sentiment'] == 'POSITIVE']['Processed_Review']
negative_reviews = reviews_df[reviews_df['Sentiment'] == 'NEGATIVE']['Processed_Review']

# Concatenate positive and negative reviews into a single list of reviews
all_reviews = positive_reviews.tolist() + negative_reviews.tolist()

# Call the function with the concatenated list of reviews
top_positive_words, top_negative_words = get_top_significant_words(all_reviews)

# Print or save the lists
print("Top 20 Significant Positive Words:", top_positive_words)
print("Top 20 Significant Negative Words:", top_negative_words)
```

```
Top 20 Significant Positive Words: {'love': 1.9187, 'perfect': 1.7157, 'strong': 1.5318, 'favorite': 1.3763999999999998, 'loyalty': 1.0846, 'enjoy': 0.9878, 'treat': 0.8838, 'comfort': 0.7224, 'like': 0.7224, 'best': 0.6369, 'great': 0.6249, 'delightful': 0.5859, 'reward': 0.5719, 'fantastic': 0.5574, 'amaze': 0.5423, 'vibrant': 0.5267, 'impressive': 0.5106, 'comfortable': 0.5106, 'friendly': 0.4939, 'hand': 0.4939}
Top 20 Significant Negative Words: {'wrong': 0.4767, 'bitter': 0.4215, 'disappoint': 0.4019, 'uncomfortably': 0.4019, 'inconvenient': 0.34, 'lack': 0.3182, 'forgotten': 0.2263, 'noisy': 0.1779, 'miss': 0.1531, 'hard': 0.1027}
```

## Generating Sentiment-Scored Word Clouds

In [9]:
```
# Custom color function class
class SimpleGroupedColorFunc(object):
    def __init__(self, color_to_words, default_color):
        self.color_to_words = color_to_words
        self.default_color = default_color

    def get_color_func(self, word, **kwargs):
        for color, words in self.color_to_words.items():
            if word in words:
                return get_single_color_func(color)(word, **kwargs)
        return get_single_color_func(self.default_color)(word, **kwargs)

# Initialize VADER sentiment analyzer
# score the words in the reviews based on their sentiment
sia = SentimentIntensityAnalyzer()

# Define color schemes
positive_color_scheme = SimpleGroupedColorFunc({}, 'green')
negative_color_scheme = SimpleGroupedColorFunc({}, 'darkred')

# Score words based on sentiment
def score_words(reviews, sentiment_threshold):
    word_scores = Counter()
    for review in reviews:
        for word in word_tokenize(review):
            sentiment_score = sia.polarity_scores(word)['compound']
            if (sentiment_threshold > 0 and sentiment_score > sentiment_threshold) or (sentiment_threshold < 0 and sentiment_score < sentiment_threshold):
                word_scores[word] += sentiment_score
    return word_scores

# Generate word clouds to visually represent the most
# significant words in both positive and negative reviews
def generate_custom_wordcloud(dataframe, sentiment, title, color_scheme, sentiment_threshold):
    reviews = dataframe[dataframe['Sentiment'] == sentiment]['Processed_Review']
    sentiment_scored_words = score_words(reviews, sentiment_threshold)
    significant_words = {word: score for word, score in sentiment_scored_words.items() if abs(score) > 0.1}

    wordcloud = WordCloud(width=800, height=400, background_color='black', color_func=color_scheme.get_color_func).generate_from_frequencies(significant_words)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title)
    plt.axis("off")
    plt.show()

# Display word clouds
generate_custom_wordcloud(reviews_df, 'POSITIVE', 'Positive Sentiment Words', positive_color_scheme, 0.1)
generate_custom_wordcloud(reviews_df, 'NEGATIVE', 'Negative Sentiment Words', negative_color_scheme, -0.1)
```



### Positive Words

- **Emotionally Positive**: Words like "love", "perfect", "enjoy", "appreciate", "comfort", "favorite", "excite", "amaze", and "fantastic" are strongly positive and reflect a high level of satisfaction and enjoyment.
- **Quality and Service**: "best", "impressive", "delightful", "friendly", "great", "commitment", and "welcome" indicate a high regard for the quality of the coffee and the service provided.
- **Ambiance and Experience**: "warm", "comfortable", "relax", "special", "vibrant", and "strong" suggest a positive atmosphere and overall experience at the coffee shop.
- **Loyalty and Rewards**: The presence of "loyalty" and "reward" could be tied to customer loyalty programs or the feeling of being rewarded with a good coffee experience.

### Negative Words

- **Dissatisfaction and Issues**: "disappoint", "bitter", "wrong", "hard", "inconvenient", and "uncomfortably" clearly indicate areas of dissatisfaction.
- **Service and Quality Concerns**: Words like "clean" and "fresh" might be highlighting concerns about cleanliness or freshness of products. "Lack" could refer to a lack of something desired.

### Observations

- The significant words align well with the sentiments we're trying to capture in each word cloud.
- The positive word cloud captures a wide range of positive experiences and attributes, from service quality to the ambiance.