Project Name: Restaurant Sentiment Analysis

Author: Hiranmayi Palanki (palanki2@illinois.edu)

Team: Solo, worked independently on the course project.

Link to the Course Project Presentation Recording:

https://github.com/palankihiran/CourseProject/blob/main/Course%20Project%20Tutorial%20Presentation.pptx

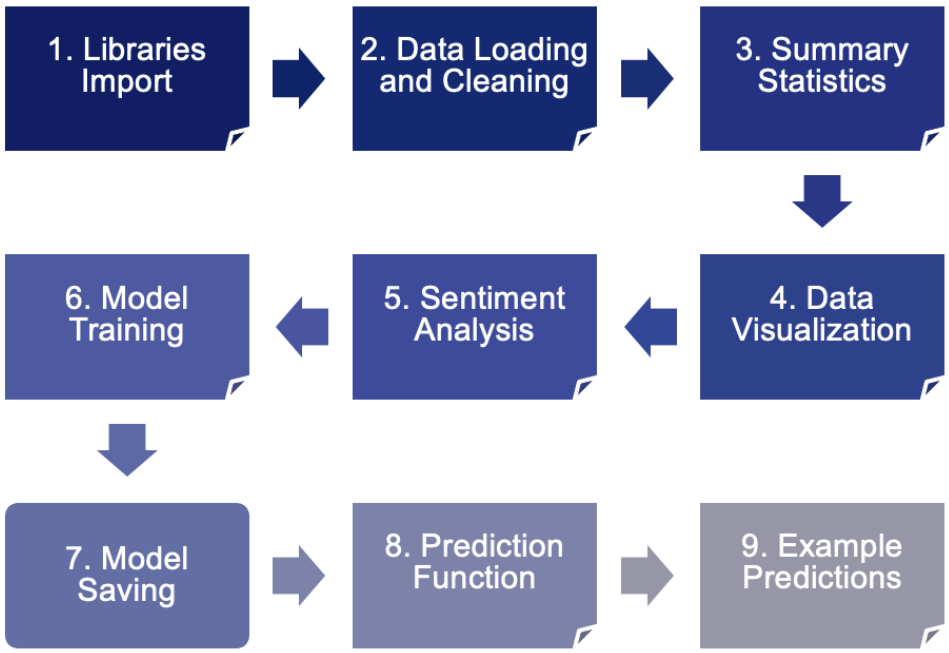
I. **Introduction**: The course project report provides insight into the project overview, purpose, software functionality, software implementation, future improvements, software usage, contribution of the team member and conclusion.

II. **Project Overview**: The objective is to build a Restaurant Review Sentiment Analysis system using a Machine Learning Model. It involves sentiment analysis on restaurant reviews from two different sources (Google and TripAdvisor). We want to predict the sentiment of a given restaurant review as either Negative, Neutral, or Positive, based on the text of the review.

III. **Project Purpose**: The main purpose is to complement the traditional rating system used for restaurant reviews with sentiment analysis. The key reasons for integrating sentiment analysis alongside a rating system are:

- Deeper Insights: Sentiment analysis provides a more detailed understanding of customer feedback by categorizing reviews as positive, negative, or neutral. This allows for deeper insights into the reasons behind the ratings.
- Filtering Ambiguity: Rating systems can be misleading when a customer gives a low score due to disliking one aspect of the service but liking other aspects. Sentiment analysis helps filter such cases more effectively.

IV. **Overview of the Software Functionality**: The overall functionality of the software is summarized via a nine-step process as depicted in the diagram below.



1. **Libraries Import**

   - Pandas: For reading files and handling data.
   - NumPy: For data manipulation and related operations.
   - Matplotlib: For plotting graphs and visualization.
   - Seaborn: For better control over visualization.
   - Warnings: To filter out warning messages.

2. **Data Loading and Cleaning**

   - Two datasets are loaded from CSV files, one from Google Reviews and the other from TripAdvisor.
   - Missing values are checked and dropped.

3. **Summary Statistics**

   - Descriptive statistics are calculated for the 'Rating' column in both datasets.

4. **Data Visualization**

   - Several visualizations are created using Matplotlib and Seaborn to understand the distribution of ratings, analyze ratings by location, and generate word clouds for reviews.

5. **Sentiment Analysis**

   - The code prepares the data for sentiment analysis by merging and filtering relevant columns.
   - Reviews are categorized into negative, neutral, or positive based on predefined bins and labels.
   - Dataset is split into training and testing sets.

6. **Model Training**

   - The sentiment analysis model is trained using a Multinomial Logistic Regression algorithm.
   - TF-IDF vectorization is applied to convert text data into numerical features.
   - Model performance is evaluated on both the training and testing sets.

7. **Model Saving**

   - The trained TF-IDF vectorizer and sentiment analysis model are saved using the pickle library.

8. **Prediction Function**

- A function (predict_rating_category) is defined to predict the sentiment category of a user input review.
- The function uses the trained model and TF-IDF vectorizer.

9. Example Predictions

- Three example reviews are provided, and their predicted sentiment categories along with probabilities are displayed.

Overall, the code implements the process of building a sentiment analysis model for restaurant reviews, including data preparation, visualization, model training, and prediction. The final trained model and vectorizer are saved for future use, and a function is defined for making predictions on new reviews.

V. Functionality of Code: The overall functionality of the code in terms of what it does and what it can be used for is elaborated in the section below.

1. Libraries Import

Purpose: Importing necessary Python libraries for data handling, visualization, and machine learning.

Use:
- Pandas: Used for reading and manipulating datasets.
- NumPy: Essential for numerical operations and array handling.
- Matplotlib and Seaborn: Used for creating data visualizations.
- Warnings: Used to filter out warning messages.

2. Data Loading and Cleaning

Purpose: Loading restaurant review datasets from Google Reviews and TripAdvisor and cleaning the data.

Use:
- Pandas: Reading CSV files, handling data frames, and dropping missing values.

3. Summary Statistics

Purpose: Computing descriptive statistics for the 'Rating' column in both datasets.

Use:
- Understanding the distribution of ratings in the datasets.

4. Data Visualization

Purpose: Creating visualizations to understand the characteristics of the datasets.

Use:

- Matplotlib and Seaborn: Generating count plots, box plots, and word clouds to visualize the distribution of ratings, analyze ratings by location, and display frequently used words in reviews.

5. Sentiment Analysis

Purpose: Preparing data for sentiment analysis, merging datasets, and categorizing reviews into negative, neutral, or positive sentiments.

Use:

- Data preprocessing and transformation for machine learning.

Splitting the dataset into training and testing sets.

6. Model Training

Purpose: Training a sentiment analysis model using a Multinomial Logistic Regression algorithm.
Use:

- sklearn library for TF-IDF vectorization, model training, and evaluation.
- TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical features.
- Multinomial Logistic Regression for sentiment classification.

7. Model Saving

Purpose: Saving the trained TF-IDF vectorizer and sentiment analysis model for future use.

Use:

- pickle library for serializing Python objects.

8. Prediction Function

Purpose: Defining a function to predict the sentiment category of a user-input restaurant review.

Use:

- Allows for real-world application of the trained model on new data.

9. Example Predictions

Purpose: Demonstrating the use of the trained model for predicting sentiment categories and probabilities for given restaurant reviews.
Use:

- Provides a practical example of applying the trained model to make predictions on new reviews.

Overall Use

- Building a sentiment analysis model for restaurant reviews to categorize them as negative, neutral, or positive.

Potential Applications

- Understanding customer sentiments towards restaurants.
- Identifying areas for improvement based on detailed sentiment analysis.
- Providing insights beyond traditional rating systems.
- Real-world application for predicting sentiment categories of new reviews.

This code essentially serves as a foundation for a restaurant review sentiment analysis project, offering insights and predictions based on machine learning techniques.

VI. Software Implementation: The section below documents how the software is implemented with sufficient detail so that others can have a basic understanding of the code for future extension or any further improvement.

1. Libraries Import and Configuration

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

This code imports essential libraries for data manipulation, analysis, and visualization.

2. Loading and Reading Data

```
DATASET_PATH = './data_cleaned/'

# Read the csv file
google_df = pd.read_csv(f'{DATASET_PATH}/GoogleReview_data_cleaned.csv')
tripadvisor_df = pd.read_csv(f'{DATASET_PATH}/TripAdvisor_data_cleaned.csv')
```

This code loads restaurant review data from two sources, Google, and TripAdvisor, which have been previously cleaned and stored in CSV files.

3. Viewing Sample Rows

```
google_df.sample(5) # View Sample rows
```

```
tripadvisor_df.sample(5) # View Sample rows
```

This code displays a random sample of 5 rows from the Google and TripAdvisor reviews DataFrame.

4. Checking for Missing Values

```
# Check for missing values
print('Trip Advisor\n')
print(tripadvisor_df.isnull().sum())
print('\nGoogle Review\n')
print(google_df.isnull().sum())
```

This code identifies and displays the count of missing values in the TripAdvisor and Google Review datasets. Detecting missing values is crucial for data preprocessing, ensuring data quality, and deciding on strategies for handling missing information.

5. Data Cleaning

```
# Data cleaning
tripadvisor_df.dropna(inplace=True)
google_df.dropna(inplace=True)
```

This code cleans the TripAdvisor and Google Review datasets by removing rows with missing values. Data cleaning is an essential step to ensure the quality and reliability of the dataset before further analysis or modeling.

6. Summary Statistics

```
# Summary statistics for TripAdvisor
tripadvisor_df.describe()
```

```
# Summary statistics for Google Review
google_df.describe()
```

The code above displays summary statistics for the TripAdvisor and Google dataset. Summary statistics provide a quick overview of the distribution and characteristics of numerical variables in the dataset.

7. Distribution of Ratings on TripAdvisor and Google Reviews

```
# Distribution of Ratings for TripAdvisor
plt.figure(figsize=(8, 6))
sns.countplot(x='Rating', data=tripadvisor_df)
plt.title('Distribution of Ratings on TripAdvisor')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```

The code above creates a bar plot with the x-axis representing different ratings and the y-axis indicating the count of reviews for each rating. This visualization helps to understand the overall sentiment of reviews in the TripAdvisor dataset.

```
# Distribution of Ratings for Google Review
plt.figure(figsize=(8, 6))
sns.countplot(x='Rating', data=google_df)
plt.title('Distribution of Ratings on Google Review')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```

The code above generates a count plot to visualize the distribution of ratings in the Google Review dataset. It provides insights into the sentiment distribution based on user ratings.

8. Distribution of ratings by Location on TripAdvisor and Google Reviews

```
# Distribution of Ratings by Location for TripAdvisor
plt.figure(figsize=(12, 6))
sns.boxplot(x='Location', y='Rating', data=tripadvisor_df)
plt.title('Distribution of Ratings by Location on TripAdvisor')
plt.xlabel('Location')
plt.ylabel('Rating')
plt.xticks(rotation=45)
plt.show()
```

```
# Distribution of Ratings by Location for GoogleReviews
plt.figure(figsize=(12, 6))
sns.boxplot(x='Location', y='Rating', data=google_df)
plt.title('Distribution of Ratings by Location on Google Review')
plt.xlabel('Location')
plt.ylabel('Rating')
plt.xticks(rotation=45)
plt.show()
```

The code above generates a box plot to visualize the distribution of ratings in the TripAdvisor and Google reviews dataset based on different locations. The plots provide insights into how ratings vary across different locations.

9. Word Cloud for TripAdvisor and Google Reviews

```
# Word Cloud for TripAdvisor Reviews
from wordcloud import WordCloud

wordcloud = WordCloud(width=800, height=400).generate(' '.join(tripadvisor_df['Review']))
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for TripAdvisor Reviews')
plt.show()
```

```
# Word Cloud for Google Reviews
wordcloud = WordCloud(width=800, height=400).generate(' '.join(google_df['Review']))
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Google Reviews')
plt.show()
```

The code above generates word clouds to visualize the most frequently occurring words in the TripAdvisor and Google reviews dataset. Word clouds provide an intuitive representation of the prominent words in a collection of text data – providing a quick overview of prevalent sentiments and topics in the reviews.

10. Calculating the length of reviews

```
# Calculate the length of reviews and create a new column
tripadvisor_df['Review Length'] = tripadvisor_df['Review'].apply(len)
google_df['Review Length'] = google_df['Review'].apply(len)
```

This code calculates the length of each review in the TripAdvisor and Google reviews datasets and creates a new column ('Review Length') to store this information. Review length can provide insight into the distribution and composition of reviews.

11. Distribution of Review Lengths

```
# Distribution of Review Lengths for TripAdvisor
plt.figure(figsize=(10, 6))
sns.histplot(data=tripadvisor_df, x='Review Length', bins=50, kde=True)
plt.title('Distribution of Review Lengths on TripAdvisor')
plt.xlabel('Review Length')
plt.ylabel('Count')
plt.show()

# Distribution of Review Lengths for Google Review
plt.figure(figsize=(10, 6))
sns.histplot(data=google_df, x='Review Length', bins=50, kde=True)
plt.title('Distribution of Review Lengths on Google Review')
plt.xlabel('Review Length')
plt.ylabel('Count')
plt.show()
```

This code generates histograms to visualize the distribution of review lengths in both the TripAdvisor and Google reviews datasets. The histograms provide a view into the spread of review lengths.

12. Column verification for merging

```
# Making Sure the columns are correct for merging
print('Trip Advisor Columns: \n')
print(tripadvisor_df.columns)

print('\nGoogle Columns: \n')
print(google_df.columns)
```

This code prints the column names of the TripAdvisor and Google Reviews datasets, to ensure that the columns are correctly identified and aligned for merging.

13. Filtering Required Columns for Sentiment Analysis

```
# Only taking the required columns
tripadvisor_df_filtered = tripadvisor_df[['Review', 'Rating']]
google_df_filtered = google_df[['Review', 'Rating']]
```

This code creates new DataFrames by selecting only the required columns, namely 'Review' and 'Rating', from the original datasets. The filtering is performed to focus the analysis on relevant information required for sentiment analysis.

14. Merge TripAdvisor and Google Reviews Datasets

```
# Merge the two datasets based on common columns
combined_df = pd.concat([tripadvisor_df_filtered, google_df_filtered], ignore_index=True)
combined_df = combined_df.reset_index(drop=True)
```

This code combines the filtered TripAdvisor and Google Reviews datasets into a single DataFrame (`combined_df`). The concatenation is performed based on common columns, and the resulting DataFrame is reset to a new index.

15. Merged Dataset Info

```
combined_df.info()
```

This code prints information about the merged dataset (`combined_df`).

16. Display initial rows of the merged dataset

```
combined_df.head()
```

This code prints the first five rows of the merged dataset.

17. Generate a copy of the merged dataset

```
ml_data = combined_df.copy()
ml_data.sample(10)
```

This code creates a copy of the merged dataset (`combined_df`) named `ml_data`. The new dataset is intended for machine learning tasks. It also displays a random sample of 10 rows from the machine learning dataset.

18. Export ML dataset to a CSV file

```
ml_data.to_csv('./ml_data.csv')
```

This code exports the machine learning dataset (`ml_data`) to a CSV file named `ml_data.csv`.

19. Split the rating into categories

```
# Define the bins and labels for rating categories
bins = [0, 2, 3, 5]
labels = ['Negative', 'Neutral', 'Positive']

# Create a new column 'Rating_Category' based on the bins and labels
ml_data['Rating_Category'] = pd.cut(ml_data['Rating'], bins=bins, labels=labels)
```

This code is responsible for categorizing the ratings in the machine learning dataset (`ml_data`) into specific sentiment categories. The sentiment categories include Negative, Neutral and Positive.

20. Retrieve random sample

```
ml_data.sample(10)
```

This code is used to retrieve a random sample of 10 rows from the dataset to provide a quick glimpse into the content / structure of the dataset.

21. Split the dataset into training and testing

```
from sklearn.model_selection import train_test_split

# Split the dataset into 80% training and 20% testing
train_df, test_df = train_test_split(ml_data, test_size=0.2, random_state=42)
```

This code utilizes the `train_test_split`function from the scikit-learn library to split the `ml_data` dataset into the training set (`train_df`) and a testing set (`test_df`). 80% of data is split into training set, and 20% to the testing set. The `random_state` parameter is set to ensure reproducibility.

22. Train the Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, classification_report

# Initialize TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# Transform the text data
X_train = tfidf_vectorizer.fit_transform(train_df['Review'])
X_test = tfidf_vectorizer.transform(test_df['Review'])

# Create the target variables
y_train = train_df['Rating_Category']
y_test = test_df['Rating_Category']

# Initialize and train a Multinomial Logistic Regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
model.fit(X_train, y_train)

# Make predictions on the train set
y_pred_train = model.predict(X_train)

# Make predictions on the test set
y_pred_test = model.predict(X_test)
```

This code implements the sentiment analysis model using TF-IDF vectorizer and a Multinomial Logistic Regression classifier. The training and testing sets are transformed into TF-IDF feature matrices, and a logistic regression model is trained on the training set. The model is then used to make predictions on both the training and testing sets.

23. Evaluate the Model

```
# Evaluate the model on Training and Testing Data
print('Training\n')
accuracy_train = accuracy_score(y_train, y_pred_train)
f1_score_train = f1_score(y_train, y_pred_train, average='weighted')
print(f"Accuracy of the model: {accuracy_train:.2f}")
print(f'F1 Score: {f1_score_train:.2f}')
print(f'Classification Report: \n {classification_report(y_train, y_pred_train)}')

print('Testig\n')
accuracy_test = accuracy_score(y_test, y_pred_test)
f1_score_test = f1_score(y_test, y_pred_test, average='weighted')
print(f"Accuracy of the model: {accuracy_test:.2f}")
print(f'F1 Score: {f1_score_test:.2f}')
print(f'Classification Report: \n {classification_report(y_test, y_pred_test)}')
```

This code evaluates the performance of the sentiment analysis model on both the training and testing datasets. It calculates accuracy, F1 scores and produces a classification report for both sets.

24. Model saving and loading

```
import pickle
```

```
# Save the TF-IDF vectorizer and model to files using pickle
with open('tfidf_vectorizer.pkl', 'wb') as tfidf_file:
    pickle.dump(tfidf_vectorizer, tfidf_file)

with open('sentiment_model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)
```

```
with open('tfidf_vectorizer.pkl', 'rb') as tfidf_file:
    tfidf_vectorizer = pickle.load(tfidf_file)

with open('sentiment_model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)
```

This code imports the `pickle` module for saving and loading the trained machine learning model. It then saves the TF-IDF vectorizer (`tfidf_vectorizer`) and the sentiment analysis model (`model`) with the `pickle` module – for future use without retraining. It then loads the TF-IDF vectorizer and the sentiment analysis model.

25. Model prediction

The code below defines the function, `predict_rating_category`, that uses the trained sentiment analysis model and TF-IDF vectorizer to predict the rating category for a given restaurant review.

```
def predict_rating_category(user_input, model=None, tfidf_vectorizer=None):
    """
    Predicts the rating category (Negative, Neutral, or Positive) for a given user input.

    Parameters:
    - model: A trained classification model (e.g., Multinomial Logistic Regression).
    - tfidf_vectorizer: A TF-IDF vectorizer fitted on the training data.
    - user_input: The user's restaurant review text.

    Returns:
    - predicted_category: The predicted rating category.
    """
    if model == None:
        try:
            with open('sentiment_model.pkl', 'rb') as model_file:
                model = pickle.load(model_file)
        except FileNotFoundError:
            print('Please make sure the sentiment_model.pkl is present inside the same level as this notebook')

    if tfidf_vectorizer == None:
        try:
            with open('tfidf_vectorizer.pkl', 'rb') as tfidf_file:
                tfidf_vectorizer = pickle.load(tfidf_file)
        except FileNotFoundError:
            print('Please make sure the tfidf_vectorizer.pkl is present inside the same level as this notebook'

    # Preprocess the user input using the same TF-IDF vectorizer
    user_input_tfidf = tfidf_vectorizer.transform([user_input])

    # Predict the rating category using the trained model
    predicted_class = model.predict(user_input_tfidf)
    probabilities = model.predict_proba(user_input_tfidf)[0]


    # Create a dictionary of probabilities for each category
    rating_categories = ['Negative', 'Neutral', 'Positive']
    probabilities_dict = {category: prob for category, prob in zip(rating_categories, probabilities)}
    # Map the predicted class label to the corresponding category
    # rating_categories = ['Negative', 'Neutral', 'Positive']
    # print(predicted_class[0])

    return predicted_class[0], probabilities_dict
```

26. Sample model predictions

```
user_input = "The food was terrible, and the service was slow."
predicted_category, probabilities = predict_rating_category(user_input, model, tfidf_vectorizer)
print(f"Predicted Rating Category: {predicted_category}")
print("Probabilities:")
for category, prob in probabilities.items():
    print(f"{category}: {prob:.4f}")
```

For the given user input above, the restaurant review is predicted as Negative with a probability of 85.27% - indicating a strongly negative sentiment.

Another example:

```
user_input = "The food was very good, and the service was excellent. Worth going to that restaurant"
predicted_category, probabilities = predict_rating_category(user_input)
print(f"Predicted Rating Category: {predicted_category}")
print("Probabilities:")
for category, prob in probabilities.items():
    print(f"{category}: {prob:.4f}")
```

The above restaurant review is predicted as Positive, with a probability of 99.89% - indicating a highly positive sentiment.

VII. Future Improvements: I'll consider using more advanced models or fine-tuning hyperparameters for better performance. I'll explore additional features or sentiment analysis techniques for improvement.

VIII. Documentation of the usage of the software: The section below provides instructions on how to run the provided code. This documentation provides users with clear instructions on installing the required dependencies, running the software using either Jupyter Notebook or a Python script, and utilizing the prediction function.

Requirements:

- Python 3.x
- Jupyter Notebook (optional, if running the provided notebook)
- Required Python packages (install using `**pip install package_name**`):
  - pandas
  - numpy
  - matplotlib
  - seaborn
  - scikit-learn
  - wordcloud
  - pickle

Installation:

1. Clone the repository:

```
git clone https://github.com/palankihiran/CourseProject.git
cd CourseProject
```

2. Install required packages

```
pip install pandas numpy matplotlib seaborn scikit-learn wordcloud
```

Running the Software:

Jupyter Notebook:

1. Open Jupyter Notebook
2. Navigate to the `**Sentiment_Analysis.ipynb**` notebook.

3. Run the notebook cell by cell to observe the step-by-step execution.

Python Script (for direct execution):

1. Open a terminal or command prompt.

2. Navigate to the project directory:

```
cd /path/to/CourseProject
```

3. Ensure model.pkl, tokenizer.pkl are in the same root folder as the python script. Then run the script:

```
python sentiment_analysis_script.py
```

Using the Prediction Function:

The software provides a function (`**predict_rating_category**`) for predicting sentiment categories based on user input.

1. Open a Python interpreter or a script.

2. Import the function:

```
from sentiment_analysis_script import predict_rating_category
```

3. Use the function with a user input:

```
user_input = "The food was excellent, and the service was prompt."
predicted_category, probabilities = predict_rating_category(user_input)
print(f"Predicted Rating Category: {predicted_category}")
print("Probabilities:")
for category, prob in probabilities.items():
    print(f"{category}: {prob:.4f}")
```

IX. Brief description of the contribution of the team member: I've worked on the course project independently, and all contributions are from myself.

X. Conclusion: The Restaurant Review Sentiment Analysis project demonstrates the implementation of a machine learning model to predict the sentiment of restaurant reviews. By combining data from TripAdvisor and Google Reviews, preprocessing, and training a Multinomial Logistic Regression model, the project provides a valuable tool for restaurant owners and stakeholders to gain deeper insights into customer feedback. The model can help identify areas for improvement and enhance the overall customer experience. Additional features such as sentiment trends over time can be incorporated by using more advanced natural language processing techniques.