3.2.2_sentiment_analysis

April 4, 2024

0.1 Sentiment Analysis Visualization of Movie Reviews (68 lines in total)

The main tasks accomplished by the provided code are as follows:

1. **Plotting Sentiment Proportions**: The code generates a stacked bar chart displaying the proportions of positive and negative sentiments for different regions (or IPs). This visualization helps in understanding the distribution of sentiments across different regions. The chart settings ensure that Chinese characters are displayed correctly, and the size of the figure is set for clear visibility. Positive sentiments are represented in green, and negative sentiments are represented in red. The data for this chart is expected to come from a DataFrame (`proportions_df`) containing at least three columns: 'Region', 'Positive', and 'Negative'.
2. **Plotting Average Sentiment Scores**: The second part of the code creates a horizontal bar chart that illustrates the average sentiment scores for different regions. This helps in identifying the overall sentiment trend per region based on the average scores. The setup for this chart is similar to the first one, with Chinese characters display capability and a specified figure size. The average sentiment scores are represented in sky blue, and the data for this chart is expected to come from another DataFrame (`sorted_avg_sentiments`) with at least two columns: 'Region' and 'average_sentiment'.

Both parts of the code share common features such as setting the font for Chinese characters, defining the size of the plot, labeling the axes, setting the title, and displaying the legend. These sections aim to visualize sentiment analysis results in an accessible and interpretable manner, facilitating a better understanding of regional sentiment trends based on the comments analyzed.

[ ]:
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from snownlp import SnowNLP # refered to the tutorial from Google
# tutorial link: https://www.datacamp.com/tutorial/text-analytics-beginners-nltk
```

0.1.1 1. Read & Group Data

[ ]:
```
# Read data and drop missing values
comment_by_ip_int = pd.read_csv('cleaned_combind_comment.csv',␣
 ↪encoding='utf-8').dropna()

# Group by 'ip' column and extract 'comment' for each group
comment_by_ip = comment_by_ip_int.groupby('ip')['comment']
```

```
/var/folders/sg/9v6g6q1n52zgjdgv93fw7stc0000gn/T/ipykernel_79509/2526507305.py:2
: DtypeWarning: Columns (4) have mixed types. Specify dtype option on import or
set low_memory=False.
  comment_by_ip_int = pd.read_csv('cleaned_combind_comment.csv',
encoding='utf-8').dropna()
```

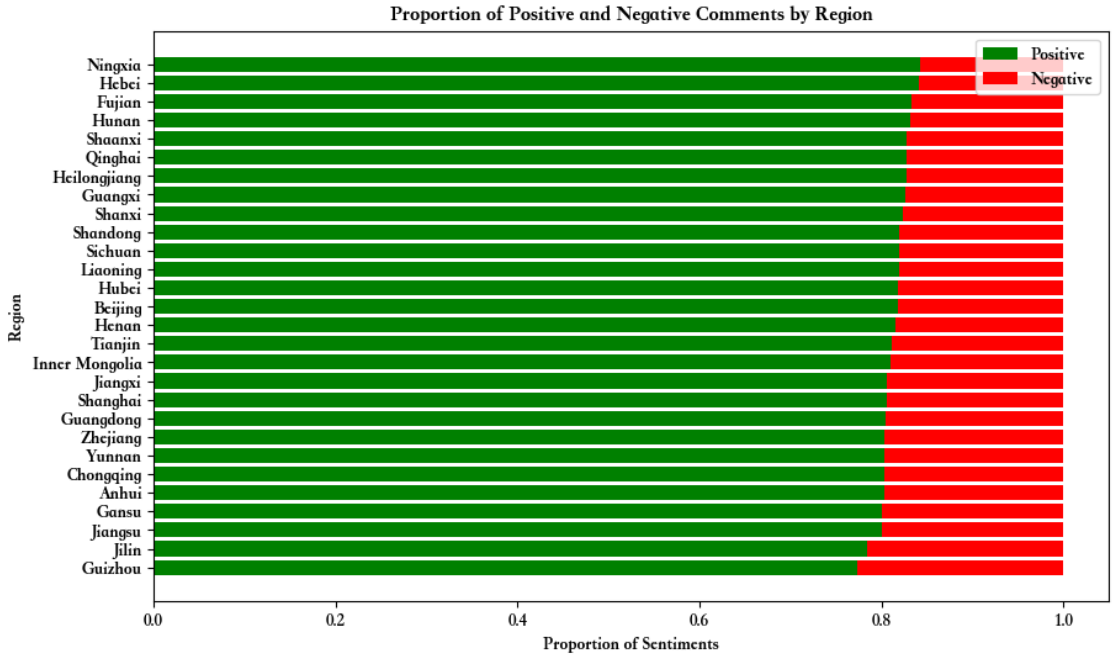
0.1.2 2. Compute Sentiment Score

[ ]:
```
# Apply sentiment analysis to the comments of each group under the guidance of␣
 ↪GPT
sentiments_df = comment_by_ip.apply(
    lambda comments: [SnowNLP(comment).sentiments for comment in comments]
).reset_index(name='sentiment_scores')
```

0.1.3 3. Construct & Plot Proportions of Positive/Negative Sentiments

[ ]:
```
# Convert sentiment scores to a dictionary
sentiments_dict = sentiments_df.set_index('ip')['sentiment_scores'].to_dict()

# Initialize dictionaries to store the proportions of positive and negative␣
 ↪sentiments
posi_proportions = {}
nega_proportions = {}

# Calculate the proportions of positive and negative sentiments for each IP
for region, sentiments in sentiments_dict.items():
    total_comments = len(sentiments)
    positive_count = len([score for score in sentiments if score > 0.5])
    negative_count = total_comments - positive_count

    posi_proportions[region] = positive_count / total_comments if␣
 ↪total_comments else 0
    nega_proportions[region] = negative_count / total_comments if␣
 ↪total_comments else 0

# Convert sentiment proportions to DataFrame and sort
proportions_df = pd.DataFrame({
    'Region': list(sentiments_dict.keys()),
    'Positive': list(posi_proportions.values()),
    'Negative': list(nega_proportions.values())
}).sort_values(by='Negative', ascending=False)

# Save the results to a CSV file
proportions_df.to_csv('posi_nega_comments_ratios.csv', index=False)

# Create and display a stacked bar chart (positive and negative sentiment␣
 ↪proportions)
```

```
plt.rcParams['font.sans-serif'] = ['Songti SC']
plt.figure(figsize=[10,6])
plt.barh(proportions_df['Region'], proportions_df['Positive'],
         color='green', label='Positive')
plt.barh(proportions_df['Region'], proportions_df['Negative'],
         left=proportions_df['Positive'], color='red', label='Negative')
plt.xlabel('Proportion of Sentiments')
plt.ylabel('Region')
plt.title('Proportion of Positive and Negative Comments by Region')
plt.legend()
plt.show()
```



0.1.4 4. Construct & Plot Average Sentiments

[ ]:
```
# Calculate and sort average sentiment scores
average_sentiments = sentiments_df.copy()  # Copy the data to avoid modifying␣
 ↪the original data
average_sentiments['average_sentiment'] =␣
 ↪average_sentiments['sentiment_scores'].apply(
    lambda scores: sum(scores) / len(scores) if scores else 0)
sorted_avg_sentiments = average_sentiments.sort_values(by='average_sentiment',␣
 ↪ascending=False)

# Save the average sentiment score results to a CSV file
sorted_avg_sentiments.to_csv('sorted_avg_sentiments.csv', index=False)
```

```
# Create and display a bar chart for average sentiment scores
plt.rcParams['font.sans-serif'] = ['Songti SC']
plt.figure(figsize=(10,6))
plt.barh(sorted_avg_sentiments['ip'],␣
 ↪sorted_avg_sentiments['average_sentiment'], color='skyblue')
plt.xlabel('Average Sentiment Score')
plt.ylabel('Region')
plt.title('Average Sentiment Score by Region')
plt.show()
```