3.2.1_wordcloud_plot

April 4, 2024

1 Word Cloud Plots (136 lines in total)

The file outlines a task focused on data analysis and visualization, specifically through the use of word clouds generated from comments associated with different levels or types of content, such as movie reviews. The main steps covered in the document include:

1. **Data Loading**: Importing datasets from CSV files that contain comments and movie information.
2. **Data Preprocessing**: Cleaning the data by removing rows with null values in specific columns like 'comment', 'level', or 'movie', and merging datasets based on common attributes.
3. **Text Cleaning and Processing**: Applying text preprocessing to the comments, which includes removing stopwords and segmenting text into words using `jieba`, a Chinese text segmentation tool.
4. **Word Cloud Generation**:
   - Extracting top words from cleaned comments based on frequency and saving these into CSV files.
   - Generating word clouds from these top words, where the text data is visualized to highlight common terms or themes within the comments. This involves creating visual representations for different categories such as levels of satisfaction ('h', 'm', 'l') or movie types ('Gay', 'Les', 'Other') based on the dataset's classification.
5. **Aggregated Visualization**: Merging new word lists with translated or aggregated terms and generating word clouds from these aggregated datasets. This step aims to provide insights into common themes across different segments of the data, reflected by the frequency and prominence of words in the visualizations.
6. **Functions and Abstraction**: The tasks are encapsulated into functions for better modularity and reusability. Functions like `extract_top_words`, `generate_aggregated_wordcloud`, and `display_wordcloud` are defined to handle specific parts of the workflow, such as extracting top words, generating a word cloud from a data frame, and displaying the generated word cloud with matplotlib, respectively.
7. **Visualization Customization**: Customizing the appearance of word clouds according to different data categories (such as 'level' or 'type') and ensuring that they are presented with clear titles and appropriate figure sizes.

The overall goal of the document is to provide a structured approach to analyzing and visualizing textual data from comments, enabling insights into the frequency and distribution of words within

different segments of the data, all within the context of Chinese language content, as indicated by the use of `jieba` for text segmentation and the presence of Chinese file paths and stopwords.

[ ]:
```
import pandas as pd
from wordcloud import WordCloud # refered to the tutorial from Google
# tutorial link: https://www.datacamp.com/tutorial/wordcloud-python
import matplotlib.pyplot as plt
import jieba
```

[ ]:
```
def extract_top_words(comments_clean, output_path, num_words=200):
    """
    Extract top words from cleaned comments and save to a CSV file.

    Parameters:
        comments_clean: String containing all cleaned comments.
        output_csv: Path to save the top words CSV file.
        num_words: Number of top words to extract. Default is 200.
    """
    # Process the cleaned comments to extract word frequency
    wordcloud = WordCloud()
    # Create a temporary WordCloud object for processing text
    process_word = wordcloud.process_text(comments_clean)

    # Sort the words by frequency in descending order
    sort = sorted(process_word.items(), key=lambda e: e[1], reverse=True)

    # Create a DataFrame for the top words
    top_words = pd.DataFrame(sort[:num_words], columns=['Word', 'Freq'])
    top_words['Word'].to_csv(output_path, index=False)

    # Return the DataFrame for further use
    return top_words

def generate_aggregated_wordcloud(input_path, top_words_df):
    """
    Generate a word cloud from an existing DataFrame of top words and a new CSV␣
 ↪file containing English translations.

    Parameters:
        input_path: Path to the CSV file containing English words.
        top_words_df: DataFrame containing the original words and their␣
 ↪frequencies.
    """
    # Read new English words from CSV
    new_column = pd.read_csv(input_path, usecols=['Word'])

    # Update the 'Word' column in the top words DataFrame with English words
```

```
    top_words_df['Word'] = [word.lower().strip('.') for word in␣
 ↪new_column['Word']]

    # Aggregate frequencies if there are duplicate words after translation
    top_words_aggregated = top_words_df.groupby('Word')['Freq'].sum().
 ↪reset_index()

    # Convert aggregated word frequencies into a dictionary
    word_freq = dict(zip(top_words_aggregated['Word'],␣
 ↪top_words_aggregated['Freq']))

    # Generate a word cloud from the aggregated frequencies
    wordcloud = WordCloud(font_path = '~/Library/Fonts/Arial Unicode.ttf',
                          width=800, height=600, background_color='white').
 ↪generate_from_frequencies(word_freq)

    # Return the WordCloud object for further use or visualization
    return wordcloud

def display_wordcloud(wordcloud, title, figsize=(10, 8)):
    """
    Displays a word cloud with a specified title and figure size.

    :param wordcloud: The WordCloud object to be displayed.
    :param title: Title for the word cloud image.
    :param figsize: Tuple indicating figure size. Defaults to (10, 8).
    """
    plt.figure(figsize=figsize)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title)
    plt.axis('off')  # Hide the axes
    plt.show()
```

1.1 Word Cloud Plots by levels of satisfaction ('h', 'm', 'l') - Chinese

[ ]:
```
# 1. Load the data
data = pd.read_csv('/Users/jessie/Documents/coding/macs30112/
 ↪final-project-1stj_3nfps/part3-analysis_and_visualization/3.
 ↪1_explorative_analysis/cleaned_combined_comment.csv')

# 2. Data preprocessing
# Ensure 'comment' and 'level' columns have no null values
data = data.dropna(subset=['comment', 'level'])

# 3. Text cleaning
# Load an external list of stopwords
```

```
stopwords_path = 'stopwords/hml.txt'  # The path for the stopwords file
with open(stopwords_path, 'r', encoding='utf-8') as f:
    stopwords = set([line.strip() for line in f.readlines()])

# 4. Process by level
levels = ['h', 'm', 'l']
for level in levels:
    comments = data[data['level'] == level]['comment'].tolist()
    comments_clean = ' '.join([word for comment in comments for word in
                               jieba.cut(comment) if word not in stopwords])

    # 5. Generate word cloud
    wordcloud = WordCloud(font_path='/System/Library/Fonts/STHeiti Light.ttc',␣
 ↪width=800, height=600, background_color='white').generate(comments_clean)
    display_wordcloud(wordcloud, f'Level {level.upper()} Comments WordCloud')
    top_words = extract_top_words(comments_clean, f"top_words_chinese/
 ↪Chinese_{level}.csv")
```

```
Building prefix dict from the default dictionary …
Dumping model to file cache
/var/folders/sg/9v6g6q1n52zgjdgv93fw7stc0000gn/T/jieba.cache
Loading model cost 0.462 seconds.
Prefix dict has been built successfully.
```

Level H Comments WordCloud

Level M Comments WordCloud

1.2 Word Cloud Plots by levels of satisfaction ('h', 'm', 'l') - English

[ ]:
```
# 6. Visualization
level_dict = {'h':'top_words_eng/English_h.csv',
              'm':'top_words_eng/English_m.csv',
              'l':'top_words_eng/English_l.csv'}
for level, filename in level_dict.items():
    new_wordcloud = generate_aggregated_wordcloud(filename, top_words)
    display_wordcloud(new_wordcloud, f'Level {level.upper()} Comments␣
 ↪WordCloud')
```

Level H Comments WordCloud

Level M Comments WordCloud

1.3 Word Cloud Plots by movie types ('Gay', 'Les', 'Other') - Chinese

[ ]:
```
# 1. Load datasets
movies = pd.read_csv('/Users/jessie/Documents/coding/macs30112/
 ↪final-project-1stj_3nfps/part3-analysis_and_visualization/3.
 ↪1_explorative_analysis/cleaned_movie_list.csv')
comments = pd.read_csv('/Users/jessie/Documents/coding/macs30112/
 ↪final-project-1stj_3nfps/part3-analysis_and_visualization/3.
 ↪1_explorative_analysis/cleaned_combined_comment.csv')

# 2. Data preprocessing (simplified for this example, adjust as needed)
# Ensure movie names are not empty in both datasets
movies = movies.dropna(subset=['movie'])
comments = comments.dropna(subset=['comment', 'movie'])

# 3. Classify movies
# Add a new column in the movie dataset to label the movie type as 'Gay',␣
 ↪'Les', or 'Other'
```

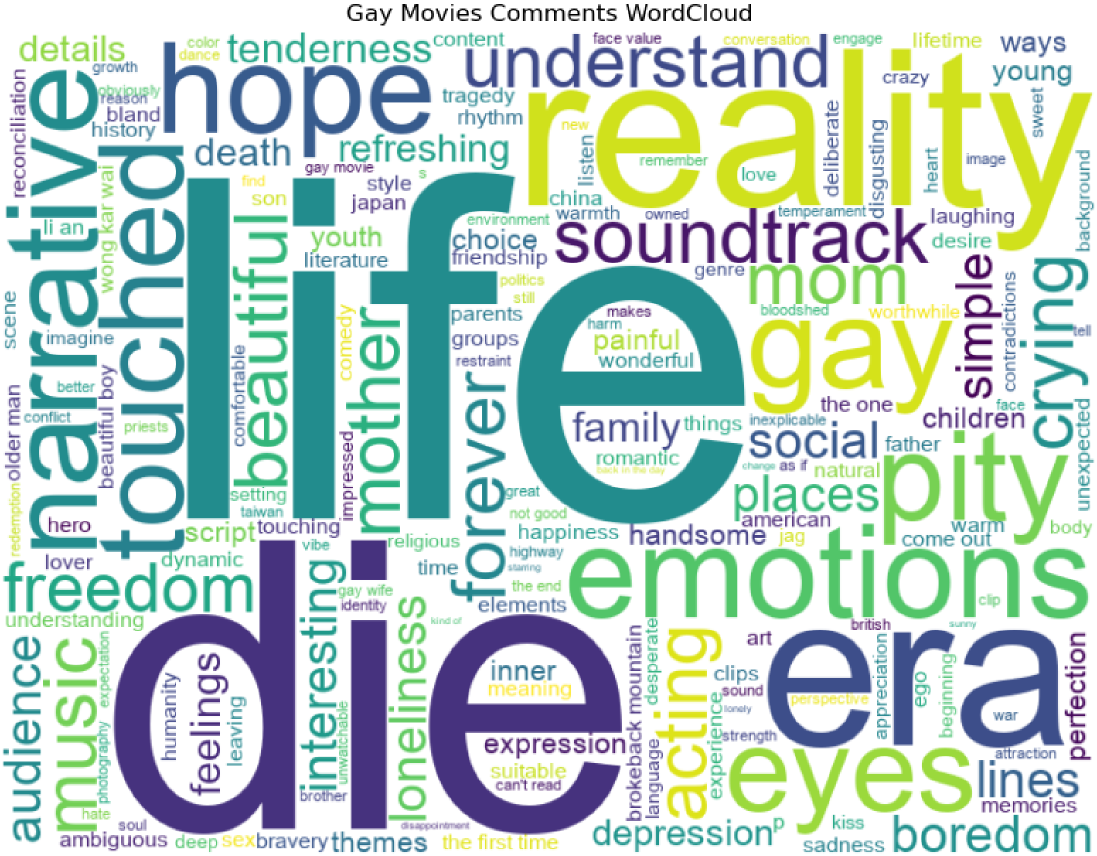
```
movies['Type'] = movies.apply(lambda x: 'Gay' if x['Gay'] == 1 else ('Les' if␣
 ↪x['Les'] == 1 else 'Other'), axis=1)

# 4. Merge datasets
# Use the movie name as the key for merging
merged_data = pd.merge(comments, movies[['movie', 'Type']], left_on='movie',␣
 ↪right_on='movie', how='left')

# Ensure the 'Type' column in the merged dataset is not empty
merged_data = merged_data.dropna(subset=['Type'])

# 5. Generate word clouds
for type in ['Gay', 'Les']:
    data = merged_data[merged_data['Type'] == type]['comment'].tolist()
    # Code for text cleaning and word cloud generation
    stopwords_path = 'stopwords/GL.txt'  # Path to the stopwords file
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = set([line.strip() for line in f.readlines()])
    comments_clean = ' '.join([word for comment in data for word in jieba.
 ↪cut(comment) if word not in stopwords])
    wordcloud = WordCloud(font_path='/System/Library/Fonts/STHeiti Light.ttc',␣
 ↪width=800, height=600, background_color='white').generate(comments_clean)
    display_wordcloud(wordcloud, f"{type} Movies Comments WordCloud")
    top_words = extract_top_words(comments_clean, f"top_words_chinese/
 ↪Chinese_{type}.csv")
```

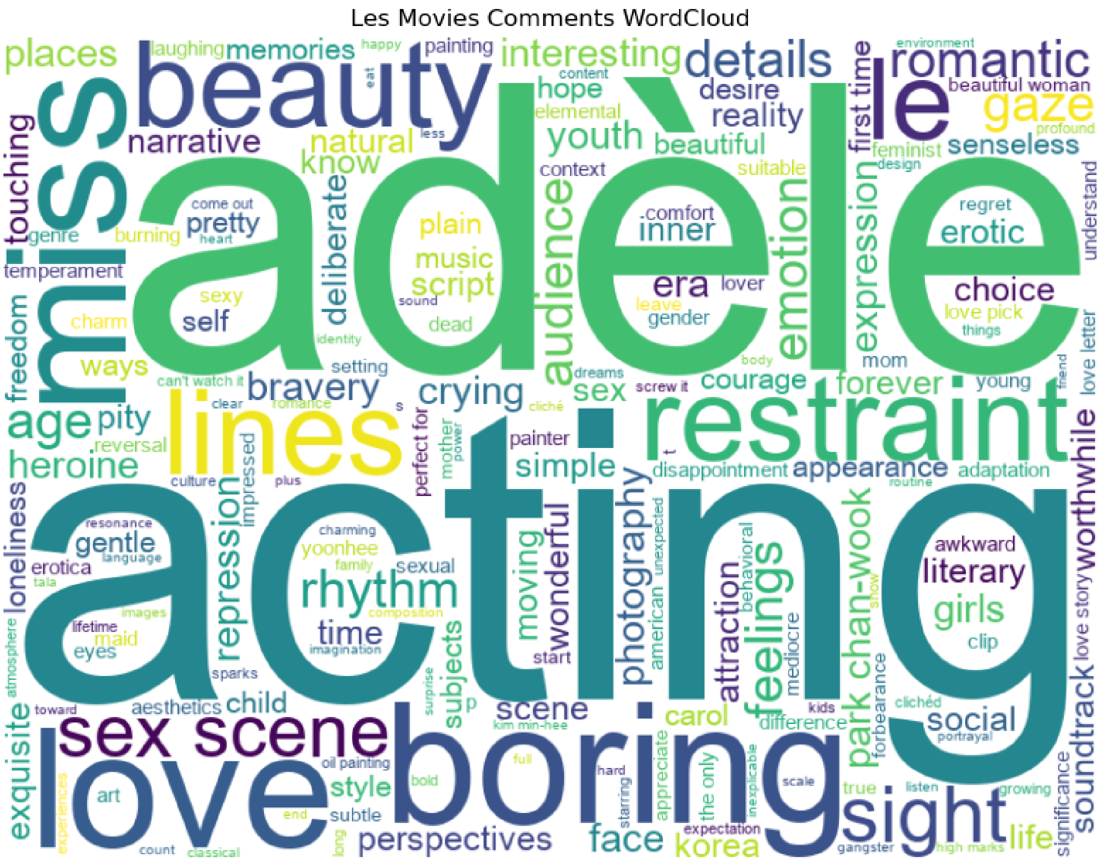
Gay Movies Comments WordCloud

1.4 Word Cloud Plots by movie types ('Gay', 'Les', 'Other') - English

[ ]:
```
# 6. Visualization
type_dict = {'Gay':'top_words_eng/English_Gay.csv',
             'Les':'top_words_eng/English_Les.csv'}
for type, filename in type_dict.items():
    new_wordcloud = generate_aggregated_wordcloud(filename, top_words)
    display_wordcloud(new_wordcloud, f"{type} Movies Comments WordCloud")
```

Gay Movies Comments WordCloud




Les Movies Comments WordCloud
adèle
acting
love
boring
beauty
miss
lines
restraint
le
sight
sex scene
details
romantic
gaze
rhythm
feelings
park chan-wook
emotion
audience
expression
photography
places
interesting
memories
laughing
painting
narrative
natural
know
youth
hope
desire
reality
beautiful
first time
senseless
beautiful woman
touching
pretty
plain
music
script
inner
comfort
era
lover
erotic
choice
regret
temperament
freedom
charm
self
deliberate
ways
bravery
crying
sex
courage
forever
young
age
pity
heroine
reversal
simple
disappointment
appearance
adaptation
worthwhile
loneliness
gentle
erotica
repression
impressed
time
moving
wonderful
attraction
awkward
literary
girls
clip
yoonhee
sexual
subjects
scene
carol
social
soundtrack
exquisite
aesthetics
child
style
subtle
perspectives
face
korea
life
art
mediocre
behavioral
american
appreciate
the only
difference
forbearance
significance
love story
love letter
understand
mom
feminist
suitable
context
elemental
genre
burning
sexy
dead
setting
painter
perfect for
mother
eyes
maid
lifetime
start
mom