search_rednote.py > ...

```python
from selenium import webdriver
import subprocess
import time
import pandas as pd
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

def getElement(eles):
    if eles and len(eles) != 0:
        return eles[0].text
    else:
        return 'None'

def get_renote_posts(key, file_path):
    browser_path = "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe"
    subprocess.Popen([browser_path, '--remote-debugging-port=9222'])
    time.sleep(5)

    options = Options()
    options.add_experimental_option("debuggerAddress", "localhost:9222")
    driver = webdriver.Chrome(options=options)

    driver.get('https://www.xiaohongshu.com/search_result?keyword='+key+'&source=web_explore_feed&type=51')
    time.sleep(10)  # Wait for the page to load

    insert_Data = []
    ids = []
    y = 0
    for i in range(20):  # Loop for 20 pages
        print(i)
        try:
            y += 600
            driver.execute_script(f'window.scrollTo(0,{y})')
            time.sleep(2.5)
            video_list = driver.find_elements(By.CSS_SELECTOR, '.note-item')
            if len(video_list) == 0:
                break
            for video in video_list:
                if video.find_elements(By.CSS_SELECTOR, '.title'):
                    url = video.find_element(By.CSS_SELECTOR, ".cover").get_attribute('href')
                    title = video.find_element(By.CSS_SELECTOR, '.title').text
                    name = video.find_element(By.CSS_SELECTOR, '.name').text
                    count = video.find_element(By.CSS_SELECTOR, '.count').text
                    shareurl = url.replace('pc_user', 'pc_share').replace('explore', 'discovery/item').replace('xsec_token=',
                                                                                                               'source=webshare&xhsshare=pc_web&xsec_token=')

                    if title in ids:
                        continue
                    print(title)
                    ids.append(title)
                    dict = {
                        'Title': title,
                        'Author': name,
                        'Like Count': count,
                        'URL': url,
                        'Share URL': shareurl,
                    }
                    insert_Data.append(dict)
        except Exception as e:
            print(e)

    df = pd.DataFrame(insert_Data)
    df.to_excel(file_path, index=False, engine='openpyxl')

# Call the function with the keyword
key = 'Macau Street Market'
video_urls = get_renote_posts(key, 'Macau_Street_Market.xlsx')
```

generate_wordcloud.py | wordcloud_english.py | translate.py | search_rednote.py | comments_collected.py

comments_collected.py > ...

```python
import csv
import io
import os
import pandas as pd
import random
import re
import subprocess
import sys
import time
from datetime import datetime, timedelta
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait

def getElement(eles):
    if eles and len(eles) != 0:
        return eles[0].text
    else:
        return 'N/A'

def scrape(url, pid):
    driver.get(url)
    time.sleep(5)
    WebDriverWait(driver, 20).until(
        EC.visibility_of_element_located((By.CSS_SELECTOR, '.list-container'))
    )
    div_element = driver.find_element(By.CSS_SELECTOR, '.note-scroller')
    y = 0
    for i in range(200):
        try:
            print('Scrolling ' + str(i))
            y += 2000
            driver.execute_script("arguments[0].scrollTop = " + str(y) + ";", div_element)
            if driver.find_elements(By.CSS_SELECTOR, '.end-container'):
                break
        except:
            print('Jumping')
        time.sleep(3)

    # Expand comments
    clicklist = driver.find_elements(By.CSS_SELECTOR, '.show-more')
    for clickitem in clicklist:
        try:
            clickitem.click()
            time.sleep(0.4)
        except:
            print('Jumping')

    # Click more comments
    for j in range(3000):
        try:
            if driver.find_elements(By.CSS_SELECTOR, '.show-more'):
                driver.find_element(By.CSS_SELECTOR, '.show-more').click()
                time.sleep(0.2)
            else:
                break
        except:
            print('Jumping')

    video_list = driver.find_elements(By.CSS_SELECTOR, '.comment-item')
    for video in video_list:
        author_name = getElement(video.find_elements(By.CSS_SELECTOR, '.author a'))
        content = getElement(video.find_elements(By.CSS_SELECTOR, '.note-text'))
        comment_date = getElement(
            video.find_elements(By.CSS_SELECTOR, '.right > div.info > div.date > span:nth-child(1)'))
        location = getElement(video.find_elements(By.CSS_SELECTOR, '.right > div.info > div.date > .location'))
        like_count = getElement(video.find_elements(By.CSS_SELECTOR, '.like .count'))
        reply_count = getElement(video.find_elements(By.CSS_SELECTOR, '.reply .count'))
        print(str(author_name))
        dict = {
            'Post ID': str(pid),
            'Author Name': str(author_name),
            'Comment Content': str(content),
            'Comment Date': str(comment_date),
            'Comment Location': str(location),
            'Comment Likes': str(like_count),
            'Comment Replies': str(reply_count),
        }
        # writer.writerow(dict)
        data_list_insert.append(dict)
    print(len(data_list_insert))

if __name__ == "__main__":
    # Create a new CSV file and write data
    csv_filename = 'data.csv'
    browser_path = "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe"
    subprocess.Popen([browser_path, '--remote-debugging-port=9222'])
    time.sleep(5)
    print('End')
    options = Options()
    options.add_experimental_option("debuggerAddress", "localhost:9222")
    driver = webdriver.Chrome(options=options)
    data_list_insert = []
    # todo: Replace with the actual URL of the post
    df = pd.read_excel('Macau Street Market.xlsx', sheet_name='Sheet1', engine='openpyxl')
    data_list = df.to_dict(orient='records')
    try:
        for pitem in data_list:
            url = pitem['url']
            pid = pitem['id']
            try:
                scrape(url, pid)
            except Exception as e:
                print(e)
    except:
        print('Failed')
    df = pd.DataFrame(data_list_insert)
    df.to_excel('Macau Street Market Comments.xlsx', index=False, engine='xlsxwriter')
```

generate_wordcloud.py | wordcloud_english.py | translate.py | search_rednote.py | comments_collected.py

generate_wordcloud.py > ...

```python
# -*- coding: utf-8 -*-
import pandas as pd
from wordcloud import WordCloud
from PIL import Image
import os
import matplotlib
import jieba

matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei']
matplotlib.rcParams['axes.unicode_minus'] = False
sheet_name = 'Sheet1'

column_name = 'Comment Content'  # Replace with the actual column name
df = pd.read_excel('Comments.xlsx', sheet_name=sheet_name, usecols=[column_name], engine='openpyxl')

# Convert data to dictionary format
data_list = df.to_dict(orient='records')

# Set the folder to save the word cloud
wordcloud_save_folder = 'Macau Street Market'
if not os.path.exists(wordcloud_save_folder):
    os.makedirs(wordcloud_save_folder)

# Combine all comments into a single text
text = ''
for item in data_list:
    if str(item['Comment Content']) != 'nan' and len(item['Comment Content']) >= 3:
        text += '\n' + str(item['Comment Content'])

# Use jieba for Chinese word segmentation
words = jieba.lcut(text)

# Load stopwords list
def add_stop_words(list):
    stop_words = set()
    for item in list:
        stop_words.add(item.strip())
    return stop_words

stopword_list = open('stopword.txt', encoding='utf-8').readlines()
stopwords = add_stop_words(stopword_list)

# Filter out stopwords and single character words
filtered_words = [word for word in words if word not in stopwords and len(word) > 1]
text = ' '.join(filtered_words)

# Generate word cloud
wordcloud = WordCloud(width=1000, height=700, background_color='white', font_path='C:\\Windows\\Fonts\\msyh.ttc').generate(text)

# Save the word cloud as an image file
image_path = os.path.join(wordcloud_save_folder, f'Word Frequency.png')
wordcloud_image = Image.fromarray(wordcloud.to_array())
wordcloud_image.save(image_path)

print(f"Word cloud has been saved to {image_path}.")
```

File Edit Selection View Go ···

Macau Street Market

generate_wordcloud.py | wordcloud_english.py | translate.py | search_rednote.py | comments_collected.py

translate.py > ...

```python
import pandas as pd
from deep_translator import GoogleTranslator

# Read Excel file, keep all columns
sheet_name = 'Sheet2'  # Replace with the actual sheet name
column_name = 'Word'  # Replace with the actual column name

# Read data
df = pd.read_excel('Word Frequency.xlsx', sheet_name=sheet_name, engine='openpyxl')

# Initialize Google Translator
translator = GoogleTranslator(source='zh-CN', target='en')

# Translate each comment
translated_comments = []
for comment in df[column_name]:
    try:
        if pd.isna(comment) or not comment.strip():  # If comment is empty or NaN
            translated_comments.append('No comment')  # Or you can choose another placeholder
        else:
            # Translate the comment content
            translated_comment = translator.translate(comment)
            translated_comments.append(translated_comment)
    except Exception as e:
        print(f"Error translating: {comment} - {e}")
        translated_comments.append('Translation Error')  # If translation fails, add error message

# Add translated results to a new column
df['Translated Comments'] = translated_comments

# Save the updated data back to the original Excel file's Sheet2, replacing the old data
with pd.ExcelWriter('Word Frequency.xlsx', engine='openpyxl', mode='a', if_sheet_exists='replace') as writer:
    df.to_excel(writer, sheet_name=sheet_name, index=False)

print("Translation complete, results saved to 'Word Frequency.xlsx' Sheet2.")

```

generate_wordcloud.py | wordcloud_english.py | translate.py | search_rednote.py | comments_collected.py

wordcloud_english.py > ...

```python
# -*- coding: utf-8 -*-
import pandas as pd
from wordcloud import WordCloud
from PIL import Image
import os
import matplotlib

matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei']
matplotlib.rcParams['axes.unicode_minus'] = False
sheet_name = 'Sheet2'
column_name = 'Translated Comments'
count_column_name = 'Count'

df = pd.read_excel('Word Frequency.xlsx', sheet_name=sheet_name,
                   usecols=[column_name, count_column_name], engine='openpyxl')

# Create a word frequency dictionary
word_freq = {}
for index, row in df.iterrows():
    comment = str(row[column_name])
    count = row[count_column_name]
    if len(comment) >= 3 and comment != 'nan':
        if comment in word_freq:
            word_freq[comment] += count
        else:
            word_freq[comment] = count

wordcloud_save_folder = 'Macau Street Market'
if not os.path.exists(wordcloud_save_folder):
    os.makedirs(wordcloud_save_folder)

# Generate the word cloud
wordcloud = WordCloud(width=1000, height=700, background_color='white',
                      font_path='C:\\Windows\\Fonts\\msyh.ttc').generate_from_frequencies(word_freq)

image_path = os.path.join(wordcloud_save_folder, f'wordcloud_translated.png')
wordcloud_image = Image.fromarray(wordcloud.to_array())
wordcloud_image.save(image_path)
print(f"The word cloud has been saved to {image_path}.")
```