CO Conversion Plot for ZSM-5 Catalyst:

This Python script generates an interactive plot to visualize the CO conversion data of a ZSM-5 catalyst against temperature. The plot is created using the Plotly library and can be saved as an HTML file for easy sharing and interaction.

Dependencies:

The following Python libraries are required to run this script:

- NumPy
- Plotly
- SciPy
- scikit-learn

You can install these dependencies using pip:

```
pip install numpy plotly scipy scikit-learn
```

Usage:

1. Prepare a text file containing the temperature and conversion data, with each line in the format `temperature|conversion`. For example:

```
100|10.5
200|25.2
300|45.8
...
```

2. Update the `temperature_array` in the script to match your temperature data range and step size.

3. Update the `conversion_file` variable with the path to your text file containing the conversion data.

4. Optionally, you can customize the plot labels and title by modifying the corresponding arguments in the `plot_conversion_from_text` function call.

5. Run the script, and it will generate an interactive plot displaying the CO conversion data against temperature.

6. The script will save the interactive plot as an HTML file named `zsm5_catalyst_activity.html` in the same directory.

Function Description:

The `plot_conversion_from_text` function takes the following arguments:

- `temperature_array` (array-like): Array containing temperature data.
- `conversion_file` (str): Path to the text file containing conversion data.
- `xlabel` (str, optional): Label for the x-axis. Defaults to 'Temperature (°C)'.
- `ylabel` (str, optional): Label for the y-axis. Defaults to 'CO Conversion (%)'.
- `title` (str, optional): Title of the plot. Defaults to 'ZSM-5 Catalyst Activity'.

The function returns a Plotly figure object displaying the CO conversion data. It also highlights the maximum conversion point on the plot and adds an annotation with the corresponding temperature and conversion value.

Customization:

You can customize various aspects of the plot by modifying the corresponding lines in the code. For example, you can change the plot layout, marker styles, line colors, and annotations by updating the respective parameters in the Plotly `update_layout` and `add_trace` functions.

Raman Spectra Visualization:

This Python script generates an interactive plot to visualize Raman spectra from text files. The plot is created using the Plotly library and can be saved as an HTML file for easy sharing and interaction. It also performs peak detection and calculates the intensity ratio between the disordered and graphitic bands.

Dependencies:

The following Python libraries are required to run this script:

- NumPy
- Plotly
- SciPy

You can install these dependencies using pip:

```
pip install numpy plotly scipy
```

Usage:

1. Prepare text files containing your Raman spectra data. Each file should have one wavenumber-intensity pair per line, separated by whitespace (e.g., tab or space).

2. Update the `text_files` variable in the script with the paths to your text files containing the Raman spectra data.

3. Optionally, you can customize the plot colors, titles, peak detection thresholds, and spacing between spectra by modifying the corresponding arguments in the `plot_raman_from_text` function call.

4. Run the script, and it will generate an interactive plot displaying the Raman spectra.

5. The script will save the interactive plot as an HTML file named `raman_spectra.html` in the same directory.

Function Description:

The `plot_raman_from_text` function takes the following arguments:

- `text_files` (str or list of str): Path or list of paths to the text file(s) containing Raman spectra data.
- `colors` (list of str, optional): List of colors to be used for each spectrum. Defaults to a predefined list of colors.

- `titles` (list of str, optional): List of titles for each spectrum. Defaults to 'Raman Spectrum X' (where X is the spectrum number).
- `peak_threshold` (float, optional): Threshold for peak detection as a fraction of the maximum intensity. Defaults to 0.1.
- `prominence_threshold` (float, optional): Threshold for peak prominence. Defaults to 0.1.
- `spacing` (int, optional): Spacing between spectra on the plot. Defaults to 200.

The function returns a Plotly figure object displaying the Raman spectra. It also performs peak detection, annotates the disordered and graphitic bands, and includes the intensity ratio (I_D/I_G) in the legend for each spectrum.

Customization:

You can customize various aspects of the plot by modifying the corresponding lines in the code. For example, you can change the plot layout, marker styles, line colors, and annotations by updating the respective parameters in the Plotly `update_layout`, `add_trace`, and `add_annotation` functions.
Normalized Raman Spectra Visualization

Normalized Raman Spectra Visualization

This Python script generates an interactive plot to visualize normalized Raman spectra from text files. The plot is created using the Plotly library and can be saved as an HTML file for easy sharing and interaction. It supports two normalization methods: area normalization and max intensity normalization. Additionally, it performs peak detection and annotates the peaks on the plot.

Dependencies:

The following Python libraries are required to run this script:

- NumPy
- Plotly
- SciPy

You can install these dependencies using pip:

```
pip install numpy plotly scipy
```

Usage:

1. Prepare text files containing your Raman spectra data. Each file should have one wavenumber-intensity pair per line, separated by whitespace (e.g., tab or space).

2. Update the `text_files` variable in the script with the paths to your text files containing the Raman spectra data.

3. Optionally, you can customize the normalization method, plot colors, titles, peak detection thresholds, and spacing between spectra by modifying the corresponding arguments in the `plot_normalized_raman_from_text` function call.

4. Run the script, and it will generate an interactive plot displaying the normalized Raman spectra.

5. The script will save the interactive plot as an HTML file named `normalized_raman_spectra_max_intensity.html` (or a different filename based on the chosen normalization method) in the same directory.

Function Descriptions:

`max_intensity_normalize_raman(y)`:

This function normalizes Raman spectra by dividing the intensities by the maximum intensity.

- `y` (array-like): Array containing the intensities.
- Returns: Normalized intensities (array-like).

`plot_normalized_raman_from_text(text_files, normalization_method='area', colors=None, titles=None, peak_threshold=0.1, prominence_threshold=0.1, spacing=200)`:

This function plots normalized Raman spectra from text files.

- `text_files` (str or list of str): Path or list of paths to the text file(s) containing Raman spectra data.
- `normalization_method` (str, optional): Method for normalization. 'area' normalizes by the area under the curve, 'max_intensity' normalizes by the maximum intensity. Defaults to 'area'.
- `colors` (list of str, optional): List of colors to be used for each spectrum. Defaults to a predefined list of colors.
- `titles` (list of str, optional): List of titles for each spectrum. Defaults to 'Normalized Raman Spectrum X' (where X is the spectrum number).
- `peak_threshold` (float, optional): Threshold for peak detection as a fraction of the maximum intensity. Defaults to 0.1.
- `prominence_threshold` (float, optional): Threshold for peak prominence. Defaults to 0.1.
- `spacing` (int, optional): Spacing between spectra on the plot. Defaults to 200.
- Returns: A Plotly figure object displaying the normalized Raman spectra.

The function performs normalization based on the selected method, plots the normalized Raman spectra, performs peak detection, annotates the peaks, and includes the area under the curve in the legend for each spectrum.

Customization:

You can customize various aspects of the plot by modifying the corresponding lines in the code. For example, you can change the plot layout, marker styles, line colors, and annotations by updating the respective parameters in the Plotly `update_layout`, `add_trace`, and `add_annotation` functions.

Normalized Raman Spectra Peak Fitting:

This Python script generates an interactive plot to visualize normalized Raman spectra within a specified wavenumber range (1500-1700 cm^-1). It performs area normalization on the Raman spectra, fits Gaussian and Lorentzian curves to the peaks within the specified range, and displays the average fits on the plot. The plot is created using the Plotly library and can be saved as an HTML file for easy sharing and interaction.

Dependencies:

The following Python libraries are required to run this script:

- NumPy
- Plotly
- SciPy
- scikit-learn

You can install these dependencies using pip:

```
pip install numpy plotly scipy scikit-learn
```

Usage:

1. Prepare text files containing your Raman spectra data. Each file should have one wavenumber-intensity pair per line, separated by whitespace (e.g., tab or space).

2. Update the `text_files` variable in the script with the paths to your text files containing the Raman spectra data.

3. Optionally, you can customize the titles for each spectrum, the peak detection prominence threshold, and the spacing between spectra by modifying the corresponding arguments in the `plot_normalized_raman_peak_range_from_text` function call.

4. Run the script, and it will generate an interactive plot displaying the normalized Raman spectra within the specified wavenumber range (1500-1700 cm^-1), along with the fitted Gaussian and Lorentzian curves.

5. The script will save the interactive plot as an HTML file named `normalized_raman_spectra_1500_1700cm^-1_range.html` in the same directory.

Function Descriptions:

`gaussian(x, a, mu, sigma)`:

This function defines the Gaussian peak function.

`lorentzian(x, a, mu, gamma)`:

This function defines the Lorentzian peak function.

`fit_peak(x, y, peak_function)`:

This function fits the specified peak function (Gaussian or Lorentzian) to the given data and returns the parameters of the fitted peak function, the fitted y-values, and the coefficient of determination (R^2).

`area_normalize_raman(x, y)`:

This function normalizes Raman spectra by dividing the intensities by the area under the curve.

`plot_normalized_raman_peak_range_from_text(text_files, titles=None, prominence_threshold=0.1, spacing=0)`:

This function plots normalized Raman spectra within the specified wavenumber range (1500-1700 cm^-1), along with Gaussian and Lorentzian fits for the peaks in that range. It also calculates and displays the average Gaussian and Lorentzian fits across all spectra.

Customization:

You can customize various aspects of the plot by modifying the corresponding lines in the code. For example, you can change the plot layout, marker styles, line colors, and annotations by updating the respective parameters in the Plotly `update_layout`, `add_trace`, and `add_annotation` functions.