TS-DiffuGen

Release 2023

Sacha Raffaud

Aug 31, 2023

CONTENTS

CHAPTER ONE

INTRODUCTION:

This file contains simple documentation regarding the TS-DiffuGen package.

CHAPTER

TWO

DATASETS:

The dataset classes can be found within the data directory.

2.1 Setting up file:

`data.Dataset_W93.setup_dataset_files.`**`create_pdb_from_file_path`**(*path_to_raw_log*, *path_to_final*)

Create a .XYZ file from a .log DFT file containing atomic coordinates.

This function reads a DFT .log file, extracts atomic symbols and coordinates, and writes them into a .PDB file format.

Parameters

- **`path_to_raw_log`** (*`str`*) – Path to the input DFT .log file.
- **`path_to_final`** (*`str`*) – Path to the output .PDB file to be created.

Returns

None

`data.Dataset_W93.setup_dataset_files.`**`extract_geometry`**(*logs*)

Extract geometry from DFT .log files.

This function extracts atomic symbols and coordinates from a list of lines obtained from a DFT .log file. It searches for the section indicating the standard nuclear orientation and extracts the atom labels and corresponding coordinates in the next line.

Parameters

`logs` (*`list`*) – List of lines from the DFT .log file.

Returns

A tuple containing a list of atomic symbols and a numpy array of atomic coordinates.

Return type

tuple

`data.Dataset_W93.setup_dataset_files.`**`process_reactions`**(*dataframe*, *directory*)

Process reaction data from .log files and create various output files.

This function processes reaction data stored in .log files and organizes the data into clean geometry files in .xyz format. It also generates images of the reactants and products and saves them, along with the geometry files, for each reaction.

Parameters

- **`dataframe`** (*`pd.DataFrame`*) – DataFrame containing reaction data, including reactant and product SMILES.

- **directory** (*str*) – Path to the main directory containing the raw log files.

Returns
None

2.2 Simple W93 Dataset without Graphs:

2.3 W93 Dataset with Graphs:

2.4 TX1 Dataset:

2.5 RGD1 Dataset:

`data.Dataset_RGD1.parse_data.`**`checking_duplicates`**`()`

Check for duplicate reactants and products based on SMILES strings.

This function reads reaction data from an HDF5 file, parses the SMILES strings of reactants and products, and identifies duplicate reactions. It keeps track of duplicate reaction information and their counts.

Returns
None

`data.Dataset_RGD1.parse_data.`**`main`**`()`

Main Function to save reaction geometries in required XYZ format.

This function reads a dataset containing reaction geometries from an HDF5 file, parses the elements and geometries of reactants, products, and transition states, and saves them as XYZ files.

Returns
None

`data.Dataset_RGD1.parse_data.`**`save_reactions_with_multiple_ts`**`(`*save_even_single=False*`)`

Save reactions with multiple TS conformers to separate directories.

This function reads reaction data from an HDF5 file along with activation energy data from a CSV file. It identifies reactions with multiple transition state (TS) conformers based on identical reaction SMILES strings. It then organizes and saves these reactions and their geometries into separate directories.

Parameters
save_even_single (*bool, optional*) – Save reactions with only one TS conformer as well. (default: False)

Returns
None

`data.Dataset_RGD1.parse_data.`**`save_xyz_file`**`(`*path*, *atoms*, *coordinates*`)`

Save XYZ file with atomic elements and corresponding coordinates.

Parameters

- **file_path** (*str*) – Path to the output XYZ file.
- **elements** (*list*) – List of atomic symbols.
- **coordinates** (*numpy.ndarray*) – Array of atomic coordinates.

Returns
None

CHAPTER

THREE

THE EQUIVARIANT GRAPH NEURAL NETWORK:

Background about equivariance and invariance as well as the architecture of the model is described in Sacha's thesis. The main EGNN files used for the diffusion process are the dynamics files:

3.1 Dynamics without Graphs:

3.2 Dynamics with Graphs:

The main difference being that the EGNN takes extra inputs of edge attributes.

3.3 Utility Functions:

The Utility functions used in the EGNN process are outlined bellow:

CHAPTER

FOUR

EXAMPLE FUNCTIONS AND CLASSES FROM THE PACKAGE INCLUDE:

See *src.evaluate_samples* for details.

src.evaluate_samples.**calc_cov_mat**(*results_matrix*, *cov_threshold=0.1*)

Calculate COV and MAT scores based on D-MAE matrix.

Parameters

- **results_matrix** (*np.ndarray*) – D-MAE matrix.
- **cov_threshold** (*float, optional*) – COV threshold. Defaults to 0.1.

Returns

Calculated MAT-R mean, median, and COV-R scores.

Return type

tuple

src.evaluate_samples.**calculate_DMAE**(*gen_mol*, *true_mol*)

Calculate D-MAE between inter-atomic distance matrices.

Parameters

- **gen_mol** (*list*) – Inter-atomic distance matrix of generated molecule.
- **true_mol** (*list*) – Inter-atomic distance matrix of true molecule.

Returns

D-MAE value.

Return type

float

src.evaluate_samples.**calculate_best_rmse**(*gen_mol*, *ref_mol*, *max_iters=100000*, *use_hydrogens=False*)

Calculate Best RMSD between RDKit Molecule Objects.

Parameters

- **gen_mol** (*Chem.Mol*) – RDKit molecule object representing generated molecule.
- **ref_mol** (*Chem.Mol*) – RDKit molecule object representing reference molecule.
- **max_iters** (*int, optional*) – Maximum atom matches. Defaults to 100_000.
- **use_hydrogens** (*bool, optional*) – True to include hydrogens. Defaults to False.

Returns

Best RMSD value.

Return type
float

`src.evaluate_samples.`**`calculate_distance_matrix`**(*coordinates*)

Calculate pairwise distance matrix from 3D coordinates.

Parameters
`coordinates` (*`list`*) – List of 3D coordinates for each atom.

Returns
Pairwise distance matrix.

Return type
np.ndarray

`src.evaluate_samples.`**`create_lists`**(*original_path*, *RMSD=False*)

Create lists of true and generated molecules from the given path.

Parameters

- **`original_path`** (*`str`*) – Path to the original directory containing molecule files.
- **`RMSD`** (*`bool, optional`*) – True if RMSD format, False if standard XYZ format.

Returns
Two lists containing RDKit molecule objects.

Return type
tuple

`src.evaluate_samples.`**`create_table`**(*true_mols*, *gen_mols*, *max_iters=1*, *metric='RMSD'*)

Create comparison table between molecules.

Parameters

- **`true_mols`** (*`list`*) – List of true molecule RDKit objects.
- **`gen_mols`** (*`list`*) – List of lists containing generated molecule RDKit objects.
- **`max_iters`** (*`int`*) – Maximum atom matches for RMSD calculation.
- **`metric`** (*`str`*) – Metric choice, “RMSD” or “DMAE”.

Returns
DataFrame of comparison metrics.

Return type
pd.DataFrame

`src.evaluate_samples.`**`evaluate`**(*sample_path*, *evaluation_type*, *cov_threshold=0.1*)

Evaluate generated samples using RMSE or D-MAE metrics.

Parameters

- **`sample_path`** (*`str`*) – Path to sample directory.
- **`evaluation_type`** (*`str`*) – Metric choice, “RMSD” or “DMAE”.
- **`cov_threshold`** (*`float, optional`*) – COV threshold. Defaults to 0.1.

`src.evaluate_samples.`**`get_paths`**(*sample_path*)

Load molecule files from a sample path and organize them into true and generated samples.

Parameters
`sample_path` (*`str`*) – Path to the sample directory containing molecule files.

Returns
Two lists containing true and generated sample file paths.

Return type
tuple

`src.evaluate_samples.`**`import_xyz_file`**(*molecule_path*, *RMSD=False*)

Import an XYZ file as an RDKit molecule object.

Parameters

- **`molecule_path`** (*`str`*) – File path of the XYZ file to be imported.
- **`RMSD`** (*`bool, optional`*) – True if RMSD format, False if standard XYZ format.

Returns
RDKit molecule object or None if loading fails.

Return type
Chem.Mol or None

PYTHON MODULE INDEX

INDEX