fragmenter Documentation

fragmenter

Aug 09, 2019

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The main purpose of `fragmenter` is to fragment molecules for quantum chemical (QC) torsion drives. Since `fragmenter` is part of the Open Force Field echo system, it also includes other functionality to automate other aspects of the torsion fitting pipeline such as enumerating reasonable tautomers, finding the torsions to drive, generating starting conformations for QC torsion drives and generating JSON input for submission to QCArchive.

The assumption when fragmenting molecules is that the chemistry is localized and that removing or changing remote substituents (defined as substituents more than 2 bonds away from the central bond that is being driven in the torsion drive) will not change the torsion potential around the bond of interest. However, that is not always the case. `fragmenter` uses the Wiberg Bond Order (WBO) as a surrogate signal to determine if the chemistry around the bond of interest was destroyed during fragmentation relative to the bond in the parent molecule.

The WBO is a measure of electronic population overlap between two atoms in a bond. It can be quickly calculated from an empirical QC calculation and is given by:

$$W_{AB} = \sum_{\mu \in A} \sum_{\nu \in B} |D_{\mu\nu}|^2$$

Where A and B are atoms A and B in a bond, D is the density matrix and μ and ν are occupied orbitals on atoms A and B respectively.

`fragmenter` calculates the WBO of the parent molecules, then fragments according to a set of rules and then recalculates the WBO of the fragments. If the WBO for the fragment of the bond of interest changes more than a user's specified threshold, `fragmenter` will add more substituents until the WBO of the bond of interest is within the user specified threshold.

1.1 Contributors

- Chaya D. Stern (MSKCC / Weill Cornell)
- John D. Chodera (MSKCC)

1.2 Acknowledgment

CDS is funded by a fellowship from The Molecular Sciences Software Institute

1.3 References

1. Stern C.D., Smith D.G.A, Bayly C.I., Chodera J.D Fragmenting molecules for QC torsion drives doi 10.5281/zenodo.3238643

CHAPTER

TWO

INSTALLING FRAGMENTER

2.1 Installing using conda

To install `fragmenter` with conda run the following

```
conda install -c omnia fragmenter
```

2.2 Installing from source

To install `fragmenter` from source, clone or download the GitHub repo. From inside the `fragmenter` directory, run the following:

```
python setup.py install
```

This command will not install dependencies. All dependencies are in the `meta.yaml` file

2.3 Prerequisites

`fragmenter` is tested with python 3.6.

This toolkit uses OpenEye as a dependency so licenses for *oechem*, *oequacpac* and *oeomega* are required.

2.4 Warning

`fragmenter` is still pre-alpha. It is not fully tested and the API is still in flux.

CHAPTER

THREE

FRAGMENTER API

`Fragmenter` is pre-alpha so the API is still in flux.

Below is an outline of the API for the main functions of `fragmenter` See *examples* for details on how to use these functions.

3.1 enumerate_states

Ionization can have a profound impact on the electronic structure and the torsion barrier which can lead to different fragmentation decisions. Therefore, it is important to enumerate reasonable ionization states at physiological pH. `fragmenter` provides a wrapper around OpenEye's

3.1.1 fragmenter.states module

`fragmenter.states.`**`enumerate_states`**(*molecule*, *tautomers=True*, *stereoisomers=True*, *verbose=False*, *return_mols=False*, *explicit_h=True*, *return_names=False*, *max_stereo_returns=1*, *filter_nitro=True*, ***kwargs*)

Expand tautomeric state and stereoisomers for molecule.

Parameters

molecule [OEMol] Molecule to enumerate states

tautomers [bool, optional, default True] If False, will not generate tautomers

stereoisomers [bool, optional, default True] If False, will not generate all stereoisomers.

verbose [bool, optional, default False] If True, output will be verbose

return_mols [bool, optional, default False] If True, will return oemols instead of SMILES. Some molecules might be duplicate states

explicit_h [bool, optional, default True] If True, SMILES of states will have explicit hydrogen

return_names [bool, optional, default True] If True, will return names of molecules with SMILES

max_stereo_returns [int, optional, default 1] If stereoisomers is set to False, and the incoming molecule is missing stereo information, OEFlipper will generate stereoisomers for missing stereo center. max_stereo_returns controls how many of those will be returned

** **max_states: int, optional, default 200** This gets passed to *_enumerate_tautomers* and *_enumerate_stereoisomers* max number of states *_enumerate_tautomers* and *_enumerate_stereoisomers* generate

** **pka_norm: bool, optional, default True** This gets passed to *_enumerate_tautomers*. If True, ionization state of each tautomer will be assigned to a predominate state at pH ~7.4

** **warts: bool, optional, default True** This gets passed to *_enumerate_tautomers* and _enumerate_stereoisomers‘ If True, adds a wart to each new state. A ‘wart’ is a systematic

** **force_flip: bool, optional, default True** This gets passed to *_enumerate_stereoisomers* Force flipping all stereocenters. If False, will only generate stereoisomers for stereocenters that are undefined

** **enum_nitorgen: bool, optional, default True** This gets passed to *_enumerate_stereoisomers* If true, invert non-planer nitrogens

Returns

states: list list of oemols or SMILES of states generated for molecule

3.2 fragment

The *fragment* module is the core of `fragmenter`. It provides two ways to fragment molecules:

1. **Combinatorial fragmentation** This scheme generates all possible fragments for a molecule without fragmenting rings and selected functional groups. It is not recommended for general use. It was used to generate the benchmark set used to validate `fragmenter`
2. **WBO fragmentation** This scheme uses the change in WBO in the rotatable bonds to decide if the fragment needs to continue being grown out. The threshold for this change can be provided by the user. The default and recommended threshold is 0.01.

3.2.1 fragmenter.fragment module

class `fragmenter.fragment.`**CombinatorialFragmenter**(*molecule*, *functional_groups=None*)

Bases: `fragmenter.fragment.Fragmenter`

This fragmenter will fragment all bonds besides the ones in rings and sepcified functional groups. Then it will generate all possible connected fragment. This class should only be used to generate validation sets. It is not recommended for general fragmentation because it generates a lot more fragments than is needed.

Parameters

molecule [OEMol] Molecule to fragment.

functional_groups [dict, optional, default None] *{f_group: SMARTS}*. Dictionary that maps the name of a functional group to its SMARTS pattern. These functional groups, if they exist in the molecule, will be tagged so they are not fragmented. If None, will use internal list of functional group. If False, will not tag any functional groups.

Attributes

n_rotors Returns number of rotatable bonds in molecule

Methods

depict_fragments(self, fname[, line_width])	Generate PDF of all combinatorial fragments with individual fragments color coded
fragment(self[, min_rotors, max_rotors, ...])	combinatorial fragmentation.
get_provenance(self)	Get version of fragmenter and options used
to_json(self)	Write out fragments to JSON with provenance

depict_fragments (*self*, *fname*, *line_width=0.75*)
Generate PDF of all combinatorial fragments with individual fragments color coded

Parameters
fname [str] filename for output PDF
line_width [float] width of drawn molecule lines

fragment (*self*, *min_rotors=1*, *max_rotors=None*, *min_heavy_atoms=4*)
combinatorial fragmentation. Fragment along every bond that is not in a ring or specified functional group.

min_rotor: int, optional, default 1 The minimum number of rotatable bond in resutling fragments
max_rotor: int, optional, default None The maximum number of rotatable bond in resulting fragment If None, the maximum number of rotors will be the amount in the parent molecule.
min_heavy_atoms: int, optional, default 4 minimum number of heavy atoms in the resulting fragments

to_json (*self*)
Write out fragments to JSON with provenance

Returns
json_dict: dict JSON dictionary of the fragments to their CMILES identifiers. Keys are canonical SMILES

class fragmenter.fragment.**Fragmenter** (*molecule*)
Bases: object

Base fragmenter class. This class is inherited by CombinatorialFragmenter and WBOFragmenter

Attributes

n_rotors Returns number of rotatable bonds in molecule

Methods

get_provenance(self)	Get version of fragmenter and options used

get_provenance (*self*)
Get version of fragmenter and options used

property n_rotors
Returns number of rotatable bonds in molecule

class fragmenter.fragment.**WBOFragmenter** (*molecule*, *functional_groups=None*, *verbose=False*)
Bases: fragmenter.fragment.Fragmenter

Fragment engine for fragmenting molecules using Wiberg Bond Order

Parameters

molecule [OEMol] Molecule to fragment.

functional_groups [dict, optional, default None] *{f_group: SMARTS}*. Dictionary that maps the name of a functional group to its SMARTS pattern. These functional groups, if they exist in the molecule, will be tagged so they are not fragmented. If None, will use internal list of functional group. If False, will not tag any functional groups.

Attributes

n_rotors Returns number of rotatable bonds in molecule

Methods

calculate_wbo(self[, fragment])	Calculate WBO
depict_fragments(self, fname)	Generate PDF of fragments for the molecule with the rotatable bond highlighted and annotated with its WBO
fragment(self[, threshold, ...])	Fragment molecules using the Wiberg Bond Order as a surrogate
get_bond(self, bond_tuple)	Get bond in molecule by atom indices of atom A and atom B
get_provenance(self)	Get version of fragmenter and options used
to_json(self)	Write out fragments to JSON with provenance
to_qcschema_mols(self, **kwargs)	Writes fragments to a list of qcschema molecules for input to QCArchive computations
to_torsiondrive_json(self, **kwargs)	Generates torsiondrive input JSON for QCArchive

calculate_wbo(*self*, *fragment=None*, ***kwargs*)

Calculate WBO

Parameters

fragment [oechem.OEMol] fragment to recalculate WBO. When fragment is None, fragmenter assumes it's the full molecule and saves the calculated values in self.molecule

Returns

fragment with WBOs

depict_fragments(*self*, *fname*)

Generate PDF of fragments for the molecule with the rotatable bond highlighted and annotated with its WBO

Parameters

fname [str] Filename to write out PDF

fragment(*self*, *threshold=0.01*, *keep_non_rotor_ring_substituents=True*, ***kwargs*)

Fragment molecules using the Wiberg Bond Order as a surrogate

Parameters

keep_non_rotor_ring_substituents: bool If True, will always keep all non rotor substiuents on ring. If False, will only add them if they are ortho to rotatable bond or if it's needed for WBO to be within the threshold

****heuristic** [str, optional, default 'path_length'] The path fragmenter should take when fragment needs to be grown out. The other option is 'wbo'

****threshold** [float, optional, default 0.01] The threshold for the central bond WBO. If the fragment WBO is below this threshold, fragmenter will grow out the fragment one bond at a time via the path specified by the heuristic option

get_bond(*self*, *bond_tuple*)

Get bond in molecule by atom indices of atom A and atom B

Parameters

bond_tuple [tuple] (mapidx, mapidx)

Returns

bond: oechem.OEBondBase The bond in the molecule given by the bond tuple

to_json(*self*)

Write out fragments to JSON with provenance

Returns

json_dict: dict maps fragment SMILES to CMILES identifiers

to_qcschema_mols(*self*, ***kwargs*)

Writes fragments to a list of qcschema molecules for input to QCArchive computations

Parameters

kwargs [parameters for `chemi.generate_conformers`]

Returns

qcschema_fragments: list list of qcschema molecules

to_torsiondrive_json(*self*, ***kwargs*)

Generates torsiondrive input JSON for QCArchive

Returns

torsiondrive_json_dict: dict dictionary with the QCArchive job label as keys that maps to the torsiondrive input for each fragment

CHAPTER

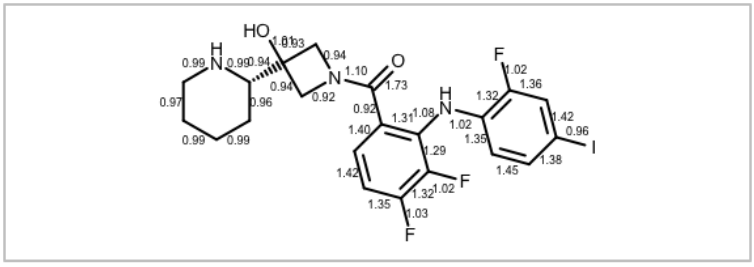
FOUR

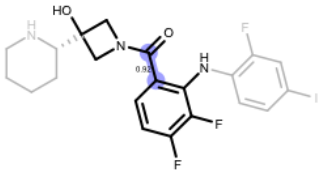
EXAMPLES

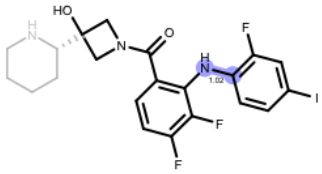
Below is an example on how to use `fragmenter` to fragment a molecule. It is also available in the examples folder

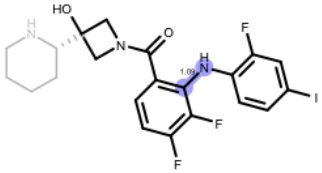
```
from fragmenter import fragment, chemi
import json

"""
This is an example script to generate fragments from a molecule
Note: The current fragmentation scheme is not optimal yet so it can be slow
for larger molecules.
"""

# Create an oemol from a SMILES
oemol = chemi.smiles_to_oemol(
↪'OC1(CN(C1)C(=O)C1=C(NC2=C(F)C=C(I)C=C2)C(F)=C(F)C=C1)[C@@H]1CCCCN1',
                              name='Cobimetinib')

# Instantiate a fragmenter engine
frag_engine = fragment.WBOFragmenter(oemol)
# Use default options to fragment molecule.
frag_engine.fragment()
# Generate PDF with fragments
frag_engine.depict_fragments(fname='example_fragments.pdf')

# Generate input for torsiondrive jobs. These jobs will drive the central rotatable␣
↪bond in the fragment
td_inputs = frag_engine.to_torsiondrive_json(max_confs=10)
with open('example_td_inputs.json', 'w') as f:
    json.dump(td_inputs, f, indent=2, sort_keys=True)

# If you want to only generate starting conformations for other calculations, use `to_
↪qcschema_mols`
qcschema_mols = frag_engine.to_qcschema_mols(max_confs=10)
with open('example_qcschema_mols.json', 'w') as f:
    json.dump(qcschema_mols, f, indent=2, sort_keys=True)
```

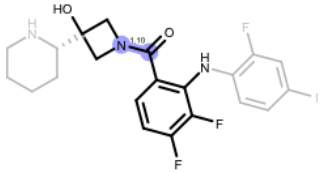
Below are the fragments this script generated and the output you get by calling `fragment_engine.to_pdf()`

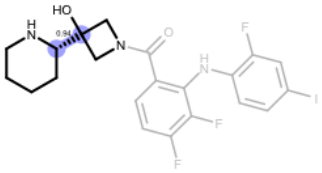












CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX