NAME

RDKitCalculateEnergy.py - Calculate energy

SYNOPSIS

RDKitCalculateEnergy.py [--forceField <UFF, or MMFF>] [--forceFieldMMFFVariant <MMFF94 or MMFF94s>] [--infileParams <Name,Value,...>] [--mp <yes or no>] [--mpParams <Name.Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--quiet <yes or no>] [-w <dir>] -i <infile> -o <outfile>

RDKitCalculateEnergy.py -h | --help | -e | --examples

DESCRIPTION

Calculate single point energy for molecules using a specified forcefield. The molecules must have 3D coordinates in input file.

The supported input file formats are: Mol (.mol), SD (.sdf, .sd)

The supported output file formats are: SD (.sdf, .sd)

OPTIONS

-f, --forceField <UFF, MMFF> [default: MMFF]

Forcefield method to use for energy calculation. Possible values: Universal Force Field (UFF) [Ref 81] or Merck Molecular Mechanics Force Field [Ref 83-87] .

--forceFieldMMFFVariant <MMFF94 or MMFF94s> [default: MMFF94]

Variant of MMFF forcefield to use for energy calculation.

-e, --examples

Print examples.

-h, --help

Print this help message.

-i, --infile <infile>

Input file name.

--infileParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for reading molecules from files. The supported parameter names for different file formats, along with their default values, are shown below:

```
SD, MOL: removeHydrogens,yes,sanitize,yes,strictParsing,yes
```

--mp <yes or no> [default: no]

Use multiprocessing.

By default, input data is retrieved in a lazy manner via mp.Pool.imap() function employing lazy RDKit data iterable. This allows processing of arbitrary large data sets without any additional requirements memory.

All input data may be optionally loaded into memory by mp.Pool.map() before starting worker processes in a process pool by setting the value of 'inputDataMode' to 'InMemory' in '--mpParams' option.

A word to the wise: The default 'chunkSize' value of 1 during 'Lazy' input data mode may adversely impact the performance. The '--mpParams' section provides additional information to tune the value of 'chunkSize'.

--mpParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for to configure multiprocessing.

The supported parameter names along with their default and possible values are shown below:

```
chunkSize, auto
inputDataMode, Lazy   [ Possible values: InMemory or Lazy ]
numProcesses, auto   [ Default: mp.cpu_count() ]
```

These parameters are used by the following functions to configure and control the behavior of multiprocessing: mp.Pool(), mp.Pool.map(), and mp.Pool.imap().

The chunkSize determines chunks of input data passed to each worker process in a process pool by

mp.Pool.map() and mp.Pool.imap() functions. The default value of chunkSize is dependent on the value of 'inputDataMode'.

The mp.Pool.map() function, invoked during 'InMemory' input data mode, automatically converts RDKit data iterable into a list, loads all data into memory, and calculates the default chunkSize using the following method as shown in its code:

```
chunkSize, extra = divmod(len(dataIterable), len(numProcesses) * 4)
if extra: chunkSize += 1
```

For example, the default chunkSize will be 7 for a pool of 4 worker processes and 100 data items.

The mp.Pool.imap() function, invoked during 'Lazy' input data mode, employs 'lazy' RDKit data iterable to retrieve data as needed, without loading all the data into memory. Consequently, the size of input data is not known a priori. It's not possible to estimate an optimal value for the chunkSize. The default chunkSize is set to 1.

The default value for the chunkSize during 'Lazy' data mode may adversely impact the performance due to the overhead associated with exchanging small chunks of data. It is generally a good idea to explicitly set chunkSize to a larger value during 'Lazy' input data mode, based on the size of your input data and number of processes in the process pool.

The mp.Pool.map() function waits for all worker processes to process all the data and return the results. The mp.Pool.imap() function, however, returns the the results obtained from worker processes as soon as the results become available for specified chunks of data.

The order of data in the results returned by both mp.Pool.map() and mp.Pool.imap() functions always corresponds to the input data.

-o, --outfile <outfile>

Output file name.

--outfileParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for writing molecules to files. The supported parameter names for different file formats, along with their default values, are shown below:

```
SD: kekulize,no
```

--overwrite

Overwrite existing files.

-q, --quiet <yes or no> [default: no]

Use quiet mode. The warning and information messages will not be printed.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To calculate single point energy using MMFF forcefield for molecules in a SD file containing 3D structures and write a new SD file, type:

```
% RDKitCalculateEnergy.py  -i Sample3D.sdf -o Sample3DOut.sdf
```

To run the first example in multiprocessing mode on all available CPUs without loading all data into memory and write out a SD file, type:

```
% RDKitCalculateEnergy.py  --mp yes -i Sample3D.sdf -o Sample3DOut.sdf
```

To run the first example in multiprocessing mode on all available CPUs by loading all data into memory and write out a SD file, type:

```
% RDKitCalculateEnergy.py  --mp yes --mpParams "inputDataMode,
  InMemory" -i Sample3D.sdf -o Sample3DOut.sdf
```

To run the first example in multiprocessing mode on specific number of CPUs and chunk size without loading all data into memory and write out a SD file, type:

```
% RDKitCalculateEnergy.py --mp yes --mpParams "inputDataMode,Lazy,
  numProcesses,4,chunkSize,8"  -i Sample3D.sdf -o Sample3DOut.sdf
```

To calculate single point energy using UFF forcefield for molecules in a SD file containing 3D structures and write a new SD file, type:

```
% RDKitCalculateEnergy.py  -f UFF -i Sample3D.sdf -o Sample3DOut.sdf
```

To calculate single point energy using MMFF94s variant of MMFF forcefield for molecules in a SD file containing 3D structures and write a new SD file, type:

```
% RDKitCalculateEnergy.py  --forceField MMFF --forceFieldMMFFVariant
  MMFF94s -i Sample3D.sdf -o Sample3DOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitCalculateRMSD.py, RDKitCalculateMolecularDescriptors.py, RDKitCompareMoleculeShapes.py, RDKitConvertFileFormat.py, RDKitGenerateConformers.py, RDKitPerformMinimization.py

COPYRIGHT

The functionality available in this script is implemented using RDKit, an open source toolkit for cheminformatics developed by Greg Landrum.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.