NAME

RDKitEnumerateTautomers.py - Enumerate tautomers of molecules

SYNOPSIS

RDKitEnumerateTautomers.py [--infileParams <Name,Value,...>] [--mp <yes or no>] [--mpParams <Name,Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--quiet <yes or no>] [--scoreTautomers <yes or no>] [--sortTautomers <yes or no>] [--tautomerizationParams <Name,Value,...>] [-w <dir>] -i <infile> -o <outfile>

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

DESCRIPTION

Enumerate tautomers for molecules and write them out to an output file. The tautomer enumerator generates both protomers and valence tautomers. You may optionally calculate tautomer scores and sort tautomers by SMILES string. The canonical tautomer is placed at the top during sorting.

The supported input file formats are: SD (.sdf, .sd), SMILES (.smi., csv, .tsv, .txt)

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

OPTIONS

-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,no,sanitize,yes,strictParsing,yes
SMILES: smilesColumn,1,smilesNameColumn,2,smilesDelimiter,space,
    smilesTitleLine,auto,sanitize,yes
```

Possible values for smilesDelimiter: space, comma or tab.

--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 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: compute2DCoords,auto,kekulize,yes,forceV3000,no
SMILES: smilesKekulize,no,smilesDelimiter,space, smilesIsomeric,yes,
    smilesTitleLine,yes,smilesMolName,yes,smilesMolProps,no
```

Default value for compute2DCoords: yes for SMILES input file; no for all other file types.

--overwrite

Overwrite existing files.

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

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

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

Calculate and write out tautomer scores [Ref 159].

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

Sort tatutomers of a molecule by SMILES string and place canonical tautomer at the top of the list.

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

A comma delimited list of parameter name and value pairs for enumerating tautomers of molecules. The supported parameter names along with their default values are shown below:

```
tautomerTransformsFile,none,
maxTautomers,1000,maxTransforms,1000,
tautomerRemoveBondStereo,yes,tautomerRemoveIsotopicHs,yes
tautomerRemoveSp3Stereo,yes,tautomerReassignStereo,yes
```

A brief description of the tatutomerization parameters, taken from RDKit documentation, is as follows:

```
tautomerTransformsFile - File containing tautomer transformations

maxTautomers - Maximum number of tautomers to generate
maxTransforms - Maximum number of transforms to apply during
    tautomer enumeration
tautomerRemoveBondStereo - Remove stereochemistry from double bonds
    involved in tautomerism
tautomerRemoveIsotopicHs: Remove isotopic Hs from centers involved in tautomerism
tautomerRemoveSp3Stereo - Remove stereochemistry from sp3 centers
    involved in tautomerism
```

```
tautomerReassignStereo - AssignStereochemistry on all generated tautomers
```

The default value is set to none for the 'tautomerTransformsFile' parameter. The script relies on RDKit to automatically load appropriate tautomer transformations from a set of internal catalog.

The contents of transformation file are described below:

```
tautomerTransformsFile - File containing tautomer transformations

    // Name               SMARTS   Bonds  Charges
    1,3 (thio)keto/enol f  [CX4!H0]-[C]=[O,S,Se,Te;X1]
    1,3 (thio)keto/enol r  [O,S,Se,Te;X2!H0]-[C]=[C]
    1,5 (thio)keto/enol f  [CX4,NX3;!H0]-[C]=[C][CH0]=[O,S,Se,Te;X1]
    ... ... ...
```

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To enumerate tautomers of molecules in a SMILES file and write out a SMILES file, type:

```
% RDKitEnumerateTautomers.py -i Sample.smi -o SampleOut.smi
```

To enumerate tautomers of molecules in a SD file, calculate tautomer scores, sort tautomers, and write out a SD file, type:

```
% RDKitEnumerateTautomers.py --scoreTautomers yes --sortTautomers yes
  -i Sample.sdf -o SampleOut.sdf
```

To enumerate tautomers of molecules in a SD fie , calculate tautomer scores, sort tautomers, and write out a SMILES file, type:

```
% RDKitEnumerateTautomers.py --scoreTautomers yes  --sortTautomers yes
  --outfileParams "smilesMolProps,yes" -i Sample.smi -o SampleOut.smi
```

To enumerate tautomers of molecules in a SD file, performing enumeration in multiprocessing mode on all available CPUs without loading all data into memory, and write out a SD file, type:

```
% RDKitEnumerateTautomers.py --mp yes -i Sample.sdf -o SampleOut.sdf
```

To enumerate tautomers of molecules in a SD file, performing enumeration in multiprocessing mode on specific number of CPUs and chunk size without loading all data into memory, and write out a SD file, type:

```
% RDKitEnumerateTautomers.py --mp yes --mpParams "inputDataMode,Lazy,
  numProcesses,4,chunkSize,8" -i Sample.sdf -o SampleOut.sdf
```

To enumerate tautomers of molecules in a SD file using specific values of parameters to contol the enumeration behavior, and write out a SD file, type:

```
% RDKitEnumerateTautomers.py  -t "maxTautomers,1000,maxTransforms,1000,
  tautomerRemoveBondStereo,yes,tautomerRemoveIsotopicHs,yes,
  tautomerRemoveSp3Stereo,yes,tautomerReassignStereo,yes"
  --scoreTautomers yes --sortTautomers yes -i Sample.sdf -o SampleOut.sdf
```

To enumerate tautomers for molecules in a CSV SMILES file, SMILES strings in column 1, name in column 2, and generate output SD file, type:

```
% RDKitEnumerateTautomers.py --infileParams
  "smilesDelimiter,comma,smilesTitleLine,yes,smilesColumn,1,
  smilesNameColumn,2" --outfileParams "compute2DCoords,yes"
  -i SampleSMILES.csv -o SampleOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitConvertFileFormat.py, RDKitRemoveDuplicateMolecules.py, RDKitRemoveInvalidMolecules.py, RDKitRemoveSalts.py, RDKitSearchFunctionalGroups.py, RDKitSearchSMARTS.py, RDKitStandardizeMolecules.py

COPYRIGHT

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