NAME

RDKitRemoveSalts.py - Remove salts

SYNOPSIS

RDKitRemoveSalts.py [--infileParams <Name,Value,...>] [--mode <remove or count>] [--mp <yes or no>] [--mpParams <Name,Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--saltsMode <ByComponent, BySMARTSFile, BySMARTS>] [--saltsFile <FileName or auto>] [--saltsSMARTS <SMARTS>] [-w <dir>] [-o <outfile>] -i <infile>

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

DESCRIPTION

Remove salts from molecules or simply count the number of molecules containing salts. Salts are identified and removed based on either SMARTS strings or by selecting the largest disconnected components in molecules as non-salt portion of molecules.

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: removeHydrogens,yes,sanitize,yes,strictParsing,yes
SMILES: smilesColumn,1,smilesNameColumn,2,smilesDelimiter,space,
    smilesTitleLine,auto,sanitize,yes
```

Possible values for smilesDelimiter: space, comma or tab.

-m, --mode <remove or count> [default: remove]

Specify whether to remove salts from molecules and write out molecules or or simply count the number of molecules containing salts.

--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.

-s, --saltsMode <ByComponent, BySMARTSFile, BySMARTS> [default: ByComponent]

Specify whether to identify and remove salts based on SMARTS strings or by selecting the largest disconnected component as non-salt portion of a molecule. Possible values: ByComponent, BySMARTSFile or BySMARTS.

--saltsFile <FileName or auto> [default: auto]

Specify a file name containing specification for SMARTS corresponding to salts or use default salts file, Salts.txt, available in RDKit data directory. This option is only used during 'BySMARTSFile' value of '-s, --saltsMode' option.

RDKit data format: Smarts<tab>Name(optional)

For example:

```
[Cl,Br,I]
[N](=O)(O)O
[CH3]C(=O)O   Acetic acid
```

--saltsSMARTS <SMARTS text>

Space delimited SMARTS specifications to use for salts identification instead their specifications in '--saltsFile'. This option is only used during 'BySMARTS' value of '-s, --saltsMode' option.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To remove salts from molecules in a SMILES file by keeping largest disconnected components as non-salt portion of molecules and write out a SMILES file, type:

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

To remove salts from molecules in a SMILES file by keeping largest disconnected components as non-salt portion of molecules, perform salt removal in multiprocessing mode on all available CPUs without loading all data into memory, and write out a SMILES file, type:

```
% RDKitRemoveSalts.py --mp yes -i Sample.smi -o SampleOut.smi
```

To remove salts from molecules in a SMILES file by keeping largest disconnected components as non-salt portion of molecules, perform salt removal in multiprocessing mode on all available CPUs by loading all data into memory, and write out a SMILES file, type:

```
% RDKitRemoveSalts.py --mp yes --mpParams "inputDataMode,InMemory"
  -i Sample.smi -o SampleOut.smi
```

To remove salts from molecules in a SMILES file by keeping largest disconnected components as non-salt portion of molecules, perform salt removal in multiprocessing mode on specific number of CPUs and chunk size without loading all data into memory, and write out a SMILES file, type:

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

To count number of molecules containing salts from in a SD file, using largest components as non-salt portion of molecules, without generating any output file, type:

```
% RDKitRemoveSalts.py -m count -i Sample.sdf
```

To remove salts from molecules in a SMILES file using SMARTS strings in default Salts.txt distributed with RDKit to identify salts and write out a SMILES file, type:

```
% RDKitRemoveSalts.py -m remove -s BySMARTSFile -i Sample.smi
  -o SampleOut.smi
```

To remove salts from molecules in a SD file using SMARTS strings in a local CustomSalts.txt to identify salts and write out a SMILES file, type:

```
% RDKitRemoveSalts.py -m remove -s BySMARTSFile --saltsFile
  CustomSalts.txt -i Sample.sdf -o SampleOut.smi
```

To remove salts from molecules in a SD file using specified SMARTS to identify salts and write out a SD file, type:

```
% RDKitRemoveSalts.py -m remove -s BySMARTS  --saltsSMARTS
  '[Cl,Br,I]  [N](=O)(O)O [N](=O)(O)O'
  -i Sample.sdf -o SampleOut.smi
```

To remove salts form molecules from a CSV SMILES file, SMILES strings in column 1, name in column 2, and generate output SD file, type:

```
% RDKitRemoveSalts.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, RDKitSearchFunctionalGroups.py, RDKitSearchSMARTS.py, RDKitStandardizeMolecules.py

COPYRIGHT