NAME

RDKitPerformSynthonSpaceSearch.py - Perform a synthon space search

SYNOPSIS

RDKitPerformSynthonSpaceSearch.py [--fingerprints <Morgan, PathLength...>] [--fingerprintsParams <Name,Value,...>] [--mode <SubstructureSearch...>] [--outfileParams <Name,Value,...>] [--outfileMode <SingleFile or MultipleFiles>] [--overwrite] [--queryPattern <SMARTS>] [--queryFileParams <Name,Value,...>] [--queryFile <filename>] [--rascalSearchParams <Name,Value,...>] [--substructureMatchParams <Name,Value,...>] [--synthonSearchParams <Name,Value,...>] [-w <dir>] -i <infile> -o <outfile>

RDKitPerformSynthonSpaceSearch.py -l | --list -i <infile>

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

DESCRIPTION

Perform a similarity or substructure search, using query molecules or SMARTS patterns, against a synthon space [Ref 174] in an input file, and write out the hit molecules to output file(s). You may optionally count the hits without building and writing them out.

In addition, you may enumerate a combinatorial library corresponding to a synthon space, generate fingerprints for a synthon space, or list information about a synthon space.

You must provide a valid synthon space text or binary database file supported by RDKit module rdSynthonSpaceSearch.

You may perform similarity search using fingerprints or employ RASCAL (RApid Similarity CALculations using Maximum Edge Subgrahps) methodology [Ref 175].

A number of fingerprints are available for performing similarity search. The similarity metric, however, is calculated using Tanimoto similarity on hashed fingerprints.

The RASCAL similarity between two molecuels is calculated based on MCES (Maximum Common Edge Subgraphs) and corresponds to Johnson similarity.

The supported input file formats are: CSV/TXT synthon space (.csv, .txt) or binary synthon space (.spc).

The supported outfile formats, for different '--mode' values, are shown below:

```
BinaryDBFileGeneration: Binary database file (.spc)
FingerprintsGeneration: Binary database file (.spc)
LibraryEnumeration: SMILES (.smi)
SimilaritySearch or SubstructureSearch: SD (.sdf, .sd), SMILES (.smi),
    CSV/TSV (.csv or .tsv)
```

Possible output files:

```
<OutfileRoot>.<sdf,sd,smi,csv,tsv>

<OutfileRoot>_Mol<Num>.<sdf,sd,smi,csv,tsv>
<OutfileRoot>_Pattern<Num>.<sdf,sd,smi,csv,tsv>

 <OutfileRoot>_HitCount.csv
```

The <OutfileRoot>_HitCount.csv contains aditional information regarding hit counts and is writter out for both similarity and substructure search.

OPTIONS

-f, --fingerprints <Morgan, PathLength...> [default: Morgan]

Fingerprints to use for performing synthon space similarity search. Supported values: AtomPairs, Morgan, MorganFeatures, PathLength, TopologicalTorsions. The PathLength fingerprints are Daylight like fingerprints. The Morgan and MorganFeature fingerprints are circular fingerprints, corresponding Scitegic's

Extended Connectivity Fingerprints (ECFP) and Features Connectivity Fingerprints (FCFP). The values of default parameters for generating fingerprints can be modified using '--fingerprintsParams' option.

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

Parameter values to use for generating fingerprints. The default values are dependent on the value of '-f, --fingerprints' option. In general, it is a comma delimited list of parameter name and value pairs for the name of fingerprints specified using '-f, --fingerprints' option. The supported parameter names along with their default values for valid fingerprints names are shown below:

```
AtomPairs: minLength,1 ,maxLength,useChirality,No,
    use2D, yes, fpSize, 2048
Morgan: radius,2, useChirality,No, useBondTypes, yes,
    useRingMembership, yes, fpSize, 2048
MorganFeatures: radius,2, useChirality,No, useBondTypes, yes,
    useRingMembership, yes, fpSize, 2048
PathLength: minPath,1, maxPath,7, useExplicitHs, yes,
    useBranchedPaths, yes,useBondOrder,yes, fpSize, 2048,
    bitsPerHash,2
TopologicalTorsions: useChirality,No, fpSize, 2048
```

A brief description of parameters, taken from RDKit documentation, is provided below:

```
AtomPairs:

minLength: Minimum distance between atoms.
maxLength: Maximum distance between atoms.
useChirality: Use chirality for atom invariants.
use2D: Use topological distance matrix.
fpSize: Size of the fingerpints bit vector.

Morgan and MorganFeatures:

radius: Neighborhood radius.
useChirality: Use chirality to generate fingerprints.
useBondTypes: Use bond type for the bond invariants.
useRingMembership: Use ring membership.
fpSize: Size of the fingerpints bit vector.

PathLength:

minPath: Minimum bond path length.
maxPath: Maximum bond path length.
useExplicitHs: Use explicit hydrogens.
useBranchedPaths: Use branched paths along with linear paths.
useBondOrder: Us bond order in the path hashes.
fpSize: Size of the fingerpints bit vector.
bitsPerHash: Number of bits set per path.

TopologicalTorsions

useChirality: Use chirality to generate fingerprints.
fpSize: Size of the fingerpints bit vector.
```

-e, --examples

Print examples.

-h, --help

Print this help message.

-i, --infile <infile>

Synthon space Input file name.

-l, --list

List information about synthon space.

-m, --mode <SubstructureSearch...> [default: SimilaritySearch]

Perform similarity or substructure search, enumerate synthon space, or list information about a synthon space. The supported values along with a brief explanation of the expected behavior are shown below:

```
BinaryDBFileGeneration: Write out a binary database file for a
    synthon space.
FingerprintsGeneration: Generate fingerints for a synthon space and
   write out a binary database file along with fingerprints.
LibraryEnumeration: Enumerate a combinatorial library for a synthon
    space and write out a SMILES file.
RASCALSimilaritySearch: Perform a RASCAL (RApid Similarity
    CALculations using Maximum Edge Subgrahps) similarity search.
SimilaritySearch: Perform a similarity search using fingerprints.
SubstructureSearch: Perform a substructure search using specified
    SMARTS patterns.
```

-o, --outfile <outfile>

Output file name. The <OutfileRoot> and <OutfileExt> are used to generate file names during 'MultipleFiles' value for '--outfileMode' option.

--outfileMode <SingleFile or MultipleFiles> [default: SingleFile]

Write out a single file containing hit molecules for substructure or similarity search or generate an individual file for each query pattern or molecule. Possible values: SingleFile or MultipleFiles. The query pattern number or molecule name is written to output file(s). The query pattern or molecule number is also appended to output file names during the generation of multiple output files.

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

A comma delimited list of parameter name and value pairs for writing molecules to files during similarity and substructue search. 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
```

Default value for compute2DCoords: yes for SMILES input file; no for all other file types. The kekulize and smilesIsomeric parameters are also used during generation of SMILES strings for CSV/TSV files.

--queryPattern <SMARTS SMARTS ...> [default: none]

A space delimited list of SMARTS patterns for performing substructure search. This is required for 'SubstructureSearch' value of '--mode' option.

--queryFile <filename> [default: none]

Input file containing query molecules for performing similarity search. This is required for 'SimilaritySearch' value of '--mode' option.

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

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

```
SD, MOL: 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.

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

Parameter values to use for RASCAL similarity search.

The default values are automatically updated to match RDKit default values. The supported parameter names along with their default values are are shown below:

```
allBestMCESs, no, completeAromaticRings, yes,
completeSmallestRings, no, exactConnectionsMatch, no,
ignoreAtomAromaticity, yes, ignoreBondOrders, no,
maxBondMatchPairs, 1000, maxFragSeparation, -1, minCliqueSize, 0,
minFragSize, -1, returnEmptyMCES, false, ringMatchesRingOnly, false,
similarityThreshold, 0.7, singleLargestFrag, no,
timeout, 60
```

A brief description of parameters, taken from RDKit documentation, is provided below:

```
allBestMCESs: Find all Maximum Common Edge Subgraphs (MCES).
completeAromaticRings: Use only complete aromatic rings.
completeSmallestRings: Only complete rings present in both
    molecules.
exactConnectionsMatch: Match atoms only when they have the same
    number of explicit connections.
ignoreAtomAromaticity: Ignore aromaticity during atom matching.
ignoreBondOrders: Ignore bond orders during atom matching.
maxBondMatchPairs: Maximum number of matching bond pairs.
maxFragSeparation: Maximum bond distance that bonds can match.
    value of -1 implies no maximum.
minCliqueSize: A value of > 0 overrides the similarityThreshold.
    This refers to the minimum number of bonds in the MCES.
minFragSize: Minimum number of atoms in a fragment. A value of -1
    implies no minimum.
returnEmptyMCES: Return empty MCES results.
ringMatchesRingOnly: Match ring bonds to only ring bonds.
similarityThreshold: Similarity threshold for matching and
    evaluating MCES.
singleLargestFrag: Find only a single fragment for the MCES. By
    default, multiple fragments are generated as necessary.
timeout: Max run time in seconds. A value of -1 implies no max.
```

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

Parameter values to use for substructure match during synthon substructure search.

The default values are automatically updated to match RDKit default values. The supported parameter names along with their default values are are shown below:

```
aromaticMatchesConjugated, no, maxMatches, 1000,
maxRecursiveMatches, 1000, recursionPossible, yes,
specifiedStereoQueryMatchesUnspecified, no,  uniquify, yes,
useChirality, no, useEnhancedStereo, no, useGenericMatchers, no,
```

A brief description of parameters, taken from RDKit documentation, is provided below:

```
aromaticMatchesConjugated: Match aromatic and conjugated bonds.
maxMatches: Maximum number of matches.
maxRecursiveMatches: Maximum number of recursive matches.
recursionPossible: Allow recursive queries.
specifiedStereoQueryMatchesUnspecified: Match query atoms and bonds
    with specified stereochemistry to atoms and bonds with unspecified
    stereochemistry.
uniquify: Uniquify match results using atom indices.
useChirality: Use chirality to match atom and bonds.
useEnhancedStereo: Use enhanced stereochemistry during the use
    of chirality.
useGenericMatchers: Use generic groups as a post-filtering step.
```

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

Parameter values to use for performing synthon substructure and similarity search.

The default values are automatically updated to match RDKit default values. The supported parameter names along with their default values are are shown below:

```
approxSimilarityAdjuster, 0.1, [ Default value for Morgan FPs ]
```

```
buildHits, yes, fragSimilarityAdjuster, 0.1, hitStart, 0,
maxHits, 1000, [ A value of -1 retrives all hits ]
maxNumFrags, 100000,
numThreads, 1 [ 0: Use maximum number of threads supported by the
    hardware; Negative value: Added to the maxiumum number of
    threads supported by the hardware ]
randomSample, no,
randomSeed, -1 [  Default value implies use random seed ]
similarityCutoff, 0.5, [ Default for Morgan FPs. Ignored during RASCAL
    similarity search; instead, RASCAL parameter similarityThreshold is
    used.  ]
timeOut, 600 [ Unit: sec. The RASCAL searches take longer and may
    need a higher value for timeOut. For example: 3600 ]
```

A brief description of parameters, taken from RDKit documentation, is provided below:

```
approxSimilarityAdjuster: Value used for reducing similarity cutoff
    during approximate similarity check for fingerprint search. A
    lower value leads to faster run times at the risk of missing
    some hits.
buildHits: A no value implies to report the maximum number of hits a
    search could generate without returning any hits.
fragSimilarityAdjuster: Value used for reducing fragment matching
    similarity cutoff to accommodate low bit densities for fragments.
hitStart: Return hits starting from the specified sequence number
    to support retrieval of hits in batches.
maxHits: Maximum number of hits to return. A value of -1 implies
    retrieve all hits.
maxNumFrags: Maximum number of fragments for breaking a query.
numThreads: Number of threads to use for search. A value of 0
    implies the use of all available hardware threads. A negative
    value is added to the number of available hardware threads to
    calculate number of threads to use.
randomSample: Return a random sample of hits up to maxHits.
randomSeed: Random number seed to use during search. A value of -1
    implies the use of a random seed.
similarityCutoff: Similarity cutoff for returning hits by fingerprint
    similarity search. A default value of 0.5 is set for Morgan
    fingeprints.
timeOut: Time limit for search, in seconds. A valus of  0 implies
    no timeout.
```

`--overwrite`

Overwrite existing files.

`-w, --workingdir <dir>`

Location of working directory which defaults to the current directory.

EXAMPLES

To list information about a synthon space in a text file, type:

```
% RDKitPerformSynthonSpaceSearch.py --list -i SampleSynthonSpace.csv
```

To generate a binary database file for a synthon space in a text file, type:

```
% RDKitPerformSynthonSpaceSearch.py -m BinaryDBFileGeneration
  -i SampleSynthonSpace.csv -o SampleSynthonSpace.spc
```

To enumerate a combnatorial library for a synthon space in a text file and write out a SMILES file, type:

```
% RDKitPerformSynthonSpaceSearch.py -m LibraryEnumeration
  -i SampleSynthonSpace.csv -o SampleSynthonSpace_Library.smi
```

To generate Morgan fingerprints for a synthon space in a text file, employing radius of 2 and bit vector size of 2048, and write out a binary database file, type:

```
% RDKitPerformSynthonSpaceSearch.py -m FingerprintsGeneration
  -i SampleSynthonSpace.csv -o SampleSynthonSpace_MorganFPs.spc
```

To perform a similarity search using Morgan fingerprints for query molecules in an input file, against a binary data base file synthon space containing Morgan fingerprints, employing radius 2 and bit vector size of 2048, finding a maximum of 1000 hits for each query molecule, and write out a single output file containing hit molecules, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SimilaritySearch
  -i SampleSynthonSpace_MorganFPs.spc
  --queryFile SampleSynthonSpaceQuery.sdf
  -o SampleSynthonSpace_SimilaritySearchResultsMorganFPs.sdf
```

or only count hits without building hits and writing them to an output file:

```
% RDKitPerformSynthonSpaceSearch.py -m SimilaritySearch
  -i SampleSynthonSpace_MorganFPs.spc
  --queryFile SampleSynthonSpaceQuery.sdf
  -o SampleSynthonSpace_SimilaritySearchResultsMorganFPs.sdf
  --synthonSearchParams "buildHits,No"
```

To run previous example for writing individual output files for each query molecule, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SimilaritySearch
  -i SampleSynthonSpace_MorganFPs.spc
  --queryFile SampleSynthonSpaceQuery.sdf
  -o SampleSynthonSpace_SimilaritySearchResultsMorganFPs.sdf
  --outfileMode MultipleFiles
```

To run previous example for retrieving all possible hits for query molecules and write out individual output files for each query molecules, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SimilaritySearch
  -i SampleSynthonSpace_MorganFPs.spc
  --queryFile SampleSynthonSpaceQuery.sdf
  -o SampleSynthonSpace_SimilaritySearchResultsMorganFPs.sdf
  --outfileMode MultipleFiles
  --synthonSearchParams "maxHits,-1"
```

To run the previous example using multi-threading employing all available threads on your machine, retrieve maximum of 1000 hits for each query molecule and generate various output files, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SimilaritySearch
  -i SampleSynthonSpace_MorganFPs.spc
  --queryFile SampleSynthonSpaceQuery.smi
  -o SampleSynthonSpace_SimilaritySearchResultsMorganFPs.smi
  --outfileMode MultipleFiles
  --synthonSearchParams "maxHits, 1000, numThreads, 0"
```

To run the previous example using multi-threading employing all but one available threads on your machine, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SimilaritySearch
  -i SampleSynthonSpace_MorganFPs.spc
  --queryFile SampleSynthonSpaceQuery.smi
  -o SampleSynthonSpace_SimilaritySearchResultsMorganFPs.smi
  --outfileMode MultipleFiles
  --synthonSearchParams "maxHits, 1000, numThreads, -1"
```

To perform a substructure search using query pattern SMARTS against a synthon space file, finding a maximum of 1000 hits for each query pattern and write out a single output file containing hit molecules, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SubstructureSearch
  -i SampleSynthonSpace.spc
  --queryPattern "c12ccc(C)cc1[nH]nc2C(=O)NCc1cncs1"
  -o SampleSynthonSpace_SubstructureSearchResults.sdf


% RDKitPerformSynthonSpaceSearch.py -m SubstructureSearch
  -i SampleSynthonSpace.csv
  --queryPattern 'c1c[n,s,o][n,s,o,c]c1C(=O)[$(N1CCCCC1),$(N1CCCC1)]'
  -o SampleSynthonSpace_SubstructureSearchResults.sdf
```

To run previous example for retrieving for writing out individual output files for each query molecules, type:

```
% RDKitPerformSynthonSpaceSearch.py -m SubstructureSearch
  -i SampleSynthonSpace.spc
  --queryPattern "CCN(C(=O)c1cc2cc(OC)ccc2nc1C)C1CCCN(C(=O)OC(C)(C)C)C1
  C=CCc1c(N[C@H](C)c2cccc(C)c2)ncnc1N(C)CCCC(=O)OC"
  -o SampleSynthonSpace_SubstructureSearchResults.sdf
  --outfileMode MultipleFiles
```

To perform RASCAL similarity search for query molecules in an input file, against a binary data base file synthon space, finding a maximum of 1000 hits for each query molecule, using multi-threadsing employing all available CPUs, timing out after 3600 seconds, and write out a single output file containing hit molecules, type:

```
% RDKitPerformSynthonSpaceSearch.py -m RASCALSimilaritySearch
  -i SampleSynthonSpace.spc
  --queryFile SampleSynthonSpaceQuery.sdf
  -o SampleSynthonSpace_RASCALSimilaritySearchResults.sdf
  --synthonSearchParams "maxHits, 1000, numThreads, 0, timeOut, 3600"
```

AUTHOR

Manish Sud(msud@san.rr.com)

ACKNOWLEDGMENTS

Dave Cosgrove

SEE ALSO

RDKitConvertFileFormat.py, RDKitPickDiverseMolecules.py, RDKitSearchFunctionalGroups.py, RDKitSearchSMARTS.py

COPYRIGHT