listb Documentation

Release 0.0

Tim B. Herbstrith

Jul 28, 2017

CONTENTS

CHAPTER

ONE

HOWTOS

1.1 Getting the data

I used the following workflow to access all papers of S. Shelah listed on 'MathSciNet'

First I retrieve the MR-numbers. I use the `mrnumbers` command from `mrtools.py`. The option `--url` should point to a search result from 'MathSciNet'. The option `--crawl` specifies, that I want the results from all pages and finally I specify where I want to dump the list of MR-numbers.

```
$ mrtools.py mrnumbers --crawl --dump files/mrnumbers.yaml --url \
> "http://tinyurl.com/shelahmsn"
```

Next I will access the bib-files associated to these numbers. For this purpose I am using the `bib` command.

```
$ mrtools.py bib --load files/mrnumbers.yaml --dump files/msn.bib
```

Now that I have the bibliographic data from 'MathSciNet', I am going to create the merge keys. This can be done using `pybibtools.py` with the `make-key` command. The option `-k` tells the script which fields should be used for the key creation. There are to special values for `-k`, namely `normauthor` and `normtitle`. They call *`listb.normalizetex.norm_author()`* or *`listb.normalizetex.norm_title()`* resp. and add these fields to the bibliography.

```
$ pybibtools.py make-key -k normauthor -k year -k normtitle \
> -o files/norm_msn.bib files/msn.bib
```

Now let's do the same for 'listb'.

```
$ pybibtools.py make-key -k normauthor -k year -k normtitle \
> -o files/norm_listb.bib files/listb.bib
```

This raises a `RuntimeError` if the generated keys are not unique. If so a note is taken in the respective 'info' file.

Finally, we are able to merge the datasets.

```
$ pybibtools.py merge --left -o files/merged.bib \
> files/norm_listb.bib files/norm_msn.bib
```

Note that the `merge` command is *not* commutative.

Here is some data of the first trial run. The last column indicates that 842 entries could be matched.

#	entries	with URL	with MR-no.
listb	1144	0	0
msn	1019	889	1019
merged	1144	752	842

CHAPTER

TWO

SCRIPTS

2.1 mrtools.py

Small command line tools for accessing bibliographic data from 'MathSciNet'.

scripts.mrtools.**chunk_list**(*l*, *n*)

Chops a list into tuples (chunks) of maximal size *n*

Parameters

- **l** (*List[Any]*) – the list to be resized
- **n** (*int*) – maximal size of the chunks

Example

```
>>> chunk_list([1, 2, 3, 4, 5], 2)
[(1, 2), (3, 4), (5,)]
```

2.2 pybibtools.py

Small command line tools for manipulating bibliographies.

scripts.pybibtools.**get_formats**(*f*, *t*, *o*, *files*)

Chooses reader and writer based on user options

Parameters

- **f** (*str*) – name of reader
- **t** (*str*) – name of writer
- **o** (*click.File*) – filehandle pointing to output file
- **files** (*List[str]*) – paths to input files

Returns Name of reader and name of writer

Return type (str, str)

scripts.pybibtools.**load**(*reader*, *fil*)

Common interface for loading with all readers

Parameters

- **reader** (*str*) – name of reader

- **fil** (*str*) – path to input file

Returns Bibliography-object

Return type (Bibliography)

CHAPTER

THREE

MODULES

3.1 mrtools

Python functions for accessing search results from MathSciNet and downloading BibTeX bibliographies associated to the results

listb.mrtools.**crawl**(*url*)

Crawls specified URL on MathSciNet

If the search result is split into 5 pages and the URL to page 3 is passed then the source codes and URLs of pages 3, 4, and 5 are returned.

Parameters **url** (*str*) – URL pointing to a search page on MathSciNet

Returns List of page source codes and list of URLs

Return type (List[str], List[str])

Note: To use this fuction you need to have access to MathSciNet.

listb.mrtools.**get_bibtex_from_msn**(*mrnumbers*, *outfile=None*)

Fetches BibTeX file from MathSciNet server using the MR-numbers

Parameters

- **mrnumbers** (*List[str]*) – the BibTeX entries for these MR-numbers are retrieved
- **outfile** (*Opitonal[str]*) – path to output file

Returns BibTeX file as string

Return type str

Note: To use this fuction you need to have access to MathSciNet.

Example

```
>>> print(get_bibtex_from_msn(['0241312']))
@article {MR0241312,
    AUTHOR = {Shelah, Saharon},
     TITLE = {Note on a min-max problem of {L}eo {M}oser},
   JOURNAL = {J. Combinatorial Theory},
    VOLUME = {6},
```

```
          YEAR = {1969},
         PAGES = {298--300},
       MRCLASS = {05.04},
     MRNUMBER = {0241312},
   MRREVIEWER = {G. F. Clements},
   }
```

listb.mrtools.**get_mrnumber**(*doc*)

Extracts MR-number from the "headlineText" of the search result

Parameters **doc** (*bs4.element.Tag*) – headlineText

Returns MR-number

Return type str

listb.mrtools.**PAT**

_sre.SRE_Pattern – precompiled pattern for extracting the MR-number

listb.mrtools.**msn_to_mrnumbers**(*msn*, *outfile=None*)

Retrieves MR-numbers from the source code of a search page

Parameters

- **msn** (*str OR file handle*) – source code of the search result
- **Optional[str]** (*outfile*) – if specified the MR-numbers get written to a yaml file located at the path

Returns List of MR-numbers found on page

Return type List[str]

Example

```
>>> msn = '''<div class="headlineText">
...            <a class="mrnum" title="Full MathSciNet Item"
...             href="[...]"><strong>MR3549381</strong></a>
...          </div>'''
>>> msn_to_mrnumbers(msn)
['3549381']
```

listb.mrtools.**yaml_dump**(*data*, *path*)

Dumps data into yaml file at *path*

Parameters

- **data** (*Dict[Any], etc.*) – data to be dumped
- **path** (*str*) – path to yaml file

listb.mrtools.**yaml_dumps**(*data*, *handle*)

Dumps data into handle

Parameters

- **data** (*Dict[Any], etc.*) – data to be dumped
- **handle** (*handle*) – handle the data should be dumped into

3.2 normalizetex

Python functions for dealing with with dirty LaTeX ;)

listb.normalizetex.**latex_to_ascii**(*tex*)

Transforms LaTeX strings to ascii text ignoring accents

Parameters **tex** (*str*) – LaTeX string

Returns unicode string containing only ascii characters

Return type str

Examples

```
>>> latex_to_ascii(r"\^ile")
'ile'
>>> latex_to_ascii(r"\^ile") == latex_to_ascii('île')
True
>>> latex_to_ascii(r"Bartoszy\'nski Ros\l anowski")
'Bartoszynski Rosl anowski'
```

listb.normalizetex.**make_key**(*record*, **keys*)

Forms a key from the specified fields of a record

Parameters

- **record** (*Dict[str]*) – record containing specified fields
- **keys** (*List[str]*) – keys of the fields used for key generation

Returns key as string

Return type str

Example

```
>>> record = dict(author='Siegfried Fischbacher and Uwe Ludwig Horn',
...               title=(r'When automorphisms of '
...                      r'{$\Cal P(\kappa)/[\kappa]^{<\aleph_0}$} '
...                      r'are trivial off a small set'),
...               year='2006')
>>> record['normauthor'] = norm_author(record)
>>> make_key(record, 'normauthor', 'year')
'Fischbacher Horn-2006'
```

listb.normalizetex.**norm_author**(*record*)

Transforms the author field into an ordered list of last names

Parameters **record** (*Dict[str]*) – record containing an author field

Returns normalized author names

Return type str

Examples

```
>>> records = [{'author': 'Siegfried Fischbacher and Uwe Ludwig Horn'},
...            {'author': 'Fischbacher, S. and Horn, U.'},
...            {'author': 'Fischbacher, Siegfried and Horn, Uwe '
...                       'Ludwig'}
...           ]
>>> [norm_author(rec) for rec in records]
['Fischbacher Horn', 'Fischbacher Horn', 'Fischbacher Horn']
>>> norm_author({'author': 'François Augiéras'})
'Augieras'
>>> norm_author({'author': 'Avraham (Abraham), Uri and '
...              'Ihoda (Haim Judah), Jaime'})
'Avraham Ihoda'
```

listb.normalizetex.**norm_title**(*record*)

Transforms the title field into a normalized string for matching

Parameters **record** (*Dict[str]*) – record containing a title field

Returns normalized title

Return type str

Example

```
>>> norm_title({'title': r'When automorphisms of '
...                      r'{$\Cal P(\kappa)/[\kappa]^{<\aleph_0}$} '
...                      r'are trivial off a small set'})
'whenautomorphismsofaretrivialoffasmallset'
```

listb.normalizetex.**PAT**

_sre.SRE_Pattern – precompiled pattern matching LaTeX formulae

listb.normalizetex.**WS**

_sre.SRE_Pattern – precompiled pattern matching any non-alphanumeric glyph

3.3 pybibtools

Functions for handling bibliograhic databases

A workflow of special interest is merging to bibliographies. The example below creates two *Bibliography* objects from two databases, that are not disjoint. The first database contains some data we want to keep, but the second database contains some additional information.

Note how in the first call to `merge()` the new dataset contains 3 entries and the "ID" field remains unchanged. In the second call to `merge()` the resulting bibliography has only two entries, namly its initial ones. However, the field "url" from the second database is present.

```
>>> data1 = [{'year': '1981',
...           'title': 'Weak compactness and the structure',
...           'author': 'Sageev, G. and Shelah, S.',
...           'ENTRYTYPE': 'incollection',
...           'ID': 'MR645920'
...          },
...          {'year': '1981',
```

```
...            'title': 'Iterated forcing and changing cofinalities',
...            'author': 'Shelah, Saharon',
...            'ENTRYTYPE': 'article',
...            'ID': 'MR636904'
...           }
...          ]
>>> data2 = [{'year': '1981',
...            'title': 'Iterated forcing and changing cofinalities',
...            'author': 'Shelah, Saharon',
...            'url': 'http://dx.doi.org/10.1090/proc/13163',
...            'ENTRYTYPE': 'article',
...            'ID': 'shelah1981'
...           },
...           {'year': '2016',
...            'title': 'Rigidity of continuous quotients',
...            'author': 'Shelah, Saharon',
...            'ENTRYTYPE': 'article',
...            'ID': 'shelah2016'
...           }
...          ]
>>> bib1 = Bibliography(data1) # Creating bibliographies
>>> bib2 = Bibliography(data2)
>>> bib1.data
[{'year': '1981', 'title': 'Weak compactness and the structure',
'author': 'Sageev, G. and Shelah, S.', 'ENTRYTYPE': 'incollection',
'ID': 'MR645920'}, {'year': '1981',
'title': 'Iterated forcing and changing cofinalities',
'author': 'Shelah, Saharon', 'ENTRYTYPE': 'article', 'ID': 'MR636904'}]
>>> # Creating normalized authors and titles for merging
>>> bib1.add_fields(normauthor=listb.normalizetex.norm_author,
...                 normtitle=listb.normalizetex.norm_title)
>>> bib2.add_fields(normauthor=listb.normalizetex.norm_author,
...                 normtitle=listb.normalizetex.norm_title)
>>> bib1.data[0]['normtitle']
'weakcompactnessandthestructure'
>>> # Let's merge these bibliographies
>>> bib1.make_key('normauthor', 'year', 'normtitle')
>>> bib2.make_key('normauthor', 'year', 'normtitle')
>>> m1 = bib1.merge(bib2, keep_key=True)
>>> m1.data
[{'year': '1981', 'title': 'Weak compactness and the structure', 'author':
'Sageev, G. and Shelah, S.', 'ENTRYTYPE': 'incollection', 'ID': 'MR645920',
'normauthor': 'Sageev Shelah', 'normtitle':
'weakcompactnessandthestructure', 'KEY': 'Sageev
Shelah-1981-weakcompactnessandthestructure'}, {'year': '1981', 'title':
'Iterated forcing and changing cofinalities', 'author': 'Shelah, Saharon',
'url': 'http://dx.doi.org/10.1090/proc/13163', 'ENTRYTYPE': 'article',
'ID': 'MR636904', 'normauthor': 'Shelah', 'normtitle':
'iteratedforcingandchangingcofinalities', 'KEY':
'Shelah-1981-iteratedforcingandchangingcofinalities'}, {'year': '2016',
'title': 'Rigidity of continuous quotients', 'author': 'Shelah, Saharon',
'ENTRYTYPE': 'article', 'ID': 'shelah2016', 'normauthor': 'Shelah',
'normtitle': 'rigidityofcontinuousquotients', 'KEY':
'Shelah-2016-rigidityofcontinuousquotients'}]
>>> # Now we only want to update the first bibliography and
>>> # ignore all other entries
>>> m2 = bib1.merge(bib2, union=False)
>>> m2.data
```

```
[{'year': '1981', 'title': 'Weak compactness and the structure',
'author': 'Sageev, G. and Shelah, S.', 'ENTRYTYPE': 'incollection',
'ID': 'MR645920', 'normauthor': 'Sageev Shelah',
'normtitle': 'weakcompactnessandthestructure'},
{'year': '1981', 'title': 'Iterated forcing and changing cofinalities',
'author': 'Shelah, Saharon', 'ENTRYTYPE': 'article', 'ID': 'MR636904',
'normauthor': 'Shelah',
'normtitle': 'iteratedforcingandchangingcofinalities',
'url': 'http://dx.doi.org/10.1090/proc/13163'}]
```

class listb.pybibtools.**Bibliography**(*data=None*)

Class for handling bibliographic data

MERGEKEY = 'KEY'

Name of the field used for merging in *merge()* and created in *make_key()*.

READERS = {'yaml': <function load at 0x10e4ec578>, 'bib': <function bibtex_load_list at 0x10e678e60>}

Supported readers

WRITERS = {'yaml': <function dump at 0x10e4ec938>, 'bib': <function bibtex_dump at 0x10e678c08>}

Supported writers

add_fields(***kargs*)

Adds fields to bibliography For each entry of `kargs` a field corresponding to the key of the entry is added. The value of the entry must be a unary function accepting an entry of the bibliography as its argument.

Parameters **kargs** (*Dict[str, function]*) – Dictionary of field names and construction functions

Example

In this example the author field is concatenated with itself and stored in the field 'doubleauthor'.

```
>>> data = [{'year': '1981',
...          'title': 'Weak compactness and the structure',
...          'author': 'Sageev, G. and Shelah, S.',
...          'ENTRYTYPE': 'incollection',
...          'ID': 'MR645920'
...         },
...         {'year': '1981',
...          'title': 'Iterated forcing and changing cofinalities',
...          'author': 'Shelah, Saharon',
...          'ENTRYTYPE': 'article',
...          'ID': 'MR636904'
...         }
...        ]
>>> bib = Bibliography(data)
>>> f = lambda entry: entry['author'] * 2
>>> bib.add_fields(doubleauthor=f)
>>> [e['doubleauthor'] for e in bib]
['Sageev, G. and Shelah, S.Sageev, G. and Shelah, S.',
'Shelah, SaharonShelah, Saharon']
```

data

Property containing the bibliographic data `data` must be a list of entries, where each entry is a `dict` containing the keys "ENTRYTYPE" and "ID". These ID-s must be unique.

Raises

- `RuntimeError` – if fields are missing or the ID-s are not unique
- `TypeError` – if `data` is of incorrect type

Example

The first example raises an error since the argument supplied to `data` is not of correct type. The second example succeeds.

```
>>> bib = Bibliography()
>>> bib.data
[]
>>> bib.data = 'Katze'
Traceback (most recent call last):
  ...
TypeError: Expected data as list of bibliographic entries got
<class 'str'>
>>> bib.data = [{'ENTRYTYPE': 'article', 'ID': 'test'}]
```

del_fields(**fields*)

Deletes the specified fields from the database

Parameters **fields** (*List[str]*) – names of fields to be deleted. If an entry does not contain a field with the specified name, nothing happens.

Example

```
>>> data = [{'year': '1981',
...           'title': 'Weak compactness and the structure',
...           'author': 'Sageev, G. and Shelah, S.',
...           'ENTRYTYPE': 'incollection',
...           'ID': 'MR645920'
...          },
...          {'year': '1981',
...           'title': 'Iterated forcing and changing cofinalities',
...           'author': 'Shelah, Saharon',
...           'ENTRYTYPE': 'article',
...           'ID': 'MR636904'
...          }
...         ]
>>> bib = Bibliography(data)
>>> bib.del_fields('title', 'year')
>>> bib.data
[{'author': 'Sageev, G. and Shelah, S.', 'ENTRYTYPE':
'incollection', 'ID': 'MR645920'}, {'author': 'Shelah, Saharon',
'ENTRYTYPE': 'article', 'ID': 'MR636904'}]
```

dump(*writer='yaml'*)

Serializes *data* using one of the predefinded writers in *WRITERS*

Parameters **writer** (*Optional[str]*) – name of one of the predefined writers

Returns representation of *data* as a string.

Return type str

Example

```
>>> data = [{"ENTRYTYPE": "article",
...           "ID": "MR3395349",
...           "author":
...              ("Baldwin, John T. and "
...               "Larson, Paul B. and "
...               "Shelah, Saharon"),
...           "journal": "J. Symb. Log.",
...           "number": "3",
...           "pages": "763--784",
...           "title": r"Almost {G}alois {$\omega$}-stable classes",
...           "volume": "80",
...           "year": "2015",
...         }]
>>> bib = Bibliography(data)
>>> print(bib.dump(writer='bib'))
@article{MR3395349,
 author = {Baldwin, John T. and Larson, Paul B. and Shelah,
           Saharon},
 journal = {J. Symb. Log.},
 number = {3},
 pages = {763--784},
 title = {Almost {G}alois {$\omega$}-stable classes},
 volume = {80},
 year = {2015}
}
```

load(*handle*, *reader='yaml'*)

Loads bibliography from handle

Parameters

- **handle** (*handle*) – file handle of biblography
- **reader** (*Optional[str]*) – name of reader (see *READERS*)

Example

Assuming that the file 'bib.yaml' exists, one can load its data into a bibliography as follows. >>> bib = Bibliography() # doctest: +SKIP >>> with open('bib.yaml', 'r') as handle: # doctest: +SKIP ... bib.load(handle, reader='yaml')

make_key(**keys*)

Creates a merge key formed out of the fields specified in `keys`

Parameters **keys** (*List[str]*) – List of field names

Raises `RuntimeError` – if the merge keys are not unique

Example

Note how the first example produces a `RuntimeError` since the years coincide. Using a combination of author and year fixes this.

```
>>> data = [{'year': '1981',
...           'title': 'Weak compactness and the structure',
```

```
...           'author': 'Sageev, G. and Shelah, S.',
...           'ENTRYTYPE': 'incollection',
...           'ID': 'MR645920'
...          },
...          {'year': '1981',
...           'title': 'Iterated forcing and changing cofinalities',
...           'author': 'Shelah, Saharon',
...           'ENTRYTYPE': 'article',
...           'ID': 'MR636904'
...          }
...         ]
>>> bib = Bibliography(data)
>>> bib.make_key('year')
Traceback (most recent call last):
  ...
RuntimeError: The following merge keys (key, ID)are duplicates:
[('1981', 'MR645920'), ('1981', 'MR636904')]
>>> bib.make_key('author', 'year')
>>> [e['KEY'] for e in bib]
['Sageev, G. and Shelah, S.-1981', 'Shelah, Saharon-1981']
```

merge(*other*, *union=True*, *keep_key=False*)

Merges two bibliographies using the merge key in field *MERGEKEY*

Parameters

- **other** (Bibliography) – The bibliography to be merged
- **union** (*Optional[bool]*) – Do you want the new database to contain the union of the keys? Otherwise only the entries of the left bibliography will be updated and entries not contained in it will be ignored. Defaults to `True`
- **keep_key** (*Optional[bool]*) – Do you want to keep the merge key? Defaults to `False`

Returns Bibliography containing the merged dataset

Return type *Bibliography*

union(*other*)

Returns the union of two bibliographies.

This is a special case of *merge()* were the merge key is just the field 'ID'

Parameters **other** (Bibliography) – bibliography to be joined

Returns union of the bibliographies entries

Return type *Bibliography*

Note: `union` is *not* commutative. See example below.

Example

```
>>> data1 = [{'ENTRYTYPE': 'article', 'ID': 'test1'},
...          {'ENTRYTYPE': 'article', 'ID': 'test2'}]
>>> data2 = [{'ENTRYTYPE': 'book', 'ID': 'test2'},
...          {'ENTRYTYPE': 'article', 'ID': 'test3'}]
```

```
        >>> bib1 = Bibliography(data1)
        >>> bib2 = Bibliography(data2)
        >>> uni = bib1.union(bib2)
        >>> uni.data
        [{'ENTRYTYPE': 'article', 'ID': 'test1'}, {'ENTRYTYPE': 'article',
        'ID': 'test2'}, {'ENTRYTYPE': 'article', 'ID': 'test3'}]
        >>> uni.data == bib2.union(bib1).data
        False
```

listb.pybibtools.**bibtex_dump**(*data*)

Turns dict into BibTex string :param data: data to be transformed :type data: List[dict]

Returns BibTex representation of dict data

Return type str

Example

```
    >>> data = [{"ENTRYTYPE": "article",
    ...          "ID": "MR3395349",
    ...          "author":
    ...             ("Baldwin, John T. and "
    ...              "Larson, Paul B. and "
    ...              "Shelah, Saharon"),
    ...          "journal": "J. Symb. Log.",
    ...          "number": "3",
    ...          "pages": "763--784",
    ...          "title": r"Almost {G}alois {$\omega$}-stable classes",
    ...          "volume": "80",
    ...          "year": "2015",
    ...        }]
    >>> print(bibtex_dump(data))
    @article{MR3395349,
     author = {Baldwin, John T. and Larson, Paul B. and Shelah, Saharon},
     journal = {J. Symb. Log.},
     number = {3},
     pages = {763--784},
     title = {Almost {G}alois {$\omega$}-stable classes},
     volume = {80},
     year = {2015}
    }
```

listb.pybibtools.**bibtex_load_list**(*handle*)

Loads bibtex data from handle :param handle: file handle of bibliography :type handle: handle

Returns entry list of bibliography

Return type List[dict]

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

l

s