paperweight Documentation

Release 0.1.1

Jonathan Sick

January 12, 2015

Contents

Paperweight is a Python package for hacking LaTeX documents.

Paperweight operates on LaTeX documents either on the local filesystem, or as a commit in a git repository. Paperweight can be used to manipulate documents, such as embedded input files and bibliographies, or can provide analytics such as rich information about references made in a document. See the API documentation below for further information.

Paperweight is developed on GitHub under an open BSD license.

CHAPTER 1

Installation

You can install the latest Paperweight and its dependencies from PyPI:

```
pip install paperweight
```

You can also install the bleeding-edge from git:

```
pip install git+git://github.com/jonathansick/paperweight.git
```

Alternatively you can develop directly on the git source repository:

```
git clone https://github.com/jonathansick/paperweight.git
cd paperweight
python setup.py develop
python setup.py build_sphinx
```

CHAPTER 2

Documentation

2.1 Tutorials

2.1.1 Paperweight Basics

This guide provides a brief demo of what you can do with Paperweight. Here's we'll use a LaTeX document that I have handy, but you may want to try the same commands with your own papers.

First, let's set our current working directory to match the paper for convenience.

```
from pprint import pprint
import os
os.chdir("/Users/jsick/Dropbox/m31/writing/skysubpub")
!rm embedded_demo.tex
```

```
rm: embedded_demo.tex: No such file or directory
```

If you're processing TeX documents in an automated fashion, you might not know *a priori* what file contains the root of a LaTeX document. By *root* we mean the file with a `'\documentclass'` declaration. Paperweight provides an API to find this. (Note that `find_root_tex_document()` can accept a directory path as an argument, so you are not restricted to the current working directory)

```
from paperweight import texutils
root_tex_path = texutils.find_root_tex_document()
print(root_tex_path)
```

```
./skysub.tex
```

Working with LaTeX Documents

Now that we know where our LaTeX document is, we can open it as a `FilesystemTexDocument`. Note we also provide a `GitTexDocument` to work with documents stored in an arbitrary commit within a git repo. We'll get to that in a separate demo.

```
from paperweight.document import FilesystemTexDocument
doc = FilesystemTexDocument(root_tex_path)
```

With a `FilesystemTexDocument` you can ask questions about the document, such as what BibTeX file it uses:

```
print(doc.bib_name)
print(doc.bib_path)
```

```
master.bib
/Users/jsick/Library/texmf/bibtex/bib/master.bib
```

or what the names and word-count locations of the sections are:

```
doc.sections
```

```
[(908, u'Introduction'),
 (2601, u'Observations'),
 (4641, u'Image Preparation'),
 (6186, u'Sky Flat Fielding and Median Sky Subtraction'),
 (7762, u'Photometric calibration'),
 (8317, u'Analysis of Sky Flat Fielding and Background Subtraction Methods'),
 (12500, u'Sky Offset Optimization'),
 (14284, u'Analysis of Scalar Sky Offsets'),
 (17032, u'Systematic Uncertainties in Surface Brightness Reconstruction'),
 (18456, u'Conclusions')]
```

or what other tex files it includes using `\input{}` commands:

```
doc.find_input_documents()
```

```
[u'tables/nightset_medsky_offset_hierarchy.tex',
 u'tables/nightset_medsky_scalar_resid_diffs.tex']
```

We can manipulate our LaTeX document too. For instance, we can embed the input TeX files and bibliography directly into the main text body:

```
doc.inline_inputs()
doc.inline_bbl()
```

Now you'll see that we no longer reference other tex files or a bibtex file since all text content is embedded into the root TeX document. This can be handy for submitting the article to a journal (in fact the preprint tool uses paperweight to do just that).

```
print(doc.find_input_documents())
print(doc.bib_name)
```

```
[]
None
```

We can delete comments from the LaTeX source as well. When we do that you'll notice that the sections now appear at earlier word count locations.

```
doc.remove_comments()
```

```
doc.sections
```

```
[(812, u'Introduction'),
 (2505, u'Observations'),
 (4365, u'Image Preparation'),
 (5530, u'Sky Flat Fielding and Median Sky Subtraction'),
 (7011, u'Photometric calibration'),
 (7548, u'Analysis of Sky Flat Fielding and Background Subtraction Methods'),
 (10809, u'Sky Offset Optimization'),
 (12497, u'Analysis of Scalar Sky Offsets'),
 (14942, u'Systematic Uncertainties in Surface Brightness Reconstruction'),
 (16082, u'Conclusions')]
```

You can easily write the modified TeX source back to the filesystem with the `write()` method:

```
doc.write("embedded_demo.tex")
```

Extracting Citation Information

One of the goals of Paperweight is to allow us to *understand* our scientific documents. A big part of that is understanding how we cite other papers.

With our document, we can ask for what references are made in the document according to the cite keys used in `\cite*{}` commands:

```
doc.bib_keys
```

```
[u'Saglia:2010',
 u'Courteau:2014',
 u'Courteau:2011',
 u'Athanassoula:2006',
 u'Nelder:1965',
 u'de-Jong:1996b',
 u'Maraston:1998',
 u'Marigo:2008',
 u'Conroy:2010b',
 u'Barmby:2006',
 u'Pforr:2012',
 u'Maraston:2005',
 u'Maraston:2006',
 u'Williams:2003',
 u'Williams:2002',
 u'Bruzual:2007',
 u'Brown:2009a',
 u'Press:2007',
 u'Olsen:2006',
 u'Bertin:2002',
 u'McConnachie:2005',
 u'McConnachie:2009',
 u'Bertin:1996',
 u'Bertin:2006',
 u'Brown:2008',
 u'Adams:1996',
 u'Dalcanton:2012',
 u'Brown:2003',
 u'Dutton:2005',
 u'Sick:2013a',
 u'Marmo:2008',
 u'Massey:2006',
 u'Irwin:2005',
 u'Skrutskie:2006',
 u'MacArthur:2004',
 u'Conroy:2013',
 u'Ibata:2005',
 u'Berriman:2008',
 u'Kormendy:2004',
 u'Taylor:2011',
 u'Beaton:2007',
 u'Puget:2004',
 u'Brown:2006',
 u'Vaduvescu:2004',
 u'Worthey:2005']
```

This is useful, but we can go deeper by understanding the context in which these works are cited. To do this we can use the `extract_citation_context()` method to generate a dictionary, keyed by bib keys, of all citation instances in the document. In this example paper I've cited 45 works:

```
cites = doc.extract_citation_context()
print(len(cites))
```

```
45
```

Each entry in the `cites` dictionary is a list of specific occurences where that work was cited. Thus its easy to count the number of times each work was cited:

```
for cite_key, instances in cites.iteritems():
    print("{0} cited {1:d} time(s)".format(cite_key, len(instances)))
```

```
Saglia:2010 cited 1 time(s)
Courteau:2014 cited 2 time(s)
Courteau:2011 cited 1 time(s)
Athanassoula:2006 cited 1 time(s)
Nelder:1965 cited 1 time(s)
de-Jong:1996b cited 1 time(s)
Maraston:1998 cited 1 time(s)
Marigo:2008 cited 1 time(s)
Conroy:2010b cited 1 time(s)
Barmby:2006 cited 5 time(s)
Pforr:2012 cited 1 time(s)
Maraston:2005 cited 1 time(s)
Maraston:2006 cited 1 time(s)
Williams:2003 cited 1 time(s)
Williams:2002 cited 1 time(s)
Bruzual:2007 cited 1 time(s)
Brown:2009a cited 1 time(s)
Press:2007 cited 1 time(s)
Olsen:2006 cited 1 time(s)
Bertin:2002 cited 1 time(s)
McConnachie:2005 cited 1 time(s)
McConnachie:2009 cited 1 time(s)
Bertin:1996 cited 2 time(s)
Bertin:2006 cited 1 time(s)
Brown:2008 cited 1 time(s)
Adams:1996 cited 4 time(s)
Dalcanton:2012 cited 2 time(s)
Brown:2003 cited 1 time(s)
Dutton:2005 cited 1 time(s)
Sick:2013a cited 1 time(s)
Marmo:2008 cited 1 time(s)
Massey:2006 cited 1 time(s)
Irwin:2005 cited 1 time(s)
Skrutskie:2006 cited 2 time(s)
MacArthur:2004 cited 1 time(s)
Conroy:2013 cited 1 time(s)
Ibata:2005 cited 1 time(s)
Berriman:2008 cited 2 time(s)
Kormendy:2004 cited 1 time(s)
Taylor:2011 cited 1 time(s)
Beaton:2007 cited 2 time(s)
Puget:2004 cited 2 time(s)
Brown:2006 cited 1 time(s)
Vaduvescu:2004 cited 7 time(s)
```

```
Worthey:2005 cited 1 time(s)
```

It looks like I've cited `Vaduvescu:2004` a lot. Lets look at where it was cited:

```
print([c['section'] for c in cites['Vaduvescu:2004']])
```

```
[(812, u'Introduction'), (2505, u'Observations'), (2505, u'Observations'), (2505, u'Observations'),
```

In the list above, the first item lists the cumulative word count where the section starts, while the second item is the name of the section.

There's a lot of other information associated with each citation instance. Here's metadata associated with the first reference to `Vaduvescu:2004`:

```
pprint(cites['Vaduvescu:2004'][0])
```

```
{'position': 2235,
 'section': (812, u'Introduction'),
 'wordsafter': u'also found detector systems , case ( decommissioned ) CFHT - IR camera , add time -
 'wordsbefore': u'Spatial structures NIR sky leave residual shapes background subtracted disk images
```

2.2 API Reference

2.2.1 paperweight.document

Object oriented access to LaTeX documents.

The `paperweight.document` module provides object-oriented interfaces for manipulating or mining LaTeX documents. Much of the functionality of the `paperweight.texutils`, `paperweight.gitio` and `paperweight.nlputils` modules can be accessed through this interface.

Depending on how the LaTeX document is stored, you should use either of two document classes. `paperweight.document.FilesystemTexDocument` should be used for regular documents in the filesystem. If you wish to operate on documents stored within a certain commit of a checked-out Git repository, then use `paperweight.document.GitTexDocument`. The interfaces for both classes are consistent since they inherit from `paperweight.document.TexDocument` under the hood.

class `paperweight.document.`**`FilesystemTexDocument`**(*path*, *recursive=True*)

Bases: `paperweight.document.TexDocument`

A TeX document derived from a file in the filesystem.

Parameters **filepath** : unicode

Path to the '.tex' on the filesystem.

recursive : bool

If *True* (default), then tex documents input by this root document will be opened.

Attributes

`bib_keys`	List of all bib keys in the document (and input documents).
`bib_name`	Name of the BibTeX bibliography file (e.g., `'mybibliography.bib'`).
`bib_path`	Absolute file path to the .bib bibliography document.

Continued on next page

Table 2.1 – continued from previous page

bibitems	List of bibitem strings appearing in the document.
sections	List with tuples of section names and positions.

Methods

extract_citation_context([n_words])	Generate a dictionary of all bib keys in the document (and input documents), with r
find_input_documents()	Find all tex documents input by this root document.
inline_bbl()	Inline a compiled bibliography (.bbl) in place of a bibliography environment.
inline_inputs()	Inline all input latex files references by this document.
remove_comments([recursive])	Remove latex comments from document (modifies document in place).
write(path)	Write the document's text to a path on the filesystem.

bib_keys
List of all bib keys in the document (and input documents).

bib_name
Name of the BibTeX bibliography file (e.g., 'mybibliography.bib').

bib_path
Absolute file path to the .bib bibliography document.

bibitems
List of bibitem strings appearing in the document.

extract_citation_context(*n_words=20*)
Generate a dictionary of all bib keys in the document (and input documents), with rich of metadata about the context of each citation in the document.

For example, suppose 'Sick:2014' is cited twice within a document. Then the dictionary returned by this method will have a length-2 list under the 'Sick:2014' key. Each item in this list will be a dictionary providing metadata of the context for that citation. Fields of this dictionary are:

- position: (int) the cumulative word count at which the citation occurs.
- wordsbefore: (unicode) text occuring before the citation.
- wordsafter: (unicode) text occuring after the citation.
- section: (unicode) name of the section in which the citation occurs.

Parameters **n_words** : int

Number of words before and after the citation to extract for context.

Returns **bib_keys** : dict

Dictionary, keyed by BibTeX cite key, where entires are lists of instances of citations. See above for the format of the instance metadata.

find_input_documents()
Find all tex documents input by this root document.

Returns **paths** : list

List of filepaths for input documents. Paths are relative to the document (i.e., as written in the latex document).

inline_bbl()
Inline a compiled bibliography (.bbl) in place of a bibliography environment. The document is modified in place.

inline_inputs()
Inline all input latex files references by this document. The inlining is accomplished recursively. The document is modified in place.

remove_comments(*recursive=True*)
Remove latex comments from document (modifies document in place).

Parameters **recursive** : bool

Remove comments from all input LaTeX documents (default `True`).

sections
List with tuples of section names and positions. Positions of section names are measured by cumulative word count.

write(*path*)
Write the document's text to a `path` on the filesystem.

class `paperweight.document.`**GitTexDocument**(*git_path*, *git_hash*, *repo_dir='.'*, *recursive=True*)
Bases: `paperweight.document.TexDocument`

A tex document derived from a file in the git repository.

Parameters **git_path** : str

Path to the document in the git repository, relative to the root of the repository.

git_hash : str

Any SHA or git tag that can resolve into a commit in the git repository.

repo_dir : str

Path from current working directory to the root of the git repository.

Attributes

`bib_keys`	List of all bib keys in the document (and input documents).
`bib_name`	Name of the BibTeX bibliography file (e.g., `'mybibliography.bib'`).
`bib_path`	Absolute file path to the .bib bibliography document.
`bibitems`	List of bibitem strings appearing in the document.
`sections`	List with tuples of section names and positions.

Methods

`extract_citation_context`([n_words])	Generate a dictionary of all bib keys in the document (and input documents), with r
`find_input_documents`()	Find all tex documents input by this root document.
`remove_comments`([recursive])	Remove latex comments from document (modifies document in place).
`write`(path)	Write the document's text to a `path` on the filesystem.

bib_keys
List of all bib keys in the document (and input documents).

bib_name

Name of the BibTeX bibliography file (e.g., `'mybibliography.bib'`).

bib_path
Absolute file path to the .bib bibliography document.

bibitems
List of bibitem strings appearing in the document.

extract_citation_context(*n_words=20*)
Generate a dictionary of all bib keys in the document (and input documents), with rich of metadata about the context of each citation in the document.

For example, suppose `'Sick:2014'` is cited twice within a document. Then the dictionary returned by this method will have a length-2 list under the `'Sick:2014'` key. Each item in this list will be a dictionary providing metadata of the context for that citation. Fields of this dictionary are:

- `position`: (int) the cumulative word count at which the citation occurs.
- `wordsbefore`: (unicode) text occuring before the citation.
- `wordsafter`: (unicode) text occuring after the citation.
- `section`: (unicode) name of the section in which the citation occurs.

Parameters **n_words** : int

Number of words before and after the citation to extract for context.

Returns **bib_keys** : dict

Dictionary, keyed by BibTeX cite key, where entires are lists of instances of citations. See above for the format of the instance metadata.

find_input_documents()
Find all tex documents input by this root document.

Returns **paths** : list

List of filepaths for input documents. Paths are relative to the document (i.e., as written in the latex document).

remove_comments(*recursive=True*)
Remove latex comments from document (modifies document in place).

Parameters **recursive** : bool

Remove comments from all input LaTeX documents (default `True`).

sections
List with tuples of section names and positions. Positions of section names are measured by cumulative word count.

write(*path*)
Write the document's text to a `path` on the filesystem.

class paperweight.document.**TexDocument**(*text*)
Bases: object

Baseclass for a tex document.

Parameters **text** : unicode

Unicode-encoded text of the latex document.

Attributes

text	(unicode) Text of the document as a unicode string.

Methods

extract_citation_context([n_words])	Generate a dictionary of all bib keys in the document (and input documents), with r
find_input_documents()	Find all tex documents input by this root document.
remove_comments([recursive])	Remove latex comments from document (modifies document in place).
write(path)	Write the document's text to a path on the filesystem.

bib_keys
List of all bib keys in the document (and input documents).

bib_name
Name of the BibTeX bibliography file (e.g., 'mybibliography.bib').

bib_path
Absolute file path to the .bib bibliography document.

bibitems
List of bibitem strings appearing in the document.

extract_citation_context(*n_words=20*)
Generate a dictionary of all bib keys in the document (and input documents), with rich of metadata about the context of each citation in the document.

For example, suppose 'Sick:2014' is cited twice within a document. Then the dictionary returned by this method will have a length-2 list under the 'Sick:2014' key. Each item in this list will be a dictionary providing metadata of the context for that citation. Fields of this dictionary are:

- position: (int) the cumulative word count at which the citation occurs.
- wordsbefore: (unicode) text occuring before the citation.
- wordsafter: (unicode) text occuring after the citation.
- section: (unicode) name of the section in which the citation occurs.

Parameters **n_words** : int

Number of words before and after the citation to extract for context.

Returns **bib_keys** : dict

Dictionary, keyed by BibTeX cite key, where entires are lists of instances of citations. See above for the format of the instance metadata.

find_input_documents()
Find all tex documents input by this root document.

Returns **paths** : list

List of filepaths for input documents. Paths are relative to the document (i.e., as written in the latex document).

remove_comments(*recursive=True*)
Remove latex comments from document (modifies document in place).

Parameters **recursive** : bool

Remove comments from all input LaTeX documents (default `True`).

sections
List with tuples of section names and positions. Positions of section names are measured by cumulative word count.

write(*path*)
Write the document's text to a `path` on the filesystem.

2.2.2 paperweight.texutils

Utilities for maniuplating latex documents.

paperweight.texutils.**find_root_tex_document**(*base_dir='.'*)
Find the tex article in the current directory that can be considered a root. We do this by searching contents for `'\documentclass'`.

Parameters **base_dir** : str

Directory to search for LaTeX documents, relative to the current working directory.

Returns **tex_path** : str

Path to the root tex document relative to the current working directory.

paperweight.texutils.**iter_tex_documents**(*base_dir='.'*)
Iterate through all .tex documents in the current directory.

paperweight.texutils.**inline**(*root_text*, *base_dir=''*, *replacer=None*, *ifexists_replacer=None*)
Inline all input latex files.

The inlining is accomplished recursively. All files are opened as UTF-8 unicode files.

Parameters **root_txt** : unicode

Text to process (and include in-lined files).

base_dir : str

Base directory of file containing `root_text`. Defaults to the current working directory.

replacer : function

Function called by `re.sub()` to replace `\input` expressions with a latex document. Changeable only for testing purposes.

ifexists_replacer : function

Function called by `re.sub()` to replace `\InputIfExists` expressions with a latex document. Changeable only for testing purposes.

Returns **txt** : unicode

Text with referenced files included.

paperweight.texutils.**inline_blob**(*commit_ref*, *root_text*, *root_path*, *repo_dir=''*)
Inline all input latex files that exist as git blobs in a tree object.

The inlining is accomplished recursively. All files are opened as UTF-8 unicode files.

Parameters **commit_ref** : str

String identifying a git commit/tag.

root_text : unicode

Text of tex document where referenced files will be inlined.

root_path : str

Path of file containing root_text relative to the git directory.

repo_dir : str

Directory of the containing git repository.

Returns **txt** : unicode

Text with referenced files included.

paperweight.texutils.**inline_bbl**(*root_tex*, *bbl_tex*)

Inline a compiled bibliography (.bbl) in place of a bibliography environment.

Parameters **root_tex** : unicode

Text to process.

bbl_tex : unicode

Text of bibliography file.

Returns **txt** : unicode

Text with bibliography included.

paperweight.texutils.**remove_comments**(*tex*)

Delete latex comments from a manuscript.

Parameters **tex** : unicode

The latex manuscript

Returns **tex** : unicode

The manuscript without comments.

2.2.3 paperweight.gitio

Utilities for reading content in git repositories.

paperweight.gitio.**read_git_blob**(*commit_ref*, *path*, *repo_dir='.'*)

Get text from a git blob.

Parameters **commit_ref** : str

Any SHA or git tag that can resolve into a commit in the git repository.

path : str

Path to the document in the git repository, relative to the root of the repository.

repo_dir : str

Path from current working directory to the root of the git repository.

Returns **text** : unicode

The document text.

paperweight.gitio.**absolute_git_root_dir**(*fpath=''*)

Absolute path to the git root directory containing a given file or directory.

2.2.4 paperweight.nlputils

Utility functions for working with NLTK

paperweight.nlputils.**wordify**(*text*)

Generate a list of words given text, removing punctuation.

Parameters **text** : unicode

A piece of english text.

Returns **words** : list

List of words.

CHAPTER 3

Indices

- *genindex*
- *modindex*
- *search*

CHAPTER 4

Contributors

- Jonathan Sick
- Matthew Sottile

Paperweight was created as a hack for .Astronomy 6 (2014) in Chicago.

p

A

B

E

F

G

I

P

R

S

T

W