afp Documentation

Release 1

Christopher Phillippi
## Contents

1 **afp package** 3
   1.1 Submodules .................................................. 3
   1.2 afp.count module ............................................... 3
   1.3 afp.keywords module .......................................... 3
   1.4 afp.linalg module ............................................. 4
   1.5 afp.matrices module ........................................... 4
   1.6 afp.normalize module ......................................... 4
   1.7 afp.settings module .......................................... 5
   1.8 Module contents ............................................. 5

2 **cleaner package** 7
   2.1 Submodules .................................................... 7
   2.2 cleaner.filers module ......................................... 7
   2.3 cleaner.helpers module ....................................... 8
   2.4 cleaner.retrieve module ...................................... 8
   2.5 cleaner.schema module ....................................... 9
   2.6 cleaner.settings module ..................................... 10
   2.7 cleaner.store module ........................................ 10
   2.8 Module contents ............................................. 11

3 **Indices and tables** 13

Python Module Index 15
Contents:
1.1 Submodules

1.2 afp.count module

1.3 afp.keywords module

This module handles the keyword mappings required when counting words in articles for matrix generation

```python
afp.keywords.formatAlias(alias)
    Formats a string in Alias format
    >>> formatAlias('A Particular Company')
    'a particular company'
```

```python
afp.keywords.formatName(name)
    Formats a string in Name format
    >>> formatName(r"someone’s financial GROUP")
    'Someone’s Financial Group'
```

```python
afp.keywords.formatTicker(ticker)
    Formats a string in Ticker format
    >>> formatTicker('boog')
    'BOOG'
```

```python
afp.keywords.getAliasToKeywordMap(csvFile='/home/docs/Dropbox/AFPdb/Keywords/keywords.csv')
    Returns a dictionary with aliases as keys and keywords as values given keywords csvFile path
```

Parameters

- `csvFile` – The path of the keywords .csv file

```python
afp.keywords.getIndexToFieldsMap(csvFile='/home/docs/Dropbox/AFPdb/Keywords/keywords.csv')
    Returns a dictionary with indices as keys and fields as values given keywords csvFile path
```

Parameters

- `csvFile` – The path of the keywords .csv file

```python
afp.keywords.getKeywordToFieldsMap(csvFile='/home/docs/Dropbox/AFPdb/Keywords/keywords.csv')
    Returns a dictionary with keywords as keys and fields as values given keywords csvFile path
```

Parameters

- `csvFile` – The path of the keywords .csv file
afp.keywords.getKeywordToIndexMap(csvFile='/home/docs/Dropbox/AFPdb/Keywords/keywords.csv')

Returns a dictionary with keywords as keys and indices at values given keywords csvFile path

Parameters csvFile – The path of the keywords .csv file

afpkeywords.getTickerList(csvFile='/home/docs/Dropbox/AFPdb/Keywords/keywords.csv')

1.4 afp.linalg module

This module contains the sparse matrix operations required for processing that do not reside in scipy/numpy

afp.linalg.corr(sparse, isSample=True)

Computes a correlation matrix for very large sparse matrices with small memory overhead. If speed is of upmost concern, just use numpy.corrcoef().

Parameters sparse(scipy.sparse.csr_matrix() or similar sparse matrix) – Sparse matrix (m x n)

afp.linalg.cov(sparse, isSample=True)

Computes a covariance matrix for very large sparse matrices with small memory overhead. If speed is of upmost concern, just use numpy.cov().

Parameters sparse(scipy.sparse.csr_matrix() or similar sparse matrix) – Sparse matrix (m x n)

afp.linalg.return_corr(price_dataframe, isSample=True)

1.5 afp.matrices module

1.6 afp.normalize module

This module contains the functors which normalize all types of inputs, from matrices to articles, to an expected format.

class afp.normalize.Article
Bases: afp.normalize.NormalizerBase

Functor normalizing articles to be searched for keywords

Replaces the following:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>'s'</td>
<td>''</td>
<td>boogle's</td>
<td>boogle</td>
</tr>
<tr>
<td>'n't'</td>
<td>' not'</td>
<td>didn't</td>
<td>did not</td>
</tr>
<tr>
<td>'/'</td>
<td>' or '</td>
<td>addle/boogle</td>
<td>addle or boogle</td>
</tr>
</tbody>
</table>

normalize(article)

Normalizes article to be outputed in iterable of words where each word has been replaced according to member: wordReplacements. Also forces all-lowercase.

wordReplacements = ["("s", ", ","("n"t", ", not"), ("'/", " or ")]

class afp.normalize.Normalize
Bases: object

Base class for Normalizing Functors

Extending Requires method: normalize()
normalize(matrix)

class afp.normalize.TfIdf
    Bases: afp.normalize.NormalizerBase
    
    Functor normalizing count matrices to tf-idf matrices

    normalize(counts)
    
    Normalizes count matrix into tf-idf matrix

    Parameters
    
    counts (numpy matrix, sparse scipy matrix, or Pandas Dataframe) – Counts of words in each article. Elements may be negative. Sign is passed-through.

1.7 afp.settings module

Contains the constants and settings for the afp package.

1.8 Module contents
2.1 Submodules

2.2 cleaner.filers module

This module contains the lower level API that handles the storing to the filesystem.

```python
class cleaner.filers.ArticleFilerBase
    Bases: object

    Base API to store an article according to regex members. Use this as a base for custom schema. See cleaner.schema.LexisNexisSchema as an example.

    Member paperDateTitleRegex  Regex retrieving ( paper, date, title ), compiled with re.compile()

    Member dateRegex  Regex retrieving ( month, day, year ), compiled with re.compile()

    Member removeFromTitleRegex  Regex class of any letter to be removed from the article title, compiled with re.compile()

    Member removeFromArticleRegex  Regex retrieving newlines in sentences, compiled with re.compile()

    Member schemaName  Name of schema to use, should be the same as one registered in cleaner.schema.getSchema()

    Member sectionDelimiter  String to use as delimiter between sections of the article.
```

dateRegex = None
getFileName (title)
    Processes filename for article to be stored

    Parameters title – Article title to be incorporated in filename.

class ArticleFilerBase
    Bases: object

    Base API to store an article according to regex members. Use this as a base for custom schema. See cleaner.schema.LexisNexisSchema as an example.

    Member paperDateTitleRegex  Regex retrieving ( paper, date, title ), compiled with re.compile()

    Member dateRegex  Regex retrieving ( month, day, year ), compiled with re.compile()

    Member removeFromTitleRegex  Regex class of any letter to be removed from the article title, compiled with re.compile()

    Member removeFromArticleRegex  Regex retrieving newlines in sentences, compiled with re.compile()

    Member schemaName  Name of schema to use, should be the same as one registered in cleaner.schema.getSchema()

    Member sectionDelimiter  String to use as delimiter between sections of the article.
```
write (article)

class cleaner.filers.BatchFiler (schema)
    Bases: object
    API to retrieve data from a given download batch

    write (filename)
    Writes a downloaded batch file to the cleaned folder given schemaName’s article filer

    Parameters filename – Filename of batch to read from.

    writeAll (articles)

class cleaner.filers.CouchDBFiler (dbName='articles', server='http://localhost:5984', *args, **kwargs)
    Bases: cleaner.filers.BatchFiler

    writeAll (articles)

class cleaner.filers.FilerResult (added=None, article=None, fileName=None, filePath=None)
    Bases: object
    Result of an attempted article filing

    Prints as:
    • file and filepath if successful
    • the article if failed to store

2.3 cleaner.helpers module

   Helper functions for cleaner

    cleaner.helpers.ensurePath (filepath)
    Create a path if it doesn’t not currently exist, else does nothing

    cleaner.helpers.flatten (deepIterable)
    Flatten iterables of iterables of ... to list with depth of 1

    cleaner.helpers.tryexcept (expression, replaceWith)
    Evaluate expression, if it throws an exception return replaceWith

2.4 cleaner.retrieve module

   High level API to retrieve cleaned files

    cleaner.retrieve.adjustedClosesFilepath (empiricalStore='/home/docs/Dropbox/AFPdb/Empirical', filename='adjustedClose.csv')

    cleaner.retrieve.benchmarkFilepath ()

    cleaner.retrieve.getCleanArticles (fileList)

    cleaner.retrieve.getCleanArticles (cleanStore='/home/docs/AFPCorpus')
    Returns iterable of all cleaned articles

    Parameters cleanStore – Absolute path to clean store
cleaner.retrieve.getCleanFileList(cleanStore='/home/docs/AFPCorpus')
Returns iterable of all cleaned files

Parameters
- **cleanStore** – Absolute path to clean store

cleaner.retrieve.getDailyArticles(date, cleanStore='/home/docs/AFPCorpus')
cleaner.retrieve.getDailyFileList(date, store='/home/docs/AFPCorpus', mergeWeekendsWithMonday=False)
cleaner.retrieve.getEmpiricalTable(tickerList, fromDate, toDate, csvFile='/home/docs/Dropbox/AFPdb/Empirical/adjustedClose.csv')
Returns a table in structure of structure format (Ticker By Date)

Parameters
- **tickerList** – A list of the tickers to be added into the table
- **fromDate** (datetime.date) – Time from which to begin the table
- **toDate** (datetime.date) – Time from which to end the table
- **empiricalStore** – The location of the Empirical file store
- **filename** – The name of the file within the Empirical file store

cleaner.retrieve.getFilteredFileList(includes=None, excludes=None, store='/home/docs/AFPCorpus')
Returns a list of files, based on includes and excludes filters

Parameters
- **includes** (dict) – Dictionary of values to be included by schema category
- **excludes** (dict) – Dictionary of values to be excluded by schema category

Example Usage: (Assume you want all February articles in 2011 that are not in the NYT)

```python
>>> import cleaner.retrieve as retrieve
>>> retrieve.getFilteredFileList( includes = { 'month' : [ 'February' ], 'year' : [ '2011' ] },
  store='/home/docs/AFPCorpus')
~/AFPCorpus/LexisNexis/2011/February\1\New York Times\40_UNDER_FORTY_Financial_advi.txt
~/AFPCorpus/LexisNexis/2011/February\1\New York Times\Walker_in_three_way_battle_for.txt
~/AFPCorpus/LexisNexis/2011/February\2\New York Times\Clinton_to_Grace_this_day_Awar.txt
...```

## 2.5 cleaner.schema module

Stores schema configurations, both for unclean and clean schemas

class cleaner.schema.LexisNexisSchema
  Bases: object
  API to normalize IO from uncleaned data to cleaned data

class LexisNexisArticleFiler
  Bases: cleaner.filers.ArticleFilerBase
  API to store a LexisNexis Article according to afp.settings

dateRegex = <_sre.SRE_Pattern object at 0x7fde8b654550>
paperDateTitleRegex = <_sre.SRE_Pattern object at 0x2dc1050>
removeFromArticleRegex = <_sre.SRE_Pattern object at 0x7fde8b63b140>
removeFromTitleRegex = <_sre.SRE_Pattern object at 0x7fde8b5361c0>
schemaName = ‘LexisNexis’
sectionDelimiter = ‘\n\n’

LexisNexisSchema.getArticleDelimiter()
LexisNexisSchema.getArticleFiler()
cleaner.schema.getFilePath(source, paper, month, day, year)

Configures cleaned file system schema

cleaner.schema.getSchema(sourceDirectory)

Given a sourceDirectory, returns the registered schema.

MUST Register schema here!

Example Usage:

>>> getSchema( ‘LexisNexis’ )
<_main__.LexisNexisSchema object at 0x022816F0>

2.6 cleaner.settings module

Constants and settings for cleaner. Compiled regexes reside here.

2.7 cleaner.store module

High level API that stores uncleaned data into cleaned store

cleaner.store.adjustedClose(tickerList, fromDate, toDate, empiricalStore='\n/home/docs/Dropbox/AFPdb/Empirical', filename='adjustedClose.csv')

Stores a .csv file of the adjusted closes of the ticker list by ordered date in empiricalStore

Parameters

• tickerList – List of tickers to be stored in csv from left to right
• fromDate (datetime.date) – Start date to get historical closes from
• toDate (datetime.date) – End date to get historical closes from
• empiricalStore – The folder to store the .csv file
• filename – The file name of the .csv file

For example:

>>> store.adjustedClose([‘GOOG’, ‘AAPL’], datetime.date(2012, 1, 10), datetime.date(2012,

cleaner.store.cleanSources(uncleanStore='\n/home/docs/Dropbox/AFPdb/Unclean', numWorkers=2)

Cleans all files in unclean directory, using numWorkers processors.

Parameters numWorkers – Number of processors to allocate. Defaults to multiprocessing.cpu_count()

Call this function directly to clean data. For example:
>>> cleanSources( settings.UNCLEAN_STORE )
Cleaning data in <C:\AFPunclean> with <8> workers.

2.8 Module contents
CHAPTER 3

Indices and tables

• genindex
• modindex
• search
Python Module Index

a
afp, 5
afp.keywords, 3
afp.linalg, 4
afp.normalize, 4
afp.settings, 5

cleaner, 11
cleaner.filers, 7
cleaner.helpers, 8
cleaner.retrieve, 8
cleaner.schema, 9
cleaner.settings, 10
cleaner.store, 10

f
filers, 7

h
helpers, 8

k
keywords, 3

l
linalg, 4

n
normalize, 4

r
retrieve, 8

s
schema, 9
settings, 10
store, 10