dhelp Documentation

Release 0.0.1

David J. Thomas

May 24, 2018

Contents

CHAPTER 1

README

1.1 dhelp

DH Python tools for scraping web pages, pre-processing data, and performing nlp analysis quickly

David J. Thomas, thePortus.com Instructor of Ancient History and Digital Humanities, Department of History, University of South Florida

For my students.

Students often see great potential for Python in Historical Analysis. But, before they see real payoff they often face too many hurdles to overcome in the space of a single semester. dhelp is a tool to allow students to quickly get to performing quick file operations, data manipulations, and even text analysis.

Check out the full project documentation dhelp.readthedocs.io

Requires Python 3.x

1.2 Table of Contents

1.3 Installation

Install with pip (recommended)

```
pip install dhelp
```

or. . .

Use easy_install

```
# clone the repo and move into the directory
git clone https://github.com/thePortus/dhelp.git && cd dhelp
# run easy_install
easy_install setup.py
```

1.4 Language Setup

The first time you use a language-specific text object, you need to run its .setup() method

```
from dhelp import EnglishText

EnglishText.setup()
```

1.5 Files Module

Use the file module to load a file as a string in a single line of code. Or, alter every text file in a folder by passing a simple function. Convert a column of text data in a .csv file into a series of .txt files, again in a single line of code.

1.5.1 TextFile

Main Examples

```
from dhelp import TextFile

# quickest method to modify a file, start by making a TextFile object...
txt_file = TextFile('some/file.txt')
# then use with/as syntax to give you the file contents in strings form
with txt_file as txt_data:
    # txt_data is contents, whatever you put in txt_file.save_data is saved
    txt_file.save_data = txt_data.replace('\n', '')

# Other methods...

# load file data as a string and print to screen
text_file = TextFile('some/file.txt')
text_file.load()
'Lorem ipsum dolor sit amet...'

# save string data to a file, returns self if successful
text_file = TextFile('some/file.txt')
text_file.save('Lorem ipsum dolor sit amet...')
'some/file.txt'
```

Other Functions

```
# use a dict to specify options, including overwriting existing files
options = {
    'encoding': 'utf-8' # set character encoding for loading/saving
    'silent': False # set True to supress messages during loading/saving
    'overwrite': False # set True to save over any previous file
}
text_file = TextFile('some/file.txt').save('Lorem ipsum dolor sit amet...',␣
↪options=options)

# check if a file exists at a location
TextFile('some/file.txt').exists
True

# move, copy, or delete files
TextFile('some/file.txt').move(destination='some/other-file.txt')
TextFile('some/other-file.txt').copy(destination='some/file.txt')
TextFile('some/other-file.txt').remove()
```

1.5.2 TextFolder

Main Examples

```
from dhelp import TextFolder

# quickest way to modify a folder, start by making a TextFolder object
text_folder = TextFolder('some/path')
# use with/as syntax to get a list of TextFile objects, then loop through
with text_folder as txt_files:
    for txt_file in txt_files:
```

(continues on next page)

(continued from previous page)

```
        # use with/as syntax on file to get contents
        with txt_file as txt_data:
            # whatever you store in .save_data will be saved to file
            txt_file.save_data = txt_data.replace('\n', '')
```

Other Methods

```
# returns a list of TextFile objects, each connected to a file in the folder
folder_files = TextFolder('some/folder').text_files
# You can loop through and load, edit, save, et.c. the TextFiles as normal
for folder_file in folder_files:
    # load the file as a string
    file_data = folder_file.load()
    # replace all endline characters with nothing
    file_data = file_data.replace('\n', '')
    # save altered data back to the file, specifying overwrite option
    folder_file.save(file_data, {'overwrite': True})

# .modify - a quick method to load/edit/and save all at once

# first, make a function showing how you want to alter the data of each file
# it gets 1 arg, the file str data, and returns 1 arg, the altered str data
def modify_file(file_data):
    # replace all endline characters with nothing and then return
    return file_data.replace('\n', '')

# send an output location and modify_file function as args and you are done!
TextFolder('some/folder').modify('some/other-folder', modify_file)
```

Other Functions

```
# use a dict to specify options, including overwriting existing files
options = {
    'encoding': 'utf-8' # set character encoding for loading/saving
    'silent': False # set True to supress messages during loading/saving
    'overwrite': False # set True to save over any previous file
}

TextFolder('some/folder').modify('some/other-folder', modify_file, options=options)

# check if a folder exists at a location
TextFile('some/folder').exists
True

# move, copy, or delete folders
TextFile('some/folder').move(destination='some/other-folder')
TextFile('some/other-folder').copy(destination='some/folder')
TextFile('some/other-folder').remove()
```

1.5.3 CSVFile

Main Examples

```
# load CSV data as a list of dictionaries
csv_file = CSVFile('some/file.csv')
```

(continues on next page)

(continued from previous page)

```
csv_file.load()
[{'id': '1', 'text': 'Lorem ipsum', 'notes': ''}, {'id': '2', 'text': 'dolor sit',
↪'notes': ''}, {'id': '3', 'text': 'amet.', 'notes': ''}]

# read column fieldnames of an existing csv
csv_file = CSVFile('some/file.csv')
csv_file.fieldnames
['id', 'text', 'notes']

# save a list of dictionary records to a csvfile
fake_data = [
    {'id', '1', 'text': 'Lorem ipsum'},
    {'id', '2', 'text': 'dolor sit'},
    {'id', '3', 'text': 'amet'}
]
fake_data_fieldnames = ['id', 'text']
csv_file = CSVFile('some/file.csv')
csv_file.save(fake_data, fake_data_fieldnames)
'some/file.csv'

# . modify - like TextFolder, load/edit/save every record

# like TextFolder, define a function, this time editing each record's data
def modify_csv_row(row_data):
    # the data will come in the form of a dictionary
    # here we just transform all text entries to lower case
    row_data['text'] = row_data['text'].lower()
    return row_data

# now call .modify, sending an output location and the function as args
CSVFile('some/file.csv').modify('some/other-file.csv', modify_csv_row)
```

Other Functions

```
# check if a csv exists at a location
CSVFile('some/file.csv').exists
True

# move, copy, or delete csvs
CSVFile('some/file.csv').move(destination='some/other-file.csv')
CSVFile('some/other-file.csv').copy(destination='some/file.csv')
CSVFile('some/other-file.csv').remove()
```

1.6 Web Module

1.6.1 WebPage

With the web module, you can download a webpage and parse it into a BeautifulSoup object with one command.

Examples

```
from dhelp import WebPage

# make beautifulsoup object
```

(continues on next page)

(continued from previous page)

```
page_soup = WebPage('https://stackoverflow.com').soup()
# narrow down to the element desired by chaining .find()
header_logo_text = page_soup.find('header')
    .find('div', class_='-main')
    .find('span', class_='-img')
 # print the text contained in the span tag
print(header_logo_text.get_text())
'Stack Overflow'
```

1.7 Text Module

The text module attaches mutiple methods for text cleaning/analysis that can be easily accessed. Use one of the Text classes to get a string-like object that comes with many convenient cleaning/nlp methods attached. You can chain any of the string transformation methods to perform many text operations at once.

1.7.1 EnglishText

Setup: Download the English Corpora

Before you use this object for any of the methods below you need to download trainer corpora.

```
from dhelp import EnglishText
EnglishText.setup()
```

Examples

```
# .rm_lines() - remove endline characters
text = EnglishText('The qui\\nck brown fox jumped over the lazy dog')
text.rm_lines()
'The quick brown fox jumped over the lazy dog'

# .rm_nonchars() - remove non-letters
text = EnglishText('Th3e quick brown fox jumped over the lazy dog')
text.rm_nonchars()
'The quick brown fox jumped over the lazy dog'

# .rm_edits() - remove text between editorial marks
text = EnglishText('The [quick] brown fox jumped over the lazy dog')
text.rm_edits()
'The brown fox jumped over the lazy dog'

# .rm_spaces() - collapses redundant whitespaces
text = EnglishText('Th3e qui\\nck b     rown fox jumped over the lazy dog')
text.rm_spaces()
'The quick brown fox jumped over the lazy dog'

# .re_search() - checks for a given pattern
text = EnglishText('The quick brown fox jumped over the lazy dog')
text.re_search('fox')
True
text.re_search('lemur')
False
```

(continues on next page)

(continued from previous page)

```
# .rm_stopwords() - removes a list of words from text
text = EnglishText('The quick brown fox jumped over the lazy dog')
text.rm_stopwords(['quick', 'brown','lazy'])
'The fox jumped over the dog'

# chain methods to perform them in one command
text = EnglishText('Th3e qui\\nck b     rown fox jumped over the lazy dog')
text.rm_lines().rm_nonchars().rm_spaces()
'The quick brown fox jumped over the lazy dog'

# lemmatize a text to make word counts/analysis
text = EnglishText('The quick brown fox jumped over the lazy dog.')
text.lemmatize()
'The quick brown fox jump over the lazy dog .'

# get 'tokens' (list of words)
text = EnglishText('The quick brown fox jumped over the lazy dog.')
EnglishText.tokenize()
['The', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog']

# get word tallys
text = EnglishText('The quick brown fox jumped over the lazy dog.')
EnglishText.word_count()
{'The': 1, 'quick': 1, 'brown': 1, 'fox': 1, 'jumped': 1, 'over': 1, 'the': 1, 'lazy
↪': 1, 'dog': 1}

# tag words with parts of speech
text = EnglishText('They hated to think of sample sentences.')
text.tag()
[('They', 'PRP'), ('hated', 'VBD'), ('to', 'TO'), ('think', 'VB'), ('of', 'IN'), (
↪'sample', 'JJ'), ('sentences', 'NNS'), ('.', '.')]

# generate ngrams...
text = EnglishText('They hated to think of sample sentences.')
text.ngrams()
[('They', 'hated', 'to'), ('hated', 'to', 'think'), ('to', 'think', 'of'), ('think',
↪'of', 'sample'), ('of', 'sample', 'sentences'), ('sample', 'sentences', '.')]

# ... or skipgrams
text = EnglishText('They hated to think of sample sentences.')
text.skipgrams()
[('They', 'hated', 'to'), ('They', 'hated', 'think'), ('They', 'to', 'think'), ('hated
↪', 'to', 'think'), ('hated', 'to', 'of'), ('hated', 'think', 'of'), ('to', 'think',
↪'of'), ('to', 'think', 'sample'), ('to', 'of', 'sample'), ('think', 'of', 'sample'),
↪ ('think', 'of', 'sentences'), ('think', 'sample', 'sentences'), ('of', 'sample',
↪'sentences'), ('of', 'sample', '.'), ('of', 'sentences', '.'), ('sample', 'sentences
↪', '.')]
```

1.7.2 LatinText

Setup: Download the Latin Corpora

Before you use this object for any of the methods below you need to download trainer corpora.

```
from dhelp import LatinText
LatinText('').setup()
```

Examples

```
# .rm_lines() - remove endline characters
text = LatinText('Gallia \\nest omnis divisa in partes tres')
text.rm_lines()
'Gallia est omnis divisa in partes tres'

# .rm_nonchars() - remove non-letters
text = LatinText('Ga3llia est omnis divisa in partes tres')
text.rm_nonchars()
'Gallia est omnis divisa in partes tres'

# .rm_edits() - remove text between editorial marks
text = LatinText('Gallia est [omnis] divisa in partes tres)
text.rm_edits()
'Gallia est omnis divisa in partes tres'

# .rm_spaces() - collapses redundant whitespaces
text = LatinText('Gallia    est omnis divisa       in partes        tres')
text.rm_spaces()
'Gallia est omnis divisa in partes tres'

# .re_search() - checks for a given pattern
text = LatinText('Gallia est omnis divisa in partes tres')
text.re_search('Gallia')
True
text.re_search('Graecia')
False

# .rm_stopwords() - removes a list of words from text
text = LatinText('Gallia est omnis divisa in partes tres')
text.rm_stopwords(['est', 'in'])
'Gallia omnis divisa partes tres'

# chain methods to perform them in one command
text = LatinText('Ga3llia    \\nest omnis divisa       in partes        tres')
text.rm_lines().rm_nonchars().rm_spaces()
'Gallia est omnis divisa in partes tres'

# tokenize words into list of strings
text = LatinText('Gallia est omnis divisa in partes tres')
text.tokenize()
['Gallia', 'est', 'omnis', 'divisa', 'in', 'partes', 'tres']

# lemmatize the text
text = LatinText('Gallia est omnis divisa in partes tres')
text.lemmatize()
'gallia edo1 omne divido in pars tres'

# generate ngrams...
text = LatinText('Gallia est omnis divisa in partes tres')
text.ngrams()
[('Gallia', 'est', 'omnis'), ('est', 'omnis', 'divisa'), ('omnis', 'divisa', 'in'), (
↪'divisa', 'in', 'partes'), ('in', 'partes', 'tres')]

# ... or skipgrams
text = LatinText('Gallia est omnis divisa in partes tres')
text.skipgrams()
```

(continues on next page)

(continued from previous page)

```
[('Gallia', 'est', 'omnis'), ('est', 'omnis', 'divisa'), ('omnis', 'divisa', 'in'), (
↪'divisa', 'in', 'partes'), ('in', 'partes', 'tres')]

# count all words
text = LatinText('Gallia est omnis divisa in partes tres tres tres')
text.word_count(word='tres')
3

# scan text for meter
text = LatinText('Arma virumque cano, Troiae qui primus ab oris')
text.scansion()
['-----------------x']

# get recognized entities as a list
text = LatinText('Gallia est omnis divisa in partes tres')
text.entities()
['Gallia']

# macronize vowels
text = LatinText('Arma virumque cano, Troiae qui primus ab oris')
text.macronize()
'arma virumque cano , trojae quī prīmus ab ōrīs'

# search for known entities
text = LatinText('Gallia est omnis divisa in partes tres')
text.entities()
['Gallia']

# compare for longest common substring
text = LatinText('Gallia est omnis divisa in partes tres')
text.compare_longest_common_substring('Galliae sunt omnis divisae in partes tres')
'in partes tres'

# compare minhash's
text = LatinText('Gallia est omnis divisa in partes tres')
text.compare_minhash('Galliae sunt omnis divisae in partes tres')
0.6444444444444445
```

1.7.3 AncientGreekText

Setup: Download the Greek Corpora

Before you use this object for any of the methods below you need to download trainer corpora.

```
from dhelp import AncientGreekText
AncientGreekText('').setup()
```

Examples

```
# .rm_lines() - remove endline characters
text = AncientGreekText('νθα \nποτ θηναοι ν ργρου μταλλα'')
text.rm_lines()
'νθα ποτ θηναοι ν ργρου μταλλα

# .rm_nonchars() - remove non-letters
```

(continues on next page)

(continued from previous page)

```
text = AncientGreekText('ν3θα ποτ θηναοι ν ργρου μταλλα'')
text.rm_nonchars()
'νθα ποτ θηναοι ν ργρου μταλλα'

# .rm_edits() - remove text between editorial marks
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.rm_edits()
'νθα ποτ θηναοι ν ργρου μταλλα'

# .rm_spaces() - collapses redundant whitespaces
text = AncientGreekText('νθα      ποτ      θηναοι ν ργρου μταλλα)
text.rm_spaces()
'νθα ποτ θηναοι ν ργρου μταλλα'

# .re_search() - checks for a given pattern
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.re_search('θηναοι')
True
text.re_search('σπαρτοι')
False

# .rm_stopwords() - removes a list of words from text
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.rm_stopwords(['ποτ', 'ργρου'])
'νθα θηναοι ν μταλλα'

# chain methods to perform them in one command
text = AncientGreekText('ν3θα      \nποτ      θηναοι ν ργρου μταλλα')
text.rm_lines().rm_nonchars().rm_spaces()
'νθα ποτ θηναοι ν ργρου μταλλα'

# normalize character encoding differences
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.normalize()
'νθα ποτ θηναοι ν ργρου μταλλα'

# perform text cleanup, designed for tlg texts
text = AncientGreekText('ν», ϵ δ τν τερον κ[α]τττ[ερον «ϵ λιον κα μει]νν στι')
text.tlgu_cleanup()
'ν ϵ δ τν τερον κατττερον ϵ λιον κα μεινν στι'

# tokenize words into list of strings
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.tokenize()
['νθα', 'ποτ', 'θηναοι', 'ν', 'ργρου', 'μταλλα']

# lemmatize the text
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.lemmatize()
'νθα ποτ θηναιο ϵμ ργυρω μταλλον'

# generate ngrams...
text.ngrams()
[('νθα', 'ποτ', 'θηναοι'), ('ποτ', 'θηναοι', 'ν'), ('θηναοι', 'ν', 'ργρου'), ('ν', 'ργρου
↪', 'μταλλα')]

# ... or skipgrams
```

(continues on next page)

(continued from previous page)

```
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.skipgrams()
[('νθα', 'ποτ', 'θηναοι'), ('νθα', 'ποτ', 'ν'), ('νθα', 'θηναοι', 'ν'), ('ποτ', 'θηναοι',
↪ 'ν'), ('ποτ', 'θηναοι', 'ργρου'), ('ποτ', 'ν', 'ργρου'), ('θηναοι', 'ν', 'ργρου'), (
↪'θηναοι', 'ν', 'μταλλα'), ('θηναοι', 'ργρου', 'μταλλα'), ('ν', 'ργρου', 'μταλλα')]

# perform part-of-speech tagging
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.tag()
[('νθα', 'D--------'), ('ποτ', 'G--------'), ('θηναοι', None), ('ν', 'V3SIIA---'), (
↪'ργρου', 'N-S---MG-'), ('μταλλα', None)]

# search for known entities
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.entities()
['θηναοι']

# compare for longest common substring
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.compare_longest_common_substring('νθα θηναοι ργρου μταλλα')
'ργρου μταλλα'

# compare minhash's
AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.compare_minhash('θηναοι ν μταλλα')
0.4594594594594595

# count all words
text = AncientGreekText('νθα ποτ θηναοι ν ργρου μταλλα')
text.word_count(word='θηναοι')
1
```

1.8 Using Objects Together

Lemmatizing an entire folder of Latin text files

```
from dhelp import TextFolder, LatinText

# define a function which defines how each file's data will be modified (in this case␣
↪removing extra spaces and lemmatizing)
def modify_function(file_data):
    file_data = LatinText(file_Data)
    return file_data.rm_space().lemmatize()

# create a TextFolder object tied to the location of the Latin txt files
text_folder = TextFolder('path/to/latin/files')
# call the modify method, sending the output path and the name of the function you␣
↪defined
text_folder.modify('path/to/latin/files-lemmatized', modify_function)

# That's it, you will now have a folder full of lemmatized Latin files
```

More Examples Coming Soon

CHAPTER 2

dhelp package

2.1 Submodules

2.2 dhelp.files module

2.3 dhelp.settings module

2.4 dhelp.text module

2.5 dhelp.web module

class dhelp.web.**WebPage**(*url*, *options={}*)

Bases: collections.UserString

Downloads and parses HTML into BeautifulSoup objects.

Provides methods to download/parse a specified webpage. Merges the request package with BeautifulSoup functions to enable users to request/soup a page in a single line.

Parameters

- **url** (str)
- **options** (dict, optional)

Examples

```
>>> from dhelp import WebPage
>>> web_page = WebPage('https://stackoverflow.com')
>>> print(web_page)
```

(continues on next page)

(continued from previous page)

```
'https://stackoverflow.com'
>>> # pass an dict to set options for delay, max_retries, or silent
>>> options = {
...     'delay': 4,
        'max_retries': 3,
        'silent': True
        'parser': 'html.parser'
... }
>>> web_page = WebPage('https://stackoverflow.com', options=options)
https://stackoverflow.com
```

fetch(*retry_counter=0*)

Returns http request from URL as a string.

Can be called to return HTML data, although not generally meant to be called directly by user. If user calls .fetch(), retry_counter should not be passed so that it will start at 0. This function is intended to be called by .soup() in order to feed its parser.

If the request was not successful, .fetch() calls itself recursively until it is either successful, or the maximum number of attempts has been reached. If the .max_retries property is set to 0, .fetch() will make inifinite requests.

Parameters **retry_counter** (`int`)

Returns `str` HTML from requested URL, in plain text format

Examples

```
>>> html_text = WebPage('https://stackoverflow.com/').fetch()
<!DOCTYPE html>\r\n<html>\r\n\r\n    <head>\r\n\r\n        <title>Stack␣
↪Overflow...
```

soup()

Returns a BeautifulSoup object loaded with HTML data from the URL

Invokes web request then returns a soup object loaded with page HTML. Uses html.parser with BeautifulSoup. Child classes may override this to use other parsers (e.g. lxml).

Returns `bs4.BeautifulSoup` BeautifulSoup object loaded with parsed data from web

Examples

```
>>> # fetch webpage and parse into BeautifulSoup object
>>> parsed_webpage = WebPage('https://stackoverflow.com/').soup()
>>> # grab the logo from the header with BeautifulSoup
>>> header_logo_text = parsed_webpage.find('header')
...     .find('div', class_='-main')
...     .find('span', class_='-img')
>>> # print the text contained in the span tag
>>> print(header_logo_text.get_text())
Stack Overflow
```

2.6 Module contents

dhelp

David J. Thomas, thePortus.com, Copyright, 2018

Students often see great potential in Python for historical analysis. But, before they see real payoff they often face too many hurdles to overcome in the space of a single semester. dhelp is a tool to allow students to quickly get to performing quick file operations, data manipulations, and even text analysis.

CHAPTER 3

CHANGELOG

3.1 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

3.1.1 [0.0.4]

- Changes
  - Web module reorganized into main package

3.1.2 [0.0.3]

- Changes
  - Provisional setup to auto-pip install cltk added
  - Badges added

3.1.3 [0.0.2]

- Changes
  - Unit testing expanded
  - readme documentation greatly epxanded
- Additions
  - code of conduct and contributing

3.1.4 [0.0.1]

- Changes...
  - Project renamed to dhelp
  - Each module passing tests(see below)
  - Project documentation added to README.md
- Additions...
  - Unit testing for files, text, and web modules added
  - Test coverage added with computer-readable (.coverage) and human-friendly (an html folder, htmlcov)
  - Continuous Integration support for Travis-CI and Coveralls
  - .editorconfig added to enforce project standards
  - To document changes, both past and future, added CHANGELOG.md and TODO.md
  - CLTK functionality added for Greek/Latin texts
  - Project documentation with sphinx added

3.1.5 [Unreleased]

- Additions...
  - files module
  - text module
  - web module

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index