pylexique Documentation

Release 1.5.0

SekouDiaoNlp

November 05, 2021

CONTENT

CHAPTER

ONE

PYLEXIQUE

Pylexique is a Python wrapper around Lexique383.

It allows the extraction of lexical information from more than 140 000 French words in an Object Oriented way.

- Free software: MIT license
- PyLexique Documentation: https://pylexique.readthedocs.io (en) – https://sekoudiaonlp.github.io/pylexique/fr_FR/ (fr)

Each lexical item is represented as a LexItem having the following LexEntryType:

```
class LexEntryType:
"""
Type information about all the lexical attributes in a LexItem object.

"""
ortho: str
phon: str
lemme: str
cgram: str
genre: str
nombre: str
freqlemfilms2: float
freqlemlivres: float
freqfilms2: float
freqlivres: float
infover: str
nbhomogr: int
nbhomoph: int
islem: bool
nblettres: int
nbphons: int
cvcv: str
p_cvcv: str
voisorth: int
voisphon: int
puorth: int
puphon: int
syll: str
nbsyll: int
cv_cv: str
orthrenv: str
phonrenv: str
orthosyll: str
cgramortho: str
deflem: float
defobs: int
old20: float
pld20: float
morphoder: str
nbmorph: int
```

The meanings of the attributes of this object are as follow:

- ortho: the word
- phon: the phonological forms of the word
- lemme: the lemmas of this word
- cgram: the grammatical categories of this word
- genre: the gender
- nombre: the number
- freqlemfilms: the frequency of the lemma according to the corpus of subtitles (per million occurrences)

- freqlemlivres: the frequency of the lemma according to the body of books (per million occurrences)
- freqfilms: the frequency of the word according to the corpus of subtitles (per million occurrences)
- freqlivres: the frequency of the word according to the body of books (per million occurrences)
- infover: modes, tenses, and possible people for verbs
- nbhomogr: number of homographs
- nbhomoph: number of homophones
- islem: indicates if it is a lemma or not
- nblettres: the number of letters
- nbphons: number of phonemes
- cvcv: the orthographic structure
- p-cvcv: the phonological structure
- voisorth: number of orthographic neighbors
- voisphon: number of phonological neighbors
- puorth: point of spelling uniqueness
- puphon: point of phonological uniqueness
- syll: syllable phonological form
- nbsyll: number of syllables
- cv-cv: syllable phonological structure
- orthrenv: reverse orthographic form
- phonrenv: reversed phonological form
- orthosyll: syllable orthographic form
- cgramortho: the different grammatical category for a given orthographic representation
- deflem: the percentage of people who said they knew the lemma of the word
- defobs: the size of the sample from which 'deflem' is derived
- old20: orthographic Levenshtein Distance
- pld20: phonological Levenshtein Distance
- morphoder: inflectional morphology
- nbmorph: the number of morphemes directly computed from 'morphoder'

You can find all the relevant information in the official documentation of Lexique383 (French).

1.1 Features

- **Extract all lexical information from a French word such as:**
  - orthographic and phonemics representations
  - associated lemmas
  - syllabation
  - grammatical category
  - gender and number
  - frequencies in a corpus of books and in a body of film subtitles, etc…
- Extract all the lexical forms of a French word.
- Easy to use Api.
- Easily integrate pylexique in your own projects as an imported library.
- Can be used as a command line tool.

1.2 Credits

Main developer SekouDiaoNlp.

Lexical corpus: Lexique383

1.2.1 About Lexique383

1.2.2 Lexique3

Lexique 3.83 is a French lexical database that provides for ~ 140,000 words of French: orthographic and phonemics representations, associated lemmas, syllabation, grammatical category, gender and number, frequencies in a corpus of books and in a body of film subtitles, etc…

Table: Lexique383.zip

Web site: http://www.lexique.org

Online: http://www.lexique.org/shiny/lexique

1.3 Publications

- New, Boris, Christophe Pallier, Marc Brysbaert, and Ludovic Ferrand. 2004. "Lexique 2: A New French Lexical Database." *Behavior Research Methods, Instruments, & Computers* 36 (3): 516–524. DOI. pdf
- New, Boris, Christophe Pallier, Ludovic Ferrand, and Rafael Matos. 2001. "Une Base de Données Lexicales Du Français Contemporain Sur Internet: LEXIQUE" *L'Année Psychologique* 101 (3): 447–462. DOI. pdf
- Boris New, Marc Brysbaert, Jean Veronis, and Christophe Pallier. 2007. "The Use of Film Subtitles to Estimate Word Frequencies." Applied Psycholinguistics 28 (4): 661–77. DOI. (pdf)

1.4 Contributors

- Boris New & Christophe Pallier
- Ronald Peereman
- Sophie Dufour
- Christian Lachaud
- and many others… (contact us to be listed)

1.5 License

BibTex Entry to cite publications about Lexique383:

```
@article{npbf04,
author = {New, B. and Pallier, C. and Brysbaert, M. and Ferrand, L.},
journal = {ehavior Research Methods, Instruments, & Computers},
number = {3},
pages = {516-524},
title = {Lexique 2 : A New French Lexical Database},
volume = {36},
year = {2004},
eprint = {http://www.lexique.org/?page_id=294},
}
```

```
@article{npfm01,
author = {New, B. and Pallier, C. and Ferrand, L. and Matos, R.},
journal = {L'Ann{\'e}e Pschologique},
number = {447-462},
pages = {1396-2},
title = {Une base de donn{\'e}es lexicales du fran\c{c}ais contemporain sur internet:␣
↪LEXIQUE},
volume = {101},
year = {2001},
}
```

```
@article{new_brysbaert_veronis_pallier_2007,
author={NEW, BORIS and BRYSBAERT, MARC and VERONIS, JEAN and PALLIER, CHRISTOPHE},
title={The use of film subtitles to estimate word frequencies},
volume={28}, DOI={10.1017/S014271640707035X},
number={4}, journal={Applied Psycholinguistics},
publisher={Cambridge University Press},
year={2007},
pages={661-677}}
```

1.6 BibTeX

If you want to cite pylexique in an academic publication use this citation format:

```
@article{pylexique,
  title={pylexique},
  author={Sekou Diao},
  journal={GitHub. Note: https://github.com/SekouDiaoNlp/pylexique Cited by},
  year={2021}
}
```

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install pylexique, run this command in your terminal:

```
$ pip install pylexique
```

This is the preferred method to install pylexique, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

You can also install pylexique by using Anaconda or Miniconda instead of pip .

To install Anaconda or Miniconda, please follow the installation instructions on their respective websites. After having installed Anaconda or Miniconda, run these commands in your terminal:

```
$ conda config --add channels conda-forge
$ conda config --set channel_priority strict
$ conda install pylexique
```

If you already have Anaconda or Miniconda available on your system, just type this in your terminal:

```
$ conda install -c conda-forge pylexique
```

Warning: If you intend to install pylexique on a Apple Macbook with an Apple M1 processor, it is advised that you install pylexique by using the conda installation method as all dependencies will be pre-compiled.

2.2 From sources

The sources for pylexique can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/SekouDiaoNlp/pylexique
```

Or download the tarball:

```
$ curl  -OL https://github.com/SekouDiaoNlp/pylexique/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

Note: The language of the lexical entries is French. The Lexical Corpus is based on *Lexique383*. Also note that pylexique only works on Python 3.X.

3.1 To use pylexique from the command line:

```
$ pylexique manger
```

The output will be a json representation of the LexItem objects:

```
{
    "manger": [
        [
            {
                "ortho": "manger",
                "phon": "m@Ze",
                "lemme": "manger",
                "cgram": "NOM",
                "genre": "m",
                "nombre": "s",
                "freqlemfilms2": 5.62,
                "freqlemlivres": 4.05,
                "freqfilms2": 5.62,
                "freqlivres": 4.05,
                "infover": "",
                "nbhomogr": 2,
                "nbhomoph": 7,
                "islem": True,
                "nblettres": 6,
                "nbphons": 4,
                "cvcv": "CVCCVC",
                "p_cvcv": "CVCV",
                "voisorth": 9,
                "voisphon": 12,
                "puorth": 6,
                "puphon": 4,
                "syll": "m@-Ze",
                "nbsyll": 2,
```

(continues on next page)

(continued from previous page)


```
                "cv_cv": "CV-CV",
                "orthrenv": "regnam",
                "phonrenv": "eZ@m",
                "orthosyll": "man-ger",
                "cgramortho": "NOM,VER",
                "deflem": "100",
                "defobs": "33",
                "old20": 1.35,
                "pld20": 1.25,
                "morphoder": "manger",
                "nbmorph": "1"
            },
            {
                "ortho": "manger",
                "phon": "m@Ze",
                "lemme": "manger",
                "cgram": "VER",
                "genre": "",
                "nombre": "",
                "freqlemfilms2": 467.82,
                "freqlemlivres": 280.61,
                "freqfilms2": 207.63,
                "freqlivres": 134.26,
                "infover": "\"inf;;\"",
                "nbhomogr": 2,
                "nbhomoph": 7,
                "islem": True,
                "nblettres": 6,
                "nbphons": 4,
                "cvcv": "CVCCVC",
                "p_cvcv": "CVCV",
                "voisorth": 9,
                "voisphon": 12,
                "puorth": 6,
                "puphon": 4,
                "syll": "m@-Ze",
                "nbsyll": 2,
                "cv_cv": "CV-CV",
                "orthrenv": "regnam",
                "phonrenv": "eZ@m",
                "orthosyll": "man-ger",
                "cgramortho": "NOM,VER",
                "deflem": "100",
                "defobs": "20",
                "old20": 1.35,
                "pld20": 1.25,
                "morphoder": "manger",
                "nbmorph": "1"
            }
        ]
    ]
}
```

```
$ pylexique boire
```


```
{
    "boire": [
        [
            {
                "ortho": "boire",
                "phon": "bwaR",
                "lemme": "boire",
                "cgram": "NOM",
                "genre": "m",
                "nombre": "s",
                "freqlemfilms2": 2.67,
                "freqlemlivres": 2.03,
                "freqfilms2": 2.67,
                "freqlivres": 2.03,
                "infover": "",
                "nbhomogr": 2,
                "nbhomoph": 2,
                "islem": True,
                "nblettres": 5,
                "nbphons": 4,
                "cvcv": "CVVCV",
                "p_cvcv": "CYVC",
                "voisorth": 9,
                "voisphon": 14,
                "puorth": 4,
                "puphon": 4,
                "syll": "bwaR",
                "nbsyll": 1,
                "cv_cv": "CYVC",
                "orthrenv": "eriob",
                "phonrenv": "Rawb",
                "orthosyll": "boi-re",
                "cgramortho": "NOM,VER",
                "deflem": "96",
                "defobs": "30",
                "old20": 1.4,
                "pld20": "1",
                "morphoder": "boire",
                "nbmorph": "1"
            },
            {
                "ortho": "boire",
                "phon": "bwaR",
                "lemme": "boire",
                "cgram": "VER",
                "genre": "",
                "nombre": "",
                "freqlemfilms2": 339.05,
                "freqlemlivres": 274.32,
                "freqfilms2": 142.15,
```


(continues on next page)

(continued from previous page)


```
                "freqlivres": 100.27,
                "infover": "\"inf;;\"",
                "nbhomogr": 2,
                "nbhomoph": 2,
                "islem": True,
                "nblettres": 5,
                "nbphons": 4,
                "cvcv": "CVVCV",
                "p_cvcv": "CYVC",
                "voisorth": 9,
                "voisphon": 14,
                "puorth": 4,
                "puphon": 4,
                "syll": "bwaR",
                "nbsyll": 1,
                "cv_cv": "CYVC",
                "orthrenv": "eriob",
                "phonrenv": "Rawb",
                "orthosyll": "boi-re",
                "cgramortho": "NOM,VER",
                "deflem": "100",
                "defobs": "30",
                "old20": 1.4,
                "pld20": "1",
                "morphoder": "boire",
                "nbmorph": "1"
            }
        ]
    ]
}
```


You can also provide multiple words and/or specify an output file to save the lexical information in a json file.

```
$ pylexique il mange une baguette

$ pylexique il boit du vin rouge -o path/to/the/output/json/file.json
```

The output will be similar as previously, with a json entry for each word in the sequence.

You can also retrieve all the lexical forms of the provided word/words by using the option '-a' or '–all_forms'

```
$ pylexique il mange une baguette -a

$ pylexique il boit du vin rouge -o path/to/the/output/json/file.json --all_forms
```

3.2 To use pylexique as a library in your own projects:

```
from pylexique import Lexique383
from pprint import pprint

#  Create new Lexique383 instance with a pre-built Lexique383.
LEXIQUE = Lexique383()

# Creates a new Lexique383 instance while supplying your own Lexique38X lexicon. The␣
↪first time it will it will be
# slow to parse the file and create a persistent data-store. Next runs should be much␣
↪faster.
RESOURCE_PATH = 'path/to/Lexique38x'
# parser_type must be either omitted if RESOURCE_PATH is a csv file and you want to use␣
↪the default csv parser.
# if parser_type is provided it should be either 'xlsb', 'pandas_csv', 'csv', 'std_csv'. 'std_
↪csv' is used by default.
LEXIQUE2 = Lexique383(RESOURCE_PATH, parser_type='std_csv')
```

There are 2 ways to access the lexical information of a word: Either use the utility method Lexique383.get_lex(item) Or you can directly access the lexicon directory through LEXIQUE.lexique[item] .

Notice that item can be either a string or a sequence of strings when using Lexique383.get_lex(item) .

```
#  Retrieves the lexical information of 'abaissait' and 'a'.
var_1 = LEXIQUE.lexique['abaissait']
var_1_bis = LEXIQUE.get_lex('abaissait')

# Check both objects are the same
var_1_equality = var_1 == var_1_bis['abaissait']
print(var_1_equality)
```

Because in French the world ‘a’ is a very polysemic word, it has several entries in Lexique 383. For this reason the LEXIQUE Dict has the value of the *ortho* property of its LexicalEntry. In th case of ‘abaissait’ there is only one LexicalItem corresponding to this dict key. But in the case of ‘a’ there are several LexItem objects corresponding to this key and then the LexItem objects are stored in a list corresponding to th value of the key.

```
var_2 = LEXIQUE.lexique['a']
var_2_bis = LEXIQUE.get_lex('a')

# Check both objects are the same
var_2_equality = var_2 == var_2_bis['a']
print(var_2_equality)

# Retrieving the lexical information of several words by passing a Sequence of␣
↪strings

var_multiple = LEXIQUE.get_lex(('il', 'mange', 'une', 'baguette'))
pprint(var_multiple)
```

You can get all the anagrams of a given word by using the get_anagrams() method.

```
var_3 = lexicon.get_anagrams('abaisser')
```

(continues on next page)

(continued from previous page)

```
pprint(var_3)
```

You can get all the forms of a given word by calling the method Lexique383.get_all_forms(word):

```
all_avoir_forms = LEXIQUE.get_all_forms('avez')
print(len(all_avoir_forms))

print('\n')

all_vouloir_forms = LEXIQUE.get_all_forms('voulu')
print(len(all_vouloir_forms))
```

You can use the method LexItem.to_dict() to produce a dictionary with key/value pairs corresponding to the LexItem

```
print('\n\n')
if isinstance(var_1, list):
    for elmt in var_1:
        pprint(elmt.to_dict())
        print('\n\n')
else:
    pprint(var_1.to_dict())
    print('\n\n')

print('\n\n')
if isinstance(var_2, list):
    for elmt in var_2:
        pprint(elmt.to_dict())
        print('\n\n')
else:
    pprint(var_2.to_dict())
    print('\n\n')

# Get all verbs in the DataSet. Because some words have the same orthography,␣
↪some keys of the dictionary
# don't have a unique LexItem object as their value, but a list of those.
verbs = []
for x in LEXIQUE.lexique.values():
    if isinstance(x, list):
        for y in x:
            if not isinstance(y, list) and y.cgram == 'VER':
                verbs.append(y)
    elif x.cgram == 'VER':
        verbs.append(x)
    else:
        continue

print('Printing the first 5 verbs found in the preceding search:')
pprint(verbs[0:5])

# Print the first 5 verbs with full lexical information.
for verb in verbs[0:5]:
    pprint(verb.to_dict())
```

Documentation for Lexique383: http://www.lexique.org

CHAPTER

FOUR

PACKAGE API DOCUMENTATION FOR PYLEXIQUE

4.1 API Reference for the classes in pylexique.pylexique.py

Main module of pylexique.

class pylexique.pylexique.**Lexique383**(*lexique_path: Optional[str] = None*, *parser_type: str = 'csv'*)

Bases: object

This is the class handling the lexique database. It provides methods for interacting with the Lexique DB and retrieve lexical items. All the lexical items are then stored in an Ordered Dict.

Parameters

- **lexique_path** – string. Path to the lexique file.
- **parser_type** – string. 'pandas_csv' and 'csv' are valid values. 'csv' is the default value.

Variables

- **lexique** – Dictionary containing all the LexicalItem objects indexed by orthography.
- **lemmes** – Dictionary containing all the LexicalItem objects indexed by lemma.
- **anagrams** – Dictionary containing all the LexicalItem objects indexed by anagram form.

static _parse_csv(*lexique_path: str*) → Generator[list, Any, None]

Parameters **lexique_path** – string. Path to the lexique file.

Returns generator of rows: Content of the Lexique38x database.

_parse_lexique(*lexique_path: str*, *parser_type: str*) → None

Parses the given lexique file and creates 2 hash tables to store the data.

Parameters

- **lexique_path** – string. Path to the lexique file.
- **parser_type** – string. Can be either 'csv', 'pandas_csv'.

Returns

_create_db(*lexicon: Generator[list, Any, None]*) → None

Creates 2 hash tables populated with the entries in lexique if it does not exist yet.

One hash table holds the LexItems, the other holds the same data but grouped by lemmma to give access to all lexical forms of a word.

Parameters **lexicon** – Iterable. Iterable containing the lexique383 entries.

Returns

_convert_entries(*row_fields: Union[List[str], List[Union[str, float, int, bool]]]*) → Tuple[str, str, str, str, str, str, float, float, float, float, str, int, int, bool, int, int, str, str, int, int, int, int, str, int, str, str, str, str, str, float, int, float, float, str, int]

Convert entries from *strings* to *int*, *bool* or *float* and generates
a new list with typed entries.

Parameters **row_fields** – List of column entries representing a row.

Returns ConvertedRow: List of typed column entries representing a typed row.

get_lex(*words: Union[Tuple[str, ...], str]*) → Dict[str, Union[*pylexique.pylexique.LexItem*, List[*pylexique.pylexique.LexItem*]]]
Recovers the lexical entries for the words in the sequence

Parameters **words** – A string or a tuple of multiple strings for getting the LexItems for multiple words.

Returns Dictionary of LexItems.

Raises TypeError.

get_all_forms(*word: str*) → List[*pylexique.pylexique.LexItem*]
Gets all lexical forms of a given word.

Parameters **word** – String.

Returns List of LexItem objects sharing the same root lemma.

Raises ValueError.

Raises TypeError.

get_anagrams(*word: str*) → List[*pylexique.pylexique.LexItem*]
Gets all lexical forms of a given word.

Parameters **word** – String.

Returns List of LexItem objects which are anagrams of the given word.

Raises ValueError.

Raises TypeError.

static _save_errors(*errors: Union[List[Tuple[List[Union[str, float, int, bool]], List[str]]], List[DefaultDict[str, List[Dict[str, str]]]]]*, *errors_path: str*) → None
Saves the mismatched key/values in Lexique383 based on type coercion.

Parameters

- **errors** – List of errors encountered while parsing Lexique38x
- **errors_path** – Path to save the errors.

Returns

class pylexique.pylexique.**LexItem**(*ortho: str*, *phon: str*, *lemme: str*, *cgram: str*, *genre: str*, *nombre: str*, *freqlemfilms2: float*, *freqlemlivres: float*, *freqfilms2: float*, *freqlivres: float*, *infover: str*, *nbhomogr: int*, *nbhomoph: int*, *islem: bool*, *nblettres: int*, *nbphons: int*, *cvcv: str*, *p_cvcv: str*, *voisorth: int*, *voisphon: int*, *puorth: int*, *puphon: int*, *syll: str*, *nbsyll: int*, *cv_cv: str*, *orthrenv: str*, *phonrenv: str*, *orthosyll: str*, *cgramortho: str*, *deflem: float*, *defobs: int*, *old20: float*, *pld20: float*, *morphoder: str*, *nbmorph: int*)

Bases: *pylexique.pylexique.LexEntryTypes*

This class defines the lexical items in Lexique383.
It uses slots for memory efficiency.

to_dict() → Dict[str, Union[str, float, int, bool]]

Converts the LexItem to a dict containing its attributes and their values

Returns OrderedDict. Dictionary with key/values correspondence wit LexItem objects.

Raises AttributeError.

class pylexique.pylexique.**LexEntryTypes**(*ortho: str*, *phon: str*, *lemme: str*, *cgram: str*, *genre: str*, *nombre: str*, *freqlemfilms2: float*, *freqlemlivres: float*, *freqfilms2: float*, *freqlivres: float*, *infover: str*, *nbhomogr: int*, *nbhomoph: int*, *islem: bool*, *nblettres: int*, *nbphons: int*, *cvcv: str*, *p_cvcv: str*, *voisorth: int*, *voisphon: int*, *puorth: int*, *puphon: int*, *syll: str*, *nbsyll: int*, *cv_cv: str*, *orthrenv: str*, *phonrenv: str*, *orthosyll: str*, *cgramortho: str*, *deflem: float*, *defobs: int*, *old20: float*, *pld20: float*, *morphoder: str*, *nbmorph: int*)

Bases: `object`

Type information about all the lexical attributes in a LexItem object.

4.2 API Reference for the classes in pylexique.utils.py

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/SekouDiaoNlp/pylexique/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

pylexique could always use more documentation, whether as part of the official pylexique docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/SekouDiaoNlp/pylexique/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *pylexique* for local development.

1. Fork the *pylexique* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pylexique.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pylexique
$ cd pylexique/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 pylexique tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7, 3.8 3.9 and for PyPy. Check https://github.com/SekouDiaoNlp/pylexique/actions and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_pylexique
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

GitHub Actions will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- SekouDiaoNlp <diao.sekou.nlp@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 1.5.0 (2021-10-28)

- Added new method 'Lexique383.get_anagrams()' to get the list of all the anagrams of a given word.
- Removed support for xlsb file format..
- Migrated the project build configuration to pyproject.toml.
- Migrated build backend to poetry.
- Updated the Documentation and the docstrings.
- Updated dependencies.

7.2 1.4.0 (2021-10-23)

- The library is now completely type annotated and type-checked.
- The method get_all_forms() now checks if the word has multiple lemmas in case of homonymy, eg: 'souris'.
- Fixed parsing bug when using 'std-csv' parser.
- Fixed formatting issue when saving the output of the CLI to a file.
- Updated the Documentation and the docstrings.
- Updated dependencies.

7.3 1.3.5 (2021-05-18)

- Uses str.lower() to normalize inputs.
- Updated the Documentation and the docstrings.

7.4 1.3.4 (2021-05-16)

- Fixed bug where Lexique383 was not shipped with the distribution.
- Made csv parsing far faster and more robust.
- Can now use different parsers : 'pandas_csv' is the pandas csv parser, 'std_csv' is the standard library csv parser, 'csv' is a custom csv parser and 'xlsb' is pandas xlsb parser using pyxlsb engine.
- Updated dependencies.

7.5 1.3.3 (2021-05-16)

- Made csv parsing faster and more robust.
- Can now use different parsers : 'pandas_csv' is the pandas csv parser, 'std_csv' is the standard library csv parser, 'csv' is a custom csv parser and 'xlsb' is pandas xlsb parser using pyxlsb engine.
- Updated dependencies.

7.6 1.3.2 (2021-05-14)

- Can now use both 'csv' and 'xlsb' files.
- Uses 'csv' file for storage and faster load times.
- Updated dependencies.

7.7 1.3.1 (2021-05-12)

- Uses pandas for now for faster resource loading.
- Uses xlsb file for storage and faster load times
- Updated dependencies.

7.8 1.3.0 (2021-05-11)

- Uses pandas for now for faster resource loading.
- In the process of integrating *faster-than-csv* when MacOs issues get resolved.
- Refactored and expanded the test suite.
- Updated dependencies.

7.9 1.2.7 (2021-05-07)

- The new method Lexique383.get_all_forms(word) is now accessible through the cli with option '-a' or '–all_forms'.
- This new method returns a list of LexItems having the same root lemma.
- Added sample commands using the new option in the docs.
- Refactored and expanded the test suite.
- Updated dependencies.

7.10 1.2.6 (2021-05-06)

- allows for new style of relative imports.
- Now all the attributes of the LexItem objects are immutable for consistency.
- Added new method Lexique383.get_all_forms(word) to get all the lexical variations of a word.
- This new method returns a list of LexItems having the same root lemma.
- Expanded sample usage of the software in the docs.
- Updated dependencies.

7.11 1.2.3 (2021-05-04)

- Enhanced behaviour of output to stdout to not conflict with the logging strategy of users importing the library in their own projects.
- Expanded sample usage of the software in the docs.
- Updated dependencies.

7.12 1.2.2 (2021-05-04)

- Enhanced Type Hinting for main module.
- Changed the property LexItem.islem to boolean instead of a binary choice 0/1.
- Expanded sample usage of the software in the docs.
- Updated dependencies.

7.13 1.2.1 (2021-04-30)

- Implemented Type Hinting for main module.
- Added a new method to the class Lexique383. The method is Lexique383._save_errors() .
- This new method checks that the value of each field in a LexItem is of the right type. If it finds errors it will record the mismatched value/type and save it in ./erros/errors.json
- Expanded sample usage of the software in the docs.
- Much better documentation including links to Lexique383 pages and manuals.

7.14 1.2.0 (2021-04-30)

- Added a new method to the class Lexique383. The method is Lexique383.get_lex() .
- This new method accepts either a single word as a string or an iterable of strings and will return the asked lexical information.
- Expanded sample usage of the software in the docs.
- Substantial update to the code and docs.
- Removed unneeded dependencies as I reimplement some functionality myself.

7.15 1.1.1 (2021-04-28)

- Added a new method to the class LexItem. The method is LexItem.to_dict() .
- This new method allows the LexItem objects to be converted into dicts with key/value pairs corresponding to the LexItem.
- This method allows easy display or serialization of the LexItem objects.
- Lexical Items having the same orthography are stored in a list at the word's orthography key to the LEXIQUE dict.
- Expanded sample usage of the software in the docs.
- Substantial update to the code and docs.

7.16 1.1.0 (2021-04-28)

- Drastically reduced dependencies by ditching HDF5 and bolcs as the package is now smaller, faster an easier to build.
- Lexical Items having the same orthography are stored in a list at the word's orthography key to the LEXIQUE dict.
- Implemented the "FlyWheel" pattern for light Lexical entries rsiding entirely in memory at run time.
- Added sample usage of the software in the docs.
- General update to the code and docs.

7.17 1.0.7 (2021-04-27)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

G

L

M

P

T