Sanskrit Text Documentation

Release 0.2.3

Hrishikesh Terdalkar

Jun 11, 2023

CONTENTS:

Sanskrit Text (Devanagari) Utility Functions

- Free software: GNU General Public License v3
- Documentation: https://sanskrit-text.readthedocs.io.

CHAPTER

ONE

FEATURES

- Syllabification
- Vara Viccheda
- Pratyāhāra Encoding-Decoding
- Uccāraa Sthāna Yatna Utility
- Several other utility functions

1.1 Sanskrit Text

Sanskrit Text (Devanagari) Utility Functions

- Free software: GNU General Public License v3
- Documentation: https://sanskrit-text.readthedocs.io.

1.1.1 Features

- Syllabification
- Vara Viccheda
- Pratyāhāra Encoding-Decoding
- Uccāraa Sthāna Yatna Utility
- Several other utility functions

1.1.2 Install

To install Sanskrit Text, run this command in your terminal:

```
$ pip install sanskrit-text
```

1.1.3 Credits

This package was created with Cookiecutter and the hrishikeshrt/cookiecutter-pypackage project template.

1.2 Installation

1.2.1 Stable release

To install Sanskrit Text, run this command in your terminal:

```
$ pip install sanskrit-text
```

This is the preferred method to install Sanskrit Text, 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.

1.2.2 From sources

The sources for Sanskrit Text can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/hrishikeshrt/sanskrit-text
```

Or download the tarball:

```
$ curl -OJL https://github.com/hrishikeshrt/sanskrit-text/tarball/master
```

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

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

1.3 Usage

To use Sanskrit Text in a project,

```
import sanskrit_text as skt
```

1.4 sanskrit_text

1.4.1 sanskrit_text package

Submodules

sanskrit_text.cli module

Console Script for `sanskrit-text`

`sanskrit_text.cli.`**`main`**`()`

Console Script for `sanskrit-text`

Module contents

Sanskrit Text Utility

`sanskrit_text.`**`ord_unicode`**`(`*ch: str*`)` → str

Get Unicode 4-character-identifier corresponding to a character

Parameters
ch (*str*) – Single character

Returns
4-character unicode identifier

Return type
str

`sanskrit_text.`**`chr_unicode`**`(`*u: str*`)` → str

Get a Unicode character corresponding to 4-chracater identifier

Parameters
u (*str*) – 4-character unicode identifier

Returns
Single character

Return type
str

`sanskrit_text.`**`form_pratyaahaara`**`(`*letters: List[str]*`)` → str

Form a pratyaahaara from a list of letters

`sanskrit_text.`**`resolve_pratyaahaara`**`(`*pratyaahaara: str*`)` → List[List[str]]

Resolve pratyaahaara into all possible lists of characters

`sanskrit_text.`**`clean`**`(`*text: str*, *punct: bool = False*, *digits: bool = False*, *spaces: bool = True*, *allow: Optional[list] = None*`)` → str

Clean a line of Sanskrit (Devanagari) text

Parameters

- **text** (*str*) – Input string
- **punct** (*bool, optional*) – If True, the punctuations are kept. The default is False.
- **digits** (*bool, optional*) – If True, digits are kept. The default is False.

- **spaces** (*bool, optional*) – If False, spaces are removed. It is recommended to not change the default value unless it is specifically relevant to a use-case. The default is True.
- **allow** (*list, optional*) – List of characters to allow. The default is None.

Returns
Clean version of the string

Return type
str

sanskrit_text.**split_lines**(*text: str*, *pattern='[\\r\\n]+'*) → List[str]

Split a string into a list of strings using regular expression

Parameters

- **text** (*str*) – Input string
- **pattern** (*regexp, optional*) – Regular expression corresponding to the split points. The default is r'[rn]+'.

Returns
List of strings

Return type
List[str]

sanskrit_text.**trim_matra**(*line: str*) → str

Trim matra from the end of a string

sanskrit_text.**is_laghu**(*syllable: str*) → bool

Checks if the current syllable is Laghu

sanskrit_text.**toggle_matra**(*syllable: str*) → str

Change the Laghu syllable to Guru and Guru to Laghu (if possible)

sanskrit_text.**marker_to_swara**(*m: str*) → str

Convert a Matra to corresponding Swara

sanskrit_text.**swara_to_marker**(*s: str*) → str

Convert a Swara to correponding Matra

sanskrit_text.**get_anunaasika**(*ch: str*) → str

Get the appropriate anunaasika from the character's group

sanskrit_text.**fix_anuswara**(*text: str*) → str

Check every anuswaara in the text and change to anunaasika if applicable

sanskrit_text.**get_syllables_word**(*word: str*, *technical: bool = False*) → List[str]

Get syllables from a Sanskrit (Devanagari) word

Parameters

- **word** (*str*) – Sanskrit (Devanagari) word to get syllables from. Spaces, if present, are ignored.
- **technical** (*bool, optional*) – If True, ensures that each element contains at most one Swara or Vyanjana. The default is False.

Returns
List of syllables

Return type
List[str]

sanskrit_text.**get_syllables**(*text: str*, *technical: bool = False*) → List[List[List[str]]]

Get syllables from a Sanskrit (Devanagari) text

Parameters

- **text** (*str*) – Sanskrit (Devanagari) text to get syllables from
- **technical** (*bool, optional*) – If True, ensures that each element contains at most one Swara or Vyanjana. The default is False.

Returns
List of syllables in a nested list format Nesting Levels: Text -> Lines -> Words

Return type
List[List[List[str]]]

sanskrit_text.**split_varna_word**(*word: str*, *technical: bool = True*) → List[str]

Obtain the Varna decomposition of a Sanskrit (Devanagari) word

Parameters

- **word** (*str*) – Sanskrit (Devanagari) word to be split.
- **technical** (*bool, optional*) – If True, a split, vowels and vowel signs are treated independently which is more useful for analysis, The default is True.

Returns
List of Varna

Return type
List[str]

sanskrit_text.**split_varna**(*text: str*, *technical: bool = True*, *flat: bool = False*) → List[List[List[str]]]

Obtain the Varna decomposition of a Sanskrit (Devanagari) text

Parameters

- **word** (*str*) – Sanskrit (Devanagari) text to be split.
- **technical** (*bool, optional*) – If True, a split, vowels and vowel signs are treated independently which is more useful for analysis, The default is True.
- **flat** (*bool, optional*) – If True, a single list is returned instead of nested lists. The default is False.

Returns

Varna decomposition of the text in a nested list format. Nesting Levels: Text -> Lines -> Words

- Varna decomposition of each word is a List[char].
- List of Varna decomposition of each word from a line.
- List of Varna decomposition of each line from the text.

If *flat=True*, Varna decomposition of the entire text is presented as a single list, also containing whitespace markers. Lines are separated by a newline character ‘n’ and words are separated by a space character ‘ ‘.

Return type
List[List[List[str]]] or List[str]

sanskrit_text.**join_varna**(*viccheda: str*, *technical: bool = True*) → str

Join Varna decomposition to form a Sanskrit (Devanagari) word

Parameters

- **viccheda** (*list*) – Viccheda output obtained by *split_varna_word* with *technical=True* (or output of *split_varna* with *technical=True* and *flat=True*) IMPORTANT: *technical=True* is necessary.
- **technical** (*bool*) – WARNING: Currently unused. Value of the same parameter passed to *split_varna_word*

Note: Currently only works for the viccheda generated with *technical=True*

Returns
s – Sanskrit word

Return type
str

sanskrit_text.**get_ucchaarana_vector**(*letter: str*, *abbrev=False*) → Dict[str, int]

Get ucchaarana sthaana and prayatna based vector of a letter

Parameters

- **letter** (*str*) – Sanskrit letter
- **abbrev** (*bool*) – If True, the output will contain English abbreviations otherwise, the output will contain Sanskrit names. The default is False.

Returns
vector – One-hot vector indicating utpatti sthaana, aabhyantara prayatna and baahya prayatna of a letter

Return type
Dict[str, int]

sanskrit_text.**get_ucchaarana_vectors**(*word: str*, *abbrev: bool = False*) → List[Tuple[str, Dict[str, int]]]

Get ucchaarana sthaana and prayatna based vector of a word or text

Parameters

- **word** (*str*) – Sanskrit word (or text)
- **abbrev** (*bool*) – If True, the output will contain English abbreviations otherwise, the output will contain Sanskrit names. The default is False.

Returns
vectors – List of (letter, vector)

Return type
List[Tuple[str, Dict[str, int]]]

sanskrit_text.**get_signature_letter**(*letter: str*, *abbrev: bool = False*) → Dict[str, str]

Get ucchaarana sthaana and prayatna based signature of a letter

Parameters

- **letter** (*str*) – Sanskrit letter

- **abbrev** (*bool*) – If True, the output will contain English abbreviations otherwise, the output will contain Sanskrit names. The default is False.

Returns
signature – utpatti sthaana, aabhyantara prayatna and baahya prayatna of a letter

Return type
Dict[str, str]

sanskrit_text.**get_signature_word**(*word: str*, *abbrev: bool = False*) → List[Tuple[str, Dict[str, str]]]

Get ucchaarana sthaana and prayatna based signature of a word

Parameters

- **word** (*str*) – Sanskrit word (or text) Caution: If multiple words are provided, the spaces are not included in the output list.
- **abbrev** (*bool*) – If True, the output will contain English abbreviations otherwise, the output will contain Sanskrit names. The default is False.

Returns
List of (letter, signature)

Return type
List[Tuple[str, Dict[str, str]]]

sanskrit_text.**get_signature**(*text: str*, *abbrev: bool = False*) → List[List[List[Tuple[str, Dict[str, str]]]]]

Get ucchaarana list of a Sanskrit text

Parameters

- **text** (*str*) – Sanskrit text (can contain newlines, spaces)
- **abbrev** (*bool*) – If True, the output will contain English abbreviations otherwise, the output will contain Sanskrit names. The default is False.

Returns
List of (letter, signature) for words in a nested list format Nesting Levels: Text -> Lines -> Words

Return type
List[List[List[Tuple[str, Dict[str, str]]]]]

sanskrit_text.**get_ucchaarana_letter**(*letter: str*, *dimension: int = 0*, *abbrev: bool = False*) → str

Get ucchaarana sthaana or prayatna of a letter

Parameters

- **letter** (*str*) – Sanskrit letter
- **dimension** (*int*) –
  - 0: sthaana
  - 1: aabhyantara prayatna
  - 2: baahya prayatna

  The default is 0.
- **abbrev** (*bool*) –

  If True,
  The output will contain English abbreviations

  Otherwise,
  The output will contain Sanskrit names

The default is False.

Returns
ucchaarana sthaana or prayatna of a letter

Return type
str

sanskrit_text.**get_ucchaarana_word**(*word: str*, *dimension: int = 0*, *abbrev: bool = False*) → List[Tuple[str, str]]

Get ucchaarana of a word

Parameters

- **word** (*str*) – Sanskrit word (or text)

  Caution: If multiple words are provided, the spaces are not included in the output list

- **dimension** (*int*) –

  - 0: sthaana
  - 1: aabhyantara prayatna
  - 2: baahya prayatna

  The default is 0.

- **abbrev** (*bool*) –

  If True,
  The output will contain English abbreviations

  Otherwise,
  The output will contain Sanskrit names

  The default is False.

Returns
List of (letter, ucchaarana)

Return type
List[Tuple[str, str]]

sanskrit_text.**get_ucchaarana**(*text: str*, *dimension: int = 0*, *abbrev: bool = False*) → List[List[List[Tuple[str, str]]]]

Get ucchaarana list of a Sanskrit text

Parameters

- **text** (*str*) – Sanskrit text (can contain newlines, spaces)
- **dimension** (*int*) –

  - 0: sthaana
  - 1: aabhyantara prayatna
  - 2: baahya prayatna

  The default is 0.

- **abbrev** (*bool*) –

  If True,
  The output will contain English abbreviations

Otherwise,
The output will contain Sanskrit names

The default is False.

Returns
List of (letter, ucchaarana) for words in a nested list format Nesting Levels: Text -> Lines -> Words

Return type
List[List[List[Tuple[str, str]]]]

sanskrit_text.**get_sthaana_letter**(*letter: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana_letter for sthaana

sanskrit_text.**get_sthaana_word**(*word: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana_word for sthaana

sanskrit_text.**get_sthaana**(*text: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana for sthaana

sanskrit_text.**get_aabhyantara_letter**(*letter: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana_letter for aabhyantara

sanskrit_text.**get_aabhyantara_word**(*word: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana_word for aabhyantara

sanskrit_text.**get_aabhyantara**(*text: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana for aabhyantara

sanskrit_text.**get_baahya_letter**(*letter: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana_letter for baahya

sanskrit_text.**get_baahya_word**(*word: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana_word for baahya

sanskrit_text.**get_baahya**(*text: str*, *abbrev: bool = False*)
Wrapper for get_ucchaarana for baahya

1.5 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:

1.5.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/hrishikeshrt/sanskrit-text/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/hrishikeshrt/sanskrit-text/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 :)

1.5.2 Get Started!

Ready to contribute? Here's how to set up *sanskrit-text* for local development.

1. Fork the *sanskrit-text* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/sanskrit-text.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 sanskrit-text
$ cd sanskrit-text/
$ 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 sanskrit-text tests
$ python setup.py test or pytest
$ 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.

1.5.3 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/hrishikeshrt/sanskrit-text/pull_requests and make sure that the tests pass for all supported Python versions.

1.5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_sanskrit_text
```

1.5.5 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:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

1.6 Credits

1.6.1 Development Lead

- Hrishikesh Terdalkar <hrishikeshrt@linuxmail.org>

1.6.2 Contributors

None yet. Why not be the first?

1.7 History

1.7.1 0.1.0 (2022-07-03)

- First release on PyPI.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX