Urdu BioMeter Documentation

Release 0.2.9

A. Sean Pue

Aug 14, 2019

CONTENTS:

CHAPTER

ONE

INSTALLATION

1.1 Stable release

To install Urdu BioMeter, run this command in your terminal:

```
$ pip install urdubiometer
```

This is the preferred method to install Urdu BioMeter, 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 From sources

The sources for Urdu BioMeter can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/urdubiometer/urdubiometer
```

Or download the tarball:

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

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

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

CHAPTER

TWO

API REFERENCE

A list of the full API reference of all public classes and functions is below.

Public members can (and should) be imported from *urdubiometer*, e.g.:

```
from urdubiometer import Scanner, GhazalScanner
```

2.1 Core Classes

class urdubiometer.**Scanner**(*transcription_parser*, *long_parser*, *short_parser*, *constraints*, *meters_list*, *find_feet=None*, *post_scan_filter=None*)

Scanner class.

Parameters

- **transcription_parser** (*graphtransliterator.GraphTransliterator*) – Transcription parser.
- **long_parser** (*graphtransliterator.GraphTransliterator*) – Long metrical unit parser.
- **short_parser** (*graphtransliterator.GraphTransliterator*) – Short metrical unit parser.
- **constraints** (*dict(str, dict(str, dict(str,list[str])))*) – Nested dict of constraints, organized by previous node, next node, previous production and finally a list of next productions, e.g. `{'-':{'-': 's_bs':['s_c']}}`.
- **meters_list** (*list[dict]*) – A list of dictionaries of meters, containing a meter regex and details.
- **find_feet** (*function*) – Method to add metrical feet to a scan
- **post_scan_filter** (*function*) – Filter to be applied after scan, used to narrow results.

class GhazalScanner(*meters_list=None*, *find_feet=None*, *meters_filter=None*, *with_mir=True*)

Ghazal scanner for Urdu ghazal.

Parameters

- **meters_list** (*: dict or None*) –
- **with_mir** (*bool*) – Allow Mir meters
- **meters_filter** (function or None) – filter returning subsection of scanner's meter list

GhazalScanner.**find_feet**(*scan*)
str: Finds feet based on a scan.

property meters_list
list of dict:: Meters list.

scan(*input*, *first_only=False*, *graph_details=False*, *show_feet=False*)
Scan input.

Parameters

- **input** (*str*) – Input string
- **first_only** (*bool*) – Return the first scan only
- **graph_details** (*bool*) – Return the graph details (list of NodeMatch)
- **show_feet** (*bool*) – Show metrical feet in scan. Default is *False*.

Returns if graph_details is False, a list of UnitMatch. if graph_details is True, a list of NodeMatch. None if no complete scans are found.

Return type list or None

transcribe(*input*)
Transcribe input using transcription parser.

Parameters **input** (*str*) – Input string

Returns Transcription of input string

Return type str

property translation_graph
urdubiometer.DirectedGraph:: Translation graph.

CHAPTER

THREE

CONTRIBUTOR COVENANT CODE OF CONDUCT

3.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

3.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

3.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

3.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at pue@msu.edu. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

3.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

CHAPTER

FOUR

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/urdubiometer/urdubiometer/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/urdubiometer/urdubiometer/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/urdubiometer.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 urdubiometer
$ cd urdubiometer
$ 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 urdubiometer 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.

4.3 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.md.
3. The pull request should work for Python 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/urdubiometer/urdubiometer/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

$ py.test tests.test_urdubiometer

4.5 Deploying

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

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

Travis will then deploy to PyPI if tests pass.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- A. Sean Pue <a@seanpue.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

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.

6.1 [Unrelease-Maybe]

- convert tokens to numbers inside ParserRules to improve performance
- convert rules to number in , or add detailed result option
- allow toggle of validation to decrease load speed
- add minimum tokens to match to graphparser edge constraints
- add coverage badge
- rename GhazalScanner to GhazalScanner
- variable rules for modern vs. classical language
- adjust code to move long and short unit markers to constants
- change rule_found to rule_id

6.2 [Unreleased-TODO]

- adjust license to support any additional software (if necessary)
- confirm settings/transcription.yml is compatible with new UTMO standard
- decide on parser/transliterator terminology
- remove visited from _add_subgraph_to_graph if cycle check means it's no longer necessary (?)
- add visualizations and check if _add_subgraph_to_graph() is broken
- add documentation
- fix contributors
- add requirements.txt
- fix code documentation and proofread
- add black to contributing
- raise warning/error in constraints if nothing matched or multiple matches

- add feet to mir meters

6.3 0.2.9 - 2019-08-14

- removed Markdown due to pypi errors, converted to RST
- fixes to CHANGELOG.rst due to bad conversion
- Adjusted setup.py to use find_namespace_packages

6.4 0.2.8 - 2019-08-13

- renamed DefaultScanner to GhazalScanner
- added basic Mir meters in settings/mir_meters.yml
- changed show_feet for missing Mir feet
- added with_mir parameter to GhazalScanner
- renamed default.py to ghazal.py
- rewrote constrained_parsers_of to allow for regular expressions in constraints and to reuse already generated parsers
- added translations/urdubiometer/messages
- adjusted settings/constraints.yml
- modified scanner/validate.py due to regex use in constraints
- preliminary tie in with Transifex
- added scripts/extract_strings.py, scripts/import_po.py, scripts/README.md
- adjusted .travis.yml
- added short long vowels to transcription.yml

6.5 0.2.7 - 2019-08-03

- fixed setup.cfg, setup.py, to correct bumpversion problem with single quotes

6.6 0.2.6 - 2019-08-03

- added black code formatting

6.7 0.2.5 - 2019-08-02

- change to .travis.yml repo name
- added python 3.7 to setup.py

6.8 0.2.4 - 2019-08-02

- adjusted urdubiometer/**init**.py to fix **all** and import
- changed "id" in meters_list to string and fixed tests in scanner/validate.py, settings/ghazal_meters.yml, test_scanner.py, test_urdubiometer.py
- modification to doc structure (following earlier docs, needs adjustment) #### Added
- added api.rst to docs (in progress)

6.9 0.2.3 - 2019-07-30

- added to PyPI
- added pyup
- added pypi, pyup badges to readme.md
- added notebooks/
- adjust docs/index and added docs/api.rst

6.10 0.2.2 - 2019-07-30

- added settings/*
- added urdubiometer/cli.py,
- added tests/test_scanner.py
- added scanner/*
- use of graphtransliterator using nodes as list rather than dict required rewrite of _minimize_ndfa()
- adjusted setup.py for markdown
- modified urdubiometer.py (minor)
- removed scanner.py

6.11 0.2.1 - 2019-07-28

- removed tests/test_graphparser.py
- removed urdubiometer/graphparser/* to replace with graphtransliterator
- removed graphparser from init.py
- adusted .travis.yml tags

6.12 0.2.0 - 2018-03-14

- added graphparser._types.py module with ParserRule, ParserOutput, OnMatchRule, WhiteSpace, and DirectedGraph classes
- added tests/test_graphparser.py
- added graphparser init and constructors: from_yaml_file, from_yaml, from_dict. They are cascaded: from_yaml_file calls from_yaml, which calls from_dict. Added a "raw" parameter, to from_dict as to whether or the dict needs to be processed from easy-reading format (default is True)
- added _unescape_charnames to graphparser module to unescape \N{CHARNAME} strings (from files, especially)
- added graphparser/validate.py to handle validation of raw and processed settings, using `Cerberus`
- created graphparser/initialize.py to convert rules, onmatch rules, and whitespace to internal types Rules, OnMatchRules, and Whitespace; and, to generate the parser's internal DirectedGraph
- added GraphParser.parse() method
- modified tests to fail
- updated contributing.md

6.13 0.1.2 - 2018-02-22

- initialized scanner.py and graphparser submodule
- added tests to check loading

6.14 0.1.1 - 2018-02-22

- fixed badges in README.md

6.15 0.1.0 - 2018-02-22

- added AUTHORS.md, CONTRIBUTING.md (from cookiecutter, converted to md from rst)
- added docs, adjusting for markdown and sphinx_rtd_theme; enabled Napo
- added requirements_dev.txt, the dev requirements for a virtualenv; included m2r, sphinx_rtd_theme, and
- added Makefile (generated by cookiecutter)
- added MANIFEST.in, with some changes for md
- added setup.cfg, setup.py (customized for markdown), and tox.ini
- added urdubiometer directory with cli.py, __init__.py, and urdubiometer.py (cookiecutter)
- added tests/test_urdubiometer.py (cookiecutter)
- generated module documentation using Sphinx
- updated README.md based off cookiecutter
- updated .gitignore

- adjusted .travis.yml (may need some work)

6.16 0.0.1 - 2018-02-21

- Added This CHANGELOG.md file to record changes.
- Added CODEOFCONDUCT.md contains guidelines for participation.
- README.md created. It links to readthedocs.org, which I have initialized, and travis-ci.
- added LICENSE.md file, which is BSD and (c) Michigan State University
- added .travis.yml file for travis-ci

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

u

INDEX