stimpool

Release 0.2.4

Mario E. Bermonti Pérez

Jun 23, 2022

CONTENTS

CHAPTER

ONE

STIMPOOL

Easily create stimuli pools for cognitive, learning, and psycholinguistics research

- GitHub repo: https://github.com/mario-bermonti/stimpool.git
- Documentation: https://stimpool.readthedocs.io/
- Free software: BSD-3 Clause

1.1 Features

- Easily create Spanish word pools for research
- Specify the characteristics that meet your needs
- Provide your own word pool or use the default one
- Get the cleaned pool or save it to a file

1.2 Getting Started

1.2.1 Installation

```
pip install stimpool
```

1.2.2 Usage

```
from stimpool import WordPool
words = ["gato", "canción", "oso", "otorrinolaringólogo"]
word_pool = WordPool(words)
word_pool.select_words_without_accented_characters()
print(word_pool.words)
```

Check the documentation for more details.

1.3 Contributing to this project

All contributions are welcome!

Will find a detailed description of all the ways you can contribute to stimpool in *the contributing guide*.

This is a beginner-friendly project so don't hesitate to ask any questions or get in touch with the project's maintainers.

Please review the *project's code of conduct* before making any contributions.

1.4 Author

This project was developed by Mario E. Bermonti-Pérez as part of his academic research. Feel free to contact me at mbermonti@psm.edu or mbermonti1132@gmail.com

1.5 Credits

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

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install stimpool, run this command in your terminal:

```
$ pip install stimpool
```

This is the preferred method to install stimpool, 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.

CHAPTER

THREE

WORD POOL (STIMPOOL.WORDS)

The WordPool class contains all the functionality for creating word pools. It contains methods for sampling from the pool, selecting words based on their characteristics, getting the cleaned word pool, and saving it to a file.

class stimpool.words.**WordPool**(*pool=None*, *clean_conjugation_suffix=True*)

Create word pools.

Create a word pool.

Parameters

- **pool** (*Iterable*) – Word pool that will be used to create subpool (the default is None, use default word pool)
- **clean_conjugation_suffix** (*bool*) – Specifies if suffixes that are used to identify word conjugations should be removed from the pool (Default=True)

Methods:

select_words_without_accented_characters()	Get words without accented characters.
select_words_of_length([min_len, max_len])	Get words of the length specified.
sample_pool(n[, reproducible])	Sample from the word pool.
save_pool([filename])	Save the word pool to a csv file.

Attributes:

words	Return the clean word pool.

select_words_without_accented_characters()

Get words without accented characters.

Accented characters:: á, é, í, ó, ú, ñ, ü

Return type None

select_words_of_length(*min_len=None*, *max_len=None*)

Get words of the length specified.

Parameters

- **min_len** (*int*) – Minimum word length (defaults to None; no min length). If a min length is not specified, a max length has to be specified.
- **max_len** (*int*) – Maximum word length (defaults to None; no max length). If a max length is not specified, a min length has to be specified.

Raises **ValueError** – If neither min_len nor max_len are specified.

Return type None

sample_pool(*n*, *reproducible=True*)
Sample from the word pool.

This is just a helper function that uses pandas.Series.sample. You can read its [complete documentation] (https://pandas.pydata.org/docs/reference/api/pandas.Series.sample.html)

Parameters

- **n** (*int*) – sample size
- **reproducible** (*bool*) – Specifies whether the sample obtained should be reproducible. This is important to guarantee the reproducibility of research (Default=True)

Return type None

save_pool(*filename='word pool'*)
Save the word pool to a csv file.

This is just a helper function that uses pandas.Series.to_csv. You can read its [complete documentation] (https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.Series.to_csv.html)

Parameters **filename** (*str*) – Name of the file without the extension (i.e., csv). (Default=word pool)

Return type None

property words
Return the clean word pool.

Return type Series

CHAPTER

FOUR

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.

4.1 Unreleased

4.2 0.2.4 - 2021-08-28

4.3 0.2.3 - 2021-08-28

- Fix an error with the sample_pool method Sampling now changes internal WordPool.words.

4.4 0.2.2 - 2021-08-25

- Improve the ease of use
- Improve the docs
- Improve the ci actions
- Minor improvements of internal codebase

4.5 0.2.1 - 2021-04-29

- Fix bug that that would build the wrong path to the root dir and did not allow the default word pool to be used.

4.6 0.2.0 - 2021-04-25

- Improve method for comparing series
- Improve type annotations
- Refactor to improve readability and maintainability
- Improve documentation

4.7 0.1.1 - 2021-03-26

4.7.1 Changed

- Includes the same code as v0.1.0 but I bump the version to try to fix a problem on the release action on the ci.

4.8 0.1.0 - 2021-03-03

4.8.1 Added

- First release on PyPI.

CHAPTER

FIVE

CONTRIBUTOR COVENANT CODE OF CONDUCT

5.1 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, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

5.2 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

5.3 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.

5.4 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.

5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at mbermonti1132@gmail or mbermonti@psm.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.

5.6 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

For answers to common questions about this code of conduct, see https://www.contributor-covenant.org/faq

CHAPTER

SIX

CONTRIBUTING

Thank you for your interest in improving this project.

All contributions are welcome and greatly appreciated! Every little bit improves the project and helps its users.

The following sections detail a variety of ways to contribute and how to get started.

Credit will always be given to the people making contributions.

If you do decide to work on an issue, please indicate so in a comment to the issue so it's assigned to you and other people don't work on it simultaneously.

This is a beginner-friendly project so don't hesitate to write a comment in the issue you are interested in if you have questions, would like to discuss some issue further, or you need help in any way.

6.1 Types of Contributions

6.1.1 Spread the word

Tell others about your experience with stimpool. You can share it on social media and follow it on GitHub.

6.1.2 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/mario-bermonti/stimpool/issues.

Please let us know about your experience using stimpool. You can tell us about the things that you like, the things that can be improved, and the things that you would like stimpool to do.

If you are proposing a feature:

- How it would help you and other users (mainly researchers).
- Explain all the details of 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 for this reason it may not be feasible to implement the feature or it may take some time.

6.1.3 Write Documentation

stimpool could always use more documentation. You can contribute to the documentation by:

- Fixing typographical, grammatical, or spelling errors.
- Improving documentation that is unclear or incorrect.
- Creating or improving examples and tutorials.
- Writing blog posts, articles, and similar content that share how you are using this project and your best practices with us.

6.1.4 Report Bugs

You can report bugs at https://github.com/mario-bermonti/stimpool/issues.

Please provide all the details that are asked when you create the issue to make sure it is understood correctly.

6.1.5 Fix existing bugs

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

6.1.6 Implement Features

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

6.2 Start contributing!

6.2.1 Set up the development environment

Ready to contribute? Here's how to set up stimpool in your local development environment.

You will need Python 3.6+ installed.

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

```
git clone git@github.com:your_name_here/stimpool.git
```

3. We use `poetry` to manage dependencies. Install it with the following command.

```
curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-
↪poetry.py | python -
```

4. Install the project, its dependencies, and the virtualenv:

```
poetry install
```

5. Make sure everything is working properly before making any changes by running `poetry run invoke dev-tasks`.

6.2.2 Development tasks (important side note)

In order to run anything inside the virtual environment every command has to be prefixed with `poetry run <command>`.

For example, to run python inside the virtual environment you would run `poetry run python`.

We have most of our development tasks pre-configured to run automatically with invoke. :grin:

The most important tasks are:

You can find see all the development tasks that pre-configured by running `poetry run invoke --list`.

6.2.3 Making changes

Workflow

We work by protecting `master` branch and only merging changes that don't break existing functionality and are tested.

How do we do it?:

1. We identify something that must change
2. We create an issue on GitHub, if it doesn't already exist
3. We create a new branch named after the issue we want "fix" (`issue-$TASKNUMBER`)
4. We make changes and test everything works
5. Style the code
6. We then create a pull request to `master` branch that is reviewed and if approved, it is merged into `master`

This way we achieve an easy and scalable development process that avoids merge conflicts and long-living branches.

In this method, the `master` branch always has the latest working version of the software, is stable, and is working.

How to make changes

Follow this steps when working on changes to the project. Please see the `Workflow section` for important details about making changes.

1. Create a branch for local development. All the changes must be in this branch.

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

2. Run the all checks to make sure everything is working before making any changes

```
$ poetry run invoke dev-tasks
```

3. Add any changes you want
4. Add tests for the new changes
5. Run the tests and make sure they all pass

```
$ poetry run invoke tests
```

6. Edit the documentation if appropriate (this is required for new features)
7. Make sure the changes to the documentation are correct and that the docs build

```
$ poetry run invoke docs
```

8. Make sure everything is fine (e.g., tests, code style, coverage)

```
$ poetry run invoke dev-tasks
```

If you find that something is not working as expected, fix it, check that it is working appropriately by running the appropriate invoke command (see `Development tasks section`).

```
$ poetry run invoke <command>
```

After it is fixed, run all development tasks again

```
$ poetry run invoke dev-tasks
```

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

```
$ git add .
$ git commit
```

Stimpool follows specific guidelines for commit messages:

- Make a reference to the relevant GitHub issues in your commit message (e.g., `Fix #1234`) We use imperative mood for commit messages (`fix x`, instead of `fixed x`). See this commit guide. A tip is to use a title for your commit message that completes "This commit will…" [Fix issue X].
- The subject line should have < 80 chars
- Leave one line blank
- [Optional] Explain any relevant details or decisions made

10. Push your changes to GitHub

```
$ git push origin name-of-your-bugfix-or-feature
```

11. Submit a pull request through GitHub (see the `Pull Request Guidelines` section).

Pull Request Guidelines

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

1. The pull request should include tests for new functionality.
2. If the pull request adds functionality, the docs should be updated.
3. The pull request should pass all tests and must work for all the supported Python versions. It must also pass all checks in the GitHub CI.

Feel free to submit your pull request early so we can discuss it and iterate on the process.

6.2.4 Tips

We really value your contributions and want to integrate your changes. The following are tips to improve the probability that your changes are accepted.

- Make sure they don't break existing functionality
- Include tests for the changes you made
- Commit often
- Make small, easy to understand commits (i.e., atomic commits)
- Keep your changes in the narrowest scope possible (e.g., create tutorial for using the `X object`)
- It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the maintainers and validate your approach.

6.3 Releasing stimpool

Maintainers, please review the guide for releasing new versions of stimpool on Github and Pypi.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

7.1 BSD 3-clause

INDEX

S

W