SpaCy Affixes Documentation

Release 0.1.4

LINHD POSTDATA Project

Nov 20, 2019

Contents:

CHAPTER 1

SpaCy Affixes

SpaCy support for affixes splitting for Freeling-like affixes rules and dictionaries.

- Free software: Apache Software License 2.0
- Documentation: https://spacy-affixes.readthedocs.io.

1.1 Usage

This library was born to split clitics from verbs so POS tagging works out-of-the-box with spaCy models.

```
from spacy_affixes import AffixesMatcher
nlp = spacy.load("es")
affixes_matcher = AffixesMatcher(nlp, split_on=["VERB"])
nlp.add_pipe(affixes_matcher, name="affixes", before="tagger")
for token in nlp("Yo mismamente podría hacérselo bien."):
    print(
        token.text,
        token.lemma_,
        token.pos_,
        token.tag_,
        token._.has_affixes,
        token._.affixes_rule,
        token._.affixes_kind,
        token._.affixes_text,
        token._.affixes_length,
    )
```

The output will be

```
Hay Hay AUX AUX__Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin False None␣
↪None None 0
que que SCONJ SCONJ___ False None None None 0
hacér hacer VERB  True suffix_selo suffix hacer 2
se se PRON PRON__Person=3 False None None None 0
lo el PRON PRON__Case=Acc|Gender=Masc|Number=Sing|Person=3|PronType=Prs False None␣
↪None None 0
todo todo PRON PRON__Gender=Masc|Number=Sing|PronType=Ind False prefix_todo None None␣
↪0
, , PUNCT PUNCT__PunctType=Comm False None None None 0
y y CONJ CCONJ___ False None None None 0
rápidamente rápidamente ADV ADV___ False suffix_mente None None 0
además además ADV ADV___ False prefix_a None None 0
. . PUNCT PUNCT__PunctType=Peri False None None None 0
```

However, words with suffixes could also be split if needed, or virtually any word for which a rule matches, just by passing a list of Universal Dependency POS's to the argument `split_on`. Passing in `split_on="*"` would make `AffixesMatcher()` try to split on everything it finds.

1.2 Rules and Lexicon

Due to licensing issues, `spacy-affixes` comes with no rules nor lexicons by default. There are two ways of getting data into `spacy-affixes`:

1. Create the rules and lexicon yourself with the entities you are interested on, and pass them in using `AffixesMatcher(nlp, rules=<rules>, dictionary=<dictionary>)`. The format for these is as follows.
   - rules: Dictionary of rules for affixes handling. Each dict uses a key that contains the pattern to match and the value is a list of dicts with the corresponding rule parameters:
     - pattern: Regular expression to match, (ex. `r"ito$"`) If a match is found, it gets removed from the token
     - kind: `AFFIXES_SUFFIX` or `AFFIXES_PREFIX`
     - pos_re: EAGLE regular expression to match, (ex. `r"V"`)
     - strip_accent: Boolean indicating whether accents should be stripped in order to find the rest of the token in the lexicon
     - affix_add: List of strings to add to the rest of the token to find it in the lexicon. Each element in the list is tried separately, as in an OR condition. The character `*` means add nothing (ex. `["*", "io"]`)
     - affix_text: List of Strings with the text to the rest of the token as individual tokens. For example, a rule for `dígamelo` might have `["me", "lo"]` as its `affix_text`
   - lexicon: Dictionary keyed by word with values for lemma, EAGLE code, UD POS, and UD Tags.
2. Convert the Freeling data. Take into account that if you use Freeling data you are effectively agreeing to their license, which might have implications in the release if your own code. If installed, `spacy-affixes` will look for the environment variables `FREELINGDIR` or `FREELINGSHARE` to find the affixes rules and dictionary files and will process them. If you don't have Freeling installed you can always run the `download` command:

```
python -m spacy_affixes download <lang> <version>
```

Where `lang` is the 2-character ISO 639-1 code for a supported language, and `version` an tagged version in their GitHub repository.

1.3 Notes

- Some decisions might feel idiosyncratic since the purpose of this library at the beginning was to just split clitics in Spanish texts.

CHAPTER 2

Installation

2.1 Stable release

To install SpaCy Affixes, run this command in your terminal:

```
$ pip install spacy-affixes
```

This is the preferred method to install SpaCy Affixes, 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.

2.2 From sources

The sources for SpaCy Affixes can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/linhd-postdata/spacy-affixes
```

Or download the tarball:

```
$ curl  -OL https://github.com/linhd-postdata/spacy-affixes/tarball/master
```

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

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

CHAPTER 3

Usage

To use SpaCy Affixes in a project:

```
import spacy_affixes
```

CHAPTER 4

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/linhd-postdata/spacy_affixes/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

SpaCy Affixes could always use more documentation, whether as part of the official SpaCy Affixes 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/linhd-postdata/spacy_affixes/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 *spacy_affixes* for local development.

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

```
$ git clone git@github.com:your_name_here/spacy_affixes.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 spacy_affixes
$ cd spacy_affixes/
$ 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 spacy_affixes 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.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/linhd-postdata/spacy_affixes/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_spacy_affixes
```

4.5 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
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- LINHD POSTDATA Project <info@linhd.uned.es>
- Javier de la Rosa versae.

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2019-04-02)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search