PyStanfordDependencies Documentation

Release stable

August 02, 2015

Contents

Python interface for converting Penn Treebank trees to Stanford Dependencies.

CHAPTER 1

Example usage

Start by getting a `StanfordDependencies` instance with `StanfordDependencies.get_instance()`:

```
>>> import StanfordDependencies
>>> sd = StanfordDependencies.get_instance(backend='subprocess')
```

`get_instance()` takes several options. `backend` can currently be `subprocess` or `jpype` (see below). If you have an existing Stanford CoreNLP or Stanford Parser jar file, use the `jar_filename` parameter to point to the full path of the jar file. Otherwise, PyStanfordDependencies will download a jar file for you and store it in locally (`~/.local/share/pystanforddeps`). You can request a specific version with the `version` flag, e.g., `version='3.4.1'`. To convert trees, use the `convert_trees()` or `convert_tree()` method (note that by default, `convert_trees()` can be considerably faster if you're doing batch conversion). These return a sentence (list of `Token` objects) or a list of sentences (list of list of `Token` objects) respectively:

```
>>> sent = sd.convert_tree('(S1 (NP (DT some) (JJ blue) (NN moose)))')
>>> for token in sent:
...     print token
...
Token(index=1, form='some', cpos='DT', pos='DT', head=3, deprel='det')
Token(index=2, form='blue', cpos='JJ', pos='JJ', head=3, deprel='amod')
Token(index=3, form='moose', cpos='NN', pos='NN', head=0, deprel='root')
```

This tells you that `moose` is the head of the sentence and is modified by `some` (with a `det` = determiner relation) and `blue` (with an `amod` = adjective modifier relation). Fields on `Token` objects are readable as attributes. See docs for additional options in `convert_tree()` and `convert_trees()`.

CHAPTER 2

Visualization

If you have the asciitree package, you can use a prettier ASCII formatter:

```
>>> print sent.as_asciitree()
 moose [root]
  +-- some [det]
  +-- blue [amod]
```

If you have Python 2.7 or later, you can use Graphviz to render your graphs. You'll need the Python graphviz package to call `as_dotgraph()`:

```
>>> dotgraph = sent.as_dotgraph()
>>> print dotgraph
digraph {
        0 [label=root]
        1 [label=some]
                3 -> 1 [label=det]
        2 [label=blue]
                3 -> 2 [label=amod]
        3 [label=moose]
                0 -> 3 [label=root]
}
>>> dotgraph.render('moose') # renders a PDF by default
'moose.pdf'
>>> dotgraph.format = 'svg'
>>> dotgraph.render('moose')
'moose.svg'
```

The Python xdot package provides an interactive visualization:

```
>>> import xdot
>>> window = xdot.DotWindow()
>>> window.set_dotcode(dotgraph.source)
```

Both `as_asciitree()` and `as_dotgraph()` allow customization. See the docs for additional options.

CHAPTER 3

Backends

Currently PyStanfordDependencies includes two backends:

- `subprocess` (works anywhere with a `java` binary, slow so batched conversions with `convert_trees()` are recommended)
- `jpype` (requires jpype1, faster than Subprocess, includes access to the Stanford CoreNLP lemmatizer)

By default, PyStanfordDependencies will attempt to use the `jpype` backend. If `jpype` isn't available or crashes on startup, PyStanfordDependencies will fallback to `subprocess` with a warning.

CHAPTER 4

Universal Dependencies status

PyStanfordDependencies mostly supports Universal Dependencies (see issue #10 for the most up to date status). PyStanfordDependencies output matches Universal Dependencies in terms of structure and dependency labels, but Universal POS tags and features are missing.

CHAPTER 5

More information

Licensed under Apache 2.0.

Written by David McClosky (homepage, code)

Bug reports and feature requests: GitHub issue tracker

CHAPTER 6

Release summaries

- 0.2.0 (2015.08.02): Universal Dependencies support (mostly), Python 3 support (fully), minor API updates
- 0.1.7 (2015.06.13): Bugfixes for JPype, support for IBM Java
- 0.1.6 (2015.02.12): Support for graphviz formatting, CoreNLP 3.5.1, better Windows portability
- 0.1.5 (2015.01.10): Support for ASCII tree formatting
- 0.1.4 (2015.01.07): Fix CCprocessed support
- 0.1.3 (2015.01.03): Bugfixes, coveralls integration, refactoring
- 0.1.2 (2015.01.02): Better CoNLL structures, test suite and Travis CI support, bugfixes
- 0.1.1 (2014.12.15): More docs, fewer bugs
- 0.1 (2014.12.14): Initial version