textract Documentation

Release 0.1.0

Dean Malmgren

July 27, 2014

Contents

As undesireable as it might be, more often than not there is extremely useful information embedded in Word documents, PowerPoint presentations, PDFs, etc—so-called "dark data"—that would be valuable for further textual analysis and visualization. While *several packages* exist for extracting content from each of these formats on their own, this package provides a single interface for extracting content from any type of file, without any irrelevant markup.

This package provides two primary facilities for doing this, the *command line interface*

```
textract path/to/file.extension
```

or the *python package*

```python
# some python file
import textract
text = textract.process("path/to/file.extension")
```

CHAPTER 1

Currently supporting

- `.doc` via antiword
- `.docx` via python-docx
- `.pptx` via python-pptx
- `.pdf` via pdftotext (default) or pdfminer
- `.txt` via python builtins.

Please recommend other file types by either mentioning them on the issue tracker or by *contributing*

CHAPTER 2

Installation

This package is built on top of several python packages and other source libraries. In particular, this package has a dependency on lxml that depends on some other libraries to be installed. On Ubuntu/Debian, you will need to run:

```
apt-get install python-dev libxml2-dev libxslt1-dev antiword poppler-utils
```

before running:

```
pip install textract
```

Contents:

2.1 Command line interface

This package ships with the `textract` command, which embodies the entire command line interface for this package. This command can be run on any supported file by simply running

```
textract path/to/some/file.extension
```

on any *supported file type*. Full documentation for the command line interface is available with the `-h/--help` command line option:

```
textract -h
```

To make the command line interface as usable as possible, autocompletion of available options with textract is enabled by @kislyuk's amazing argcomplete package. Follow instructions to enable global autocomplete and you should be all set. As an example, this is also configured in the virtual machine provisioning for this project.

2.2 Python package

The core of this package is in the `textract.parsers` submodule organized by file extension. For example, the `.docx` parser is located in `textract.parsers.docx`. Every parser submodule must have a method called extract that does the default text extraction for that file type.

2.2.1 textract.parsers

2.2.2 textract.parsers.doc

2.2.3 textract.parsers.docx

2.2.4 textract.parsers.pdf

2.2.5 textract.parsers.pptx

2.2.6 textract.parsers.txt

2.3 Contributing

The overarching goal of this project is to make it as easy as possible to extract raw text from any document for the purposes of most natural language processing tasks. In practice, this means that this project should preferentially provide tools that correctly produce output that has words in the correct order but that whitespace between words, formatting, etc is totally irrelevant.

Importantly, this project is committed to being as agnostic about how the content is extracted as it is about the means in which the text is analyzed downstream. This means that `textract` should support multiple modes of extracting text from any document and provide reasonably good defaults (defaulting to tools that tend to produce the correct word sequence).

Another important aspect of this project is that we want to have extremely good documentation. If you notice a type-o, error, confusing statement etc, please fix it!

2.3.1 Quick start

1. Fork and clone the project:

   ```
   git clone https://github.com/YOUR-USERNAME/textract.git
   ```

2. Install Vagrant and Virtualbox and launch the development virtual machine:

   ```
   vagrant up && vagrant provision
   ```

   On `vagrant ssh`ing to the virtual machine, note that the `PYTHONPATH` and `PATH` environment variables have been altered in this virtual machine so that any changes you make to textract in development are automatically incorporated into the command.

3. On the virtual machine, make sure everything is working by running the suite of functional tests:

   ```
   ./tests/run_functional_tests.sh
   ```

   These functional tests are designed to be run on an Ubuntu 12.04 LTS server, just like the virtual machine and the server that runs the travis-ci test suite. There are some other tests that have been added along the way in the Travis configuration. For your convenience, you can run all of these tests with:

   ```
   ./tests/run.py
   ```

   Current build status:

4. Contribute! There are several open issues that provide good places to dig in. Check out the contribution guidelines and send pull requests; your help is greatly appreciated!

2.4 changelog

This project uses semantic versioning to track version numbers, where backwards incompatible changes (highlighted in **bold**) bump the major version of the package.

2.4.1 latest

2.4.2 0.3.0

- support for `.txt` files, haha (#8)
- fixed installation bug with not properly including requirements files in the manifest

2.4.3 0.2.0

- support for `.doc` files (#2)
- support for `.pdf` files (#3)
- several bug fixes, including:
  - fixing tab complete bug no file paths (#6)
  - fixing tests to make sure the work properly on travis-ci

2.4.4 0.1.0

- Initial release, support for `.docx` and `.pptx`

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*