invenio-classifier Documentation

Release 1.0.0.dev20160404

CERN

September 06, 2016

Contents

Invenio module for record classification.

- Free software: GPLv2 license
- Documentation: https://pythonhosted.org/invenio-classifier

CHAPTER 1

Features

Classifier automatically extracts keywords from fulltext documents. The automatic assignment of keywords to textual documents has clear benefits in the digital library environment as it aids catalogization, classification and retrieval of documents.

CHAPTER 2

Keyword extraction is simple

Note: Classifier requires Python RDFLib in order to process the RDF/SKOS taxonomy.

In order to extract relevant keywords from a document `fulltext.pdf` based on a controlled vocabulary `thesaurus.rdf`, you would run Classifier as follows:

```
${INVENIO_WEB_INSTANCE} classifier extract -k thesaurus.rdf -f fulltext.pdf
```

Launching `${INVENIO_WEB_INSTANCE} classifier --help` shows the options available.

As an example, running classifier on document nucl-th/0204033 using the high-energy physics RDF/SKOS taxonomy (`HEP.rdf`) would yield the following results (based on the HEP taxonomy from October 10th 2008):

```
Input file: 0204033.pdf

Author keywords:
Dense matter
Saturation
Unstable nuclei

Composite keywords:
10  nucleus: stability [36, 14]
6  saturation: density [25, 31]
6  energy: symmetry [35, 11]
4  nucleon: density [13, 31]
3  energy: Coulomb [35, 3]
2  energy: density [35, 31]
2  nuclear matter: asymmetry [21, 2]
1  n: matter [54, 36]
1  n: density [54, 31]
1  n: mass [54, 16]

Single keywords:
61  K0
23  equation of state
12  slope
4  mass number
4  nuclide
3  nuclear model
3  mass formula
2  charge distribution
2  elastic scattering
2  binding energy
```

CHAPTER 3

Thesaurus

Classifier performs an extraction of keywords based on the recurrence of specific terms, taken from a controlled vocabulary. A controlled vocabulary is a thesaurus of all the terms that are relevant in a specific context. When a context is defined by a discipline or branch of knowledge then the vocabulary is said to be a *subject thesaurus*. Various existing subject thesauri can be found here.

A subject thesaurus can be expressed in several different formats. Different institutions/disciplines have developed different ways of representing their vocabulary systems. The taxonomy used by classifier is expressed in RDF/SKOS. It allows not only to list keywords but to specify relations between the keywords and alternative ways to represent the same keyword.

```
<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#scalar">
 <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.fieldtheoryscalar"/>
 <prefLabel xml:lang="en">scalar</prefLabel>
 <note xml:lang="en">nostandalone</note>
</Concept>

<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#fieldtheory">
 <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.fieldtheoryscalar"/>
 <prefLabel xml:lang="en">field theory</prefLabel>
 <altLabel xml:lang="en">QFT</altLabel>
 <hiddenLabel xml:lang="en">/field theor\w*/</hiddenLabel>
 <note xml:lang="en">nostandalone</note>
</Concept>

<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#Composite.fieldtheoryscalar">
 <compositeOf rdf:resource="http://cern.ch/thesauri/HEP.rdf#scalar"/>
 <compositeOf rdf:resource="http://cern.ch/thesauri/HEP.rdf#fieldtheory"/>
 <prefLabel xml:lang="en">field theory: scalar</prefLabel>
 <altLabel xml:lang="en">scalar field</altLabel>
</Concept>
```

In RDF/SKOS, every keyword is wrapped around a *concept* which encapsulates the full semantics and hierarchical status of a term - including synonyms, alternative forms, broader concepts, notes and so on - rather than just a plain keyword.

The specification of the SKOS language and various manuals that aid the building of a semantic thesaurus can be found at the SKOS W3C website. Furthermore, Classifier can function on top of an extended version of SKOS, which includes special elements such as key chains, composite keywords and special annotations.

CHAPTER 4

Keyword extraction

Classifier computes the keywords of a fulltext document based on the frequency of thesaurus terms in it. In other words, it calculates how many times a thesaurus keyword (and its alternative and hidden labels, defined in the taxonomy) appears in a text and it ranks the results. Unlike other similar systems, Classifier does not use any machine learning or AI methodologies - a just plain phrase matching using regular expressions: it exploits the conformation and richness of the thesaurus to produce accurate results. It is then clear that Classifier performs best on top of rich, well-structured, subject thesauri expressed in the RDF/SKOS language.

Happy hacking and thanks for flying Invenio-Classifier.

Invenio Development Team
Email: info@inveniosoftware.org
IRC: #invenio on irc.freenode.net
Twitter: http://twitter.com/inveniosoftware
GitHub: https://github.com/inveniosoftware/invenio-classifier
URL: http://inveniosoftware.org

4.1 User's Guide

This part of the documentation will show you how to get started in using Invenio-Classifier.

4.1.1 Installation

Invenio-Classifier is available on PyPi:

```
pip install "invenio-classifier>=1.0.0"
```

4.1.2 Configuration

4.1.3 Usage

4.2 API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

4.2.1 API Docs

Extraction API

Flask extension

4.3 Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

4.3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-classifier/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.

Fix Bugs

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

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/invenio-classifier/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 :)

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/invenio-classifier.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 invenio-classifier
$ cd invenio-classifier/
$ pip install -e .[all]
```

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 tests:

```
$ ./run-tests.sh
```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

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

```
$ git add .
$ git commit -s -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.

Pull Request Guidelines

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

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.com/inveniosoftware/invenio-classifier/pull_requests and make sure that the tests pass for all supported Python versions.

4.3.2 Changes

Version 0.1.0 (release 2015-08-19)

- Initial public release.

4.3.3 License

4.3.4 Authors

Invenio module for record classification.

- Alberto Rodriguez Peon <alberto.rodriguez.peon@cern.ch>
- Alberto Pepe <alberto.pepe@cern.ch>
- Benoit Thiell <bthiell@cfa.harvard.edu>
- Georgios Papoutsakis <georgios.papoutsakis@cern.ch>
- Jan Aage Lavik <jan.age.lavik@cern.ch>
- Jerome Caffaro <jerome.caffaro@cern.ch>
- Jiri Kuncar <jiri.kuncar@cern.ch>
- Lars Holm Nielsen <lars.holm.nielsen@cern.ch>
- Leonardo Rossi <leonardo.r@cern.ch>
- Nikolaos Kasioumis <nikolaos.kasioumis@cern.ch>
- Patrick Glauner <patrick.oliver.glauner@cern.ch>
- Peter Halliday <phalliday@cornell.edu>
- Roman Chyla <roman.chyla@gmail.com>
- Samuele Kaplun <samuele.kaplun@cern.ch>
- Tibor Simko <tibor.simko@cern.ch>