ArchivesOrg.Latin Documentation

Release 0.0.1

Thibault Clérice

September 22, 2016

What ?

This piece of software is intended to be used with the 11K Latin Texts produced by David Bamman (http://www.cs.cmu.edu/~dbamman/latin.html). It supports only the plain text formats and the metadata github repo CSV file. This has been tested with *Python3* only. I welcome any new functions or backward compatibility support.

How to install ?

- **With development version:**
    - Clone the repository : `git clone https://github.com/ponteineptique/archives_org_latin_to`
    - Go to the directory : `cd archives_org_latin_toolkit`
    - Install the source with develop option : `python setup.py install`
- **With pip:**
    - Install from pip : `pip install archives_org_latin_toolkit`

Example

The following example should run with the data in tests/test_data. The example can be run with `python example.py`

```python
# We import the main classes from the module
from archives_org_latin_toolkit import Repo, Metadata
from pprint import pprint

# We initiate a Metadata object and a Repo object
metadata = Metadata("./test/test_data/latin_metadata.csv")
# We want the text to be set in lowercase
repo = Repo("./test/test_data/archive_org_latin/", metadata=metadata, lowercase=True)

# We define a list of token we want to search for
tokens = ["ecclesiastico", "ecclesia", "ecclesiis", "&quot;"]

# We instantiate a result storage
results = []

# We iter over text having those tokens :
# Note that we need to "unzip" the list
for text_matching in repo.find(*tokens):

    # For each text, we iter over embeddings found in the text
    # We want 3 words left, 3 words right,
    # and we want to keep the original token (Default behaviour)
    for embedding in text_matching.find_embedding(*tokens, window=3, ignore_center=False):
        # We add it to the results
        results.append(embedding)

# We print the result (list of list of strings)
pprint(results)
```

Contents

4.1 Archives.org Latin Toolkit Documentation

4.2 Classes

class `archives_org_latin_toolkit.`**`Metadata`**(*csv_file*)

 Bases: `object`

 Metadata object for a file

   Parameters **`csv_file`** (`str`) – Path to the CSV file to parse

 `getDate`(*identifier*)

  Get the date of a text given its identifier

    Parameters **`identifier`** (`str`) – Filename or identifier

    Returns Date of composition

    Return type int

class `archives_org_latin_toolkit.`**`Text`**(*file*, *metadata=None*, *lowercase=False*)

 Bases: `object`

 Text reading object for archive_org

   Parameters

    • **`file`** (`str`) – File path

    • **`metadata`** (`Metadata`) – Metadata registry

    • **`lowercase`** (`bool`) – Clean Text will be in lowercase

   Variables

    • *name* – Name of the file

    • *composed* – Date of composition

 `clean`

  Clean version of the text : normalized space, remove new line, dehyphenize, remove punctuation and number.

 `cleanUp`()

  Clean textual information and free RAM

 `composed`

find_embedding(**strings*, *window=50*, *ignore_center=False*, *memory_efficient=True*)
Check if given string is in the file

> **Parameters**
>
>> • **strings** – Strings as multiple arguments
>>
>> • **window** – Number of lines to retrieve
>>
>> • **ignore_center** – Remove the word found from the embedding

has_strings(**strings*)
Check if given string is in the file

> **Parameters strings** – Strings as multiple arguments
>
> **Returns** If found, return True
>
> **Return type** bool

name

raw

class archives_org_latin_toolkit.**Repo**(*directory*, *metadata=None*, *lowercase=False*)
Bases: object

Repo reading object for archive_org

> **Parameters**
>
>> • **file** (*str*) – File path
>>
>> • **metadata** (Metadata) – Metadata registry
>>
>> • **lowercase** (*bool*) – Clean Text will be in lowercase

find(**strings*, *multiprocess=None*, *memory_efficient=True*)
Find files who contains given strings

> **Parameters**
>
>> • **strings** – Strings as multiple arguments
>>
>> • **multiprocess** (*int*) – Number of process to spawn
>>
>> • **memory_efficient** (*bool*) – Drop the content of files to avoid filling the ram with unused content
>
> **Returns** Files who are matching the strings
>
> **Return type** generator

get(*identifier*)
Get the Text object given its identifier

> **Parameters identifier** (*str*) – Filename or identifier
>
> **Returns** Text object
>
> **Return type** *Text*

4.3 Helpers

archives_org_latin_toolkit.**period**(*x*)
Parse a period in metadata. If there is multiple dates, returns the mean

---

> > **Parameters x** (*str*) – Value to parse
>
> > **Returns** Parsed numeral
>
> > **Return type** int

archives_org_latin_toolkit.**bce**(*x*)
> Format A BCE string

> > **Parameters x** (*str*) – Value to parse
>
> > **Returns** Parsed numeral
>
> > **Return type** str

archives_org_latin_toolkit.**__window__**(*array*, *window*, *i*)
> Compute embedding using i

> > **Parameters**
> >
> > - **strings** –
> >
> > - **window** – Number of word to take left, then right [len(result) = (2*window)+1]
> >
> > - **i** – Index of the word
> >
> > - **memory_efficient** (*bool*) – Drop the content of files to avoid filling the ram with unused content
>
> > **Returns** List of words

archives_org_latin_toolkit.**__find_multiprocess__**(*args*)
> Find files who contains given strings

> > **Parameters args** – Tuple where first element are Strings as list and second element is list of file objects
>
> > **Returns** Files who are matching the strings
>
> > **Return type** list

Indices and tables

- Importing Modules
- genindex
- modindex
- search

Symbols