Perdido: Python Library for Geoparsing and Geocoding French Texts

Ludovic Moncla[1,*], Mauro Gaio[2]

[1]Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS, UMR 5205, F-69621
[2]Université de Pau et des Pays de l'Adour, LMAP, UMR 5142, Pau, France

Abstract

This paper introduces the Perdido Python library for geoparsing and geocoding French texts. The architecture of the Perdido Geoparser, which includes three layers: back-office, API, and Python library, is outlined. We also provide details on the methods used in the development of the processing chain and the various tasks covered, such as named entity recognition and classification (NERC), and toponym resolution. Lastly, we showcase the different features of the Python library and explain how to use it. The library is built as an overlay using API services, enabling users to manipulate, visualize, and export the results of geoparsing and geocoding. A Jupyter notebook[1] is also provided to demonstrate all the functionalities implemented in the library.

Keywords

Geoparsing, geocoding, named entity recognition, toponym disambiguation

1. Introduction

This article presents the Perdido Python library for geoparsing of French texts. Geoparsing is a very important task in geographic information retrieval and more widely in Natural Language Processing (NLP). It is composed of two main subtasks: (1) named entity and spatial information recognition and classification (or *geotagging*) and (2) toponym resolution (or *geocoding*) [1]. Many definitions of the notion of named entities exist, but in a rather general way we can define the task of named entity recognition as the action of locating and categorizing in a text the words or groups of words (most often involving a proper noun), allowing to be the referent of a world object in a stable and unambiguous way. In the case of geoparsing, we are more specifically interested in locating geographical information, i.e. elements of the text referring to a place, a location (absolute or relative) or a moving object [2]. This is called geotagging. In addition, geoparsing also includes the resolution of named entities (or *entity linking*), which in this case can be summarized as the resolution of locations mentions (or toponyms). This is called geocoding. The objective of this task is to link place name instances with spatial footprints.

[1]https://github.com/ludovicmoncla/perdido/blob/main/notebooks/perdido-geoparser-GeoExT-ECIR23.ipynb

GeoExT 2023: First International Workshop on Geographic Information Extraction from Texts at ECIR 2023 April 2, 2023, Dublin, Ireland

*Corresponding author.
ludovic.moncla@insa-lyon.fr (L. Moncla); mauro.gaio@univ-pau.fr (M. Gaio)
https://ludovicmoncla.github.io (L. Moncla)
0000-0002-1590-9546 (L. Moncla); 0000-0002-8041-4240 (M. Gaio)

CEUR Workshop Proceedings (CEUR-WS.org)

Geocoding methods in the literature are divided into two categories: those that rely on external resources such as knowledge bases and gazetteers, and those that rely on trained models [3]. The former generally yield more accurate results, as the coordinates retrieved from a gazetteer typically correspond to a real location. However, they also require a disambiguation step. The latter, on the other hand, requires a large amount of labeled data but do not necessitate querying gazetteers or dealing with ambiguities. Ambiguities such as metonymy, homonymy, and name changes over time can also arise in geocoding [4].

The architecture presented in this article has been developed and enriched during different projects such as itinerary reconstruction from hike descriptions [5], mapping of Paris street names cited in a corpus of 19th century novels [6], and the retrieval and classification of named entities in encyclopedic articles [7].

2. The architecture

Perdido Geoparser is implemented in three layers: the back-office part hosted on a server, a REST API that exposes the back-office functionalities in the form of web services and the Python library that offers an extra layer to query the services and manipulate, visualize and export the results.

2.1. Back-office

Back-office implements a processing chain for geoparsing: pre-processing (tokenization, lemmatization, morpho-syntactic annotation), named entity recognition and classification and toponym resolution. The pre-processing steps are performed using Treetagger[1]. Named entity recognition and spatial information annotation rely on a dual cascade of transducers that use lexical resources and pattern descriptions (local context-free grammars, morpho-syntactic patterns, ...). The transducers are implemented within the Unitex[2] platform and act by insertion to tag named entities and spatial information in the text. The processing chain produces two output formats, an XML-TEI[3] format [8] file and a GeoJSON file. Figure 1 shows an excerpt of the markup used to annotate the named entity *la rivière d'Arques*. The GeoJSON file contains only the geospatial aspects of the named entity such as its spatial footprint, associated with its name or its nature.

2.2. API

A web service has been developed for each subtask of the processing chain so that they can be executed autonomously but also combined together by service composition [9]. This leaves the user free to use all or part of the different services. In addition to these services, we have also developed two stand-alone services for geoparsing and geocoding [10]. Our API is deployed using FastAPI framework[4] and the ASGI Python Uvicorn server[5].

[1] https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
[2] https://unitexgramlab.org
[3] https://tei-c.org
[4] https://fastapi.tiangolo.com
[5] https://www.uvicorn.org

```xml
<rs type="place" subtype="ene">
    <term type="place">
        <w pos="DET" lemma="le">la</w>
        <w pos="N" lemma="rivière">rivière</w>
    </term>
    <w pos="PREP" lemma="de">d'</w>
    <rs type="place">
        <name type="place">
            <w pos="NPr" lemma="">Arques</w>
            <location>
                <geo source="nominatim" rend="Arques-la-Bataille, [***]">1.126523 49.8806133</geo>
            </location>
        </name>
    </rs>
    <location>
        <geo source="nominatim" rend="L'Arques, Martin-Église, [***]">1.1127559 49.9062435</geo>
    </location>
</rs>
```

Figure 1: Extract of the XML-TEI output of Perdido for the annotation of the named entity *la petite rivière d'Arques*.

2.3. Python library

The Perdido[6] Python library is available as an open-source on GitHub and can also be easily installed through the PIP package management system[7]. This makes it convenient to integrate into a Python environment and use with minimal coding required.

The library provides three main classes: `Geoparser` and `Geocoder` which allow to call the corresponding web services of the API and `Perdido` which allows to manipulate, visualize and export the results. Other classes are also available, such as the `PerdidoCollection` class, which extends the role of the `Perdido` class for a set of documents processed by Perdido, or the `Token`, `Entity`, and `Toponym` classes, which provide various attributes and methods for retrieving and viewing the objects manipulated by the `Perdido` class.

The constructor of the `Geoparser` class takes several optional arguments in parameter: for both the geotagging and geocoding stages (these last parameters correspond to those of the constructor of the `Geocoder` class). Concerning the geotagging, the `version` parameter allows to select which version of the annotation cascades will be used among the two currently existing versions: *Standard* (default) and *Encyclopedie*. The *Standard* version has been developed for geotagging texts with a very important spatial dimension, such as descriptions of routes or hikes [5]. As its name indicates, the *Encyclopedie* version, has been adapted specifically for the processing of encyclopedic articles and allows annotating certain linguistic constructions specific to encyclopedic discourse and thus improves the stages of recognition and classification of named entities compared to the *Standard* version [7]. Concerning the geocoding, several parameters can be specified in order to filter the results and limit ambiguities when querying gazetteers. As an example it could be specified, the maximum number of locations returned for each toponym (`max_rows`), a country code (`country_code`), or a bounding box (`bbox`).

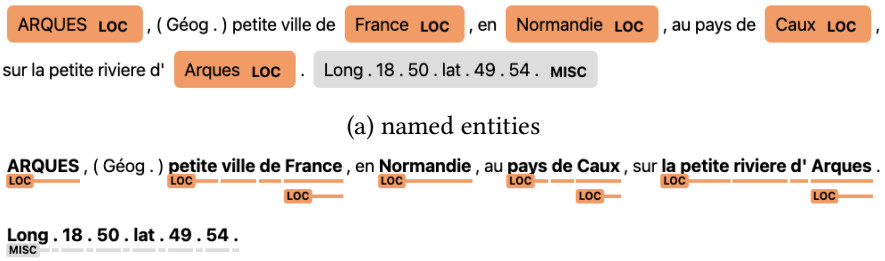
The methods `parse()` and `geocode()` of the `Geoparser` and `Geocoder` classes, respectively, call the geoparsing and geocoding web services of the API and return a `Perdido` object. These are

[6] https://github.com/ludovicmoncla/perdido
[7] https://pypi.org/project/perdido/

the methods that are executed when an instance of the classes `Geoparser` or `Geocoder` is used as a function. The method `parse()` takes as parameter the text that we want to geoparser and the method `geocode` takes as parameter a place name (or a list of place names) to geocode. For disambiguation, the method `cluster_disambiguation()` of the class `Perdido` implements a spatial density clustering (DBSCAN) [11] and makes it possible to remove a great number of ambiguities when the places of the text are close (an epsilon parameter is used to set the maximum distance for two points to be grouped within the same cluster).

2.3.1. Output formats, visualization and export of results

The `Perdido` class provides different attributes and methods to access the output formats and propose different ways of visualizing the geoparsed results. For example, the attribute `tei` allows to retrieve directly the XML-TEI format returned by the geoparsing web service (see Figure 1). The method `tsv_format()` of the class `Token` allows to retrieve tokens in TSV format according to the IOB (short for inside, outside, beginning) annotation scheme[8]. The TSV format allows to store one token per line and for each token: its index, its form, its lemma, its part of speech and its semantic category(ies). For display purpose, the `to_spacy_doc()` method is provided by the `Perdido` class. This method transforms a `Perdido` object into a SpaCy `Doc`[9] object, allowing to use the displaCy[10] library for NER visualization. Two modes are possible, the first one displays only named entities (i.e. proper names) (Fig. 2a), the second one displays nested named entities (Fig. 2b). Perdido provides also the `get_folium_map()` method for visualizing results on a map (Fig 3).



(a) named entities

(b) nested named entities

Figure 2: Display with displaCy of annotations produced by Perdido for the sentence: *"Arques, a small town in France, in Normandy, in the Pays de Caux, on the small river Arques. Long. 18. 50. lat. 49. 54."*

Finally, Perdido proposes several methods to export the results of geoparsing, such as the method `to_xml()`, which saves the content of the attribute `tei` in an XML file, the method `to_geojson()`, which saves the content of the attribute `geojson` in a json file, or the method `to_iob()`, which saves the results of the annotation of named entities in TSV format according

[8] IOB/BIO is a common tagging format for tagging tokens in a chunking task in computational linguistics, a token is annotated B-<tag> if it is the beginning of a chunk, I-<tag> indicates that the tag is inside a chunk. An O-<tag> indicates that a token belongs to no entity/chunk.

[9] https://spacy.io/api/doc

[10] https://spacy.io/universe/project/displacy

```
geoparser = Geoparser(version="Encyclopedie", bbox=[0.766296,49.680070,1.507874,49.988318])
doc = geoparser("Arques, petite ville de France, en Normandie, au pays de Caux, sur la petite rivière d'Arques.
doc.get_folium_map()
```

✓ 5.1s Python

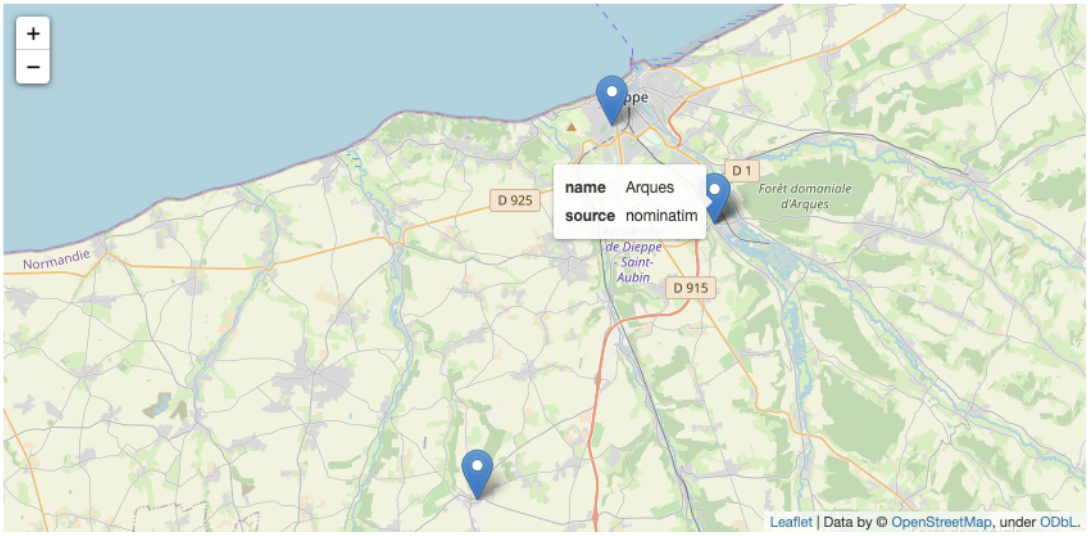


Figure 3: An example of using the geoparser and displaying the results.

to the IOB annotation scheme. These methods take as parameter the path to which the user wants to save the files.

2.3.2. Datasets

Two datasets are currently available in the library. The first contains 3,385 encyclopedic articles (corresponding to volume 7 of Diderot and d'Alembert's Encyclopedia (1751-1772)), provided by ARTFL[11] within the framework of the GEODE[12] project. The second one contains 30 descriptions of hikes collected in the framework of the ANR CHOUCAS[13] project, where each description is associated with its GPS track.

3. Perspectives

This article describes the overall architecture of the Perdido geoparsing tool and the recent development of its Python library. The library offers two main functions: geoparsing and geocoding of French texts. However, it is still a work in progress, and several improvements are planned. One proposed improvement is the implementation of a trained model for automatic annotation of nominal entities (or unnamed entities) upstream of the existing annotation cascade.

[11] https://artfl-project.uchicago.edu
[12] https://geode-project.github.io
[13] http://choucas.ign.fr

Another improvement being considered is the use of machine learning to train models, which will be integrated with the current approach to make it more versatile for analyzing diverse texts. Besides technical improvements, we also plan to conduct an evaluation campaign using benchmarks and our own corpora for the comparison of our approach with baselines.

Additionally, several other options are being explored for the geocoding step, such as using centroids, distances, or interpreting the spatial context extracted from the text to improve toponym disambiguation.

Acknowledgments

The authors are grateful to the ASLAN project (ANR-10-LABX-0081) of the Université de Lyon, for its financial support within the French program "Investments for the Future" operated by the National Research Agency (ANR).

References

[1] M. Gritta, M. T. Pilehvar, N. Limsopatham, N. Collier, What's missing in geographical parsing?, Language Resources and Evaluation 52 (2018) 603–623.

[2] J. L. Leidner, Toponym resolution in text: Annotation, evaluation and applications of spatial grounding, SIGIR Forum 41 (2007) 124–126.

[3] J. Fize, L. Moncla, B. Martins, Deep learning for toponym resolution: Geocoding based on pairs of toponyms, ISPRS International Journal of Geo-Information 10 (2021) 818.

[4] D. Buscaldi, Approaches to disambiguating toponyms, SIGSPATIAL Special 3 (2011) 16–19.

[5] M. Gaio, L. Moncla, Geoparsing and geocoding places in a dynamic space context, The Semantics of Dynamic Space in French: Descriptive, experimental and formal studies on motion expression 66 (2019) 354–386.

[6] L. Moncla, M. Gaio, T. Joliveau, Y.-F. Le Lay, N. Boeglin, P.-O. Mazagol, Mapping urban fingerprints of odonyms automatically extracted from french novels, International Journal of Geographical Information Science 33 (2019) 2477–2497.

[7] D. Vigier, L. Moncla, A. Brenon, K. Mcdonough, T. Joliveau, Classification des entités nommées dans l'encyclopédie ou dictionnaire raisonné des sciences des arts et des métiers par une société de gens de lettres (1751-1772), in: 7ème Congrès Mondial de Linguistique Française, 2020.

[8] L. Moncla, M. Gaio, A multi-layer markup language for geospatial semantic annotations, in: Proceedings of the 9th Workshop on Geographic Information Retrieval, 2015, pp. 1–10.

[9] M. S. Halilali, E. Gouardères, M. Gaio, F. Devin, Geospatial web services discovery through semantic annotation of wps, ISPRS International Journal of Geo-Information 11 (2022) 254.

[10] L. Moncla, M. Gaio, Services web pour l'annotation sémantique d'information spatiale à partir de corpus textuels, Revue Internationale de Géomatique 28 (2018) 439–459.

[11] L. Moncla, W. Renteria-Agualimpia, J. Nogueras-Iso, M. Gaio, Geocoding for texts with fine-grain toponyms: an experiment on a geoparsed hiking descriptions corpus, in: Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas, TX, 2014, p. 183–192.