Perdido : librairie Python pour le geoparsing et le geocoding de textes en français

Ludovic Moncla, Mauro Gaio

Perdido : librairie Python pour le geoparsing et le geocoding de textes en français

Ludovic Moncla*, Mauro Gaio**

*Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS, UMR 5205, F-69621
ludovic.moncla@insa-lyon.fr
https://ludovicmoncla.github.io
**Université de Pau et des Pays de l'Adour, LMAP, UMR 5142, Pau, France
mauro.gaio@univ-pau.fr

Résumé. Cet article présente la librairie Python Perdido pour le geoparsing et le geocoding de textes en français. Nous présentons l'architecture générale de l'outil Perdido composée de trois couches : back-office, API et librairie Python. Nous détaillons les méthodes utilisées pour le développement de la chaîne de traitement et des différentes tâches (reconnaissance et classification des entités nommées et résolution des toponymes). Enfin, nous présentons les différentes fonctionnalités de la librairie Python et la façon de l'utiliser. La librairie est développée comme une surcouche faisant appel aux services de l'API et permet de manipuler, visualiser et exporter les résultats du geoparsing et du geocoding. Un notebook [1] Jupyter décrit, sous la forme d'un tutoriel, l'ensemble des fonctionnalités implémentées dans la librairie.

1 Introduction

Cet article présente la librairie Python Perdido [2] pour le geoparsing et le geocoding de textes en français. Le geoparsing est une tâche très importante en recherche d'information géographique (Jones et Purves, 2008) et plus largement en Traitement Automatique des Langues (TAL). Elle se décompose en deux sous-tâches : (1) la reconnaissance et la classification d'entités nommées et d'informations spatiales (ou *geotagging*) et (2) la résolution de toponymes (ou *geocoding*). De nombreuses définitions de la notion d'entités nommées existent, mais de manière assez générale nous pouvons définir la tâche de reconnaissance d'entités nommées comme l'action de repérer et de catégoriser dans un texte les mots ou groupes de mots (le plus souvent des noms propres ou descriptions définies), permettant d'identifier un objet de manière stable et non ambiguë (Nouvel et al., 2015). Dans le cas du geoparsing, nous nous intéressons plus spécifiquement au repérage d'informations spatiales (ou géographiques) c'est-à-dire d'éléments du texte faisant référence à un lieu, une localisation (absolue ou relative) ou encore un déplacement. On parle alors de geotagging. En complément, le geoparsing comprend également l'étape de résolution des entités nommées (ou *entity linking*), qui dans ce cas peut

1. https://github.com/ludovicmoncla/demo-perdido-egc-2023
2. https://github.com/ludovicmoncla/perdido

se résumer à la résolution des entités de lieux (ou toponymes). On parle alors de geocoding. L'objectif de cette tâche est d'associer aux toponymes repérés dans le texte leurs coordonnées géographiques (ex : latitude / longitude).

De nombreuses méthodes de geocoding ont été proposées dans la littérature (Hu et al., 2022). De manière simplifiée, elles peuvent être classées en deux catégories : les méthodes qui s'appuient sur des ressources externes (bases de connaissances, gazetiers, ...) et les méthodes qui s'appuient sur l'entraînement de modèles pour la prédiction de coordonnées (Fize et al., 2021). La première catégorie permet en général d'obtenir des résultats plus précis (en effet, lorsqu'elles existent, les coordonnées retrouvées dans une base de connaissances font référence à une localisation réelle) mais nécessite une étape importante de désambiguïsation. La seconde catégorie nécessite une très grande quantité de données mais permet de ne pas avoir à interroger de gazetiers ou à traiter les ambiguïtés. De nombreuses formes d'ambiguïtés existent (Gritta et al., 2018), telles que la métonymie, l'homonymie ou les changements de noms au cours du temps. Le paramétrage du geocoding au sein de la librairie permet de filtrer un certain nombre d'ambiguïtés.

L'architecture présentée dans cet article a été développée et enrichie lors de différents projets tels que la reconstruction d'itinéraire à partir de descriptions de randonnées (Moncla et al., 2016; Gaio et Moncla, 2019), la cartographie des noms de rues Parisiennes citées dans un corpus de romans du XIXème siècle (Moncla et al., 2019) et le repérage et la classification des entités nommées dans les articles encyclopédiques (Vigier et al., 2020).

2 L'architecture

L'outil Perdido est implémenté en trois couches : la partie back-office hébergée sur un serveur, une API REST qui permet d'exposer les fonctionnalités du back-office sous forme de services web et la librairie Python qui offre une sur-couche pour interroger les services et manipuler, visualiser et exporter les résultats.

2.1 Back-office

Le back-office implémente une chaîne de traitement pour le geoparsing (Gaio et Moncla, 2019), qui reprend les différentes étapes classiques en TAL : pré-traitement (tokenisation, lemmatisation, annotation morpho-syntaxique), reconnaissance et classification des entités nommées et résolution des toponymes.

Les étapes de pré-traitement sont réalisées par l'outil Treetagger [3]. La reconnaissance des entités nommées et l'annotation des informations spatiales reposent sur une double cascade de transducteurs (Gaio et Moncla, 2019) qui utilisent des ressources lexicales et des descriptions locales de motifs (patrons morpho-syntaxiques, expressions régulières, ...). Les transducteurs sont implémentés au sein de la plateforme Unitex [4] et agissent par insertion afin de baliser les entités nommées et informations spatiales du texte. Ils permettent de produire une sortie au format XML-TEI [5] (Moncla et Gaio, 2015). La résolution des toponymes s'appuie sur l'interrogation de plusieurs bases de données géographiques (gazetiers). Les gazetiers suivants peuvent

3. https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
4. https://unitexgramlab.org
5. https://tei-c.org

```
<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>
```

FIG. 1 – Extrait de la sortie XML-TEI de Perdido pour l'annotation de l'entité nommée « *la petite rivière d'Arques* ».

être requêtés : l'API Nominatim [6] (OpenStreetMap), l'API du Géoportail [7] (IGN), Geonames [8] ainsi que le World Historical Gazetteer [9] et Pleiades [10] pour les données historiques et les données de l'antiquité.

La chaîne de traitement produit deux formats en sortie, un fichier XML-TEI et un fichier GeoJSON. Le fichier XML-TEI contient une version annotée et enrichie du document donné en entrée. La figure 1 montre un extrait du balisage utilisé pour annoter l'entité nommée « la rivière d'Arques ». Le fichier GeoJSON contient uniquement les informations géospatiales associées aux entités de lieux repérées dans le texte telles que la géométrie de l'objet, son nom ou sa nature.

2.2 API REST

Un service Web a été développé pour chaque sous-tâche de la chaîne de traitement afin qu'elles puissent être exécutées de manière autonome mais également combinées entre elles par composition de services (Halilali et al., 2022). L'objectif est de fournir différents services mais de laisser la possibilité à l'utilisateur d'utiliser des services tiers pour composer lui-même sa chaîne de traitement. Les services sont documentés et fonctionnent avec un système d'entrée / sortie afin que différents services offrant la même fonctionnalité puissent être interopérables. En complément de ces services « atomiques », nous avons également développé un service proposant l'ensemble de la chaîne de geoparsing et un service de geocoding (Moncla et Gaio, 2018).

Pour le déploiement de notre API nous avons utilisé le framework FastAPI [11] et le ser-

6. https://nominatim.org/release-docs/latest/
7. https://geoservices.ign.fr
8. http://www.geonames.org
9. https://whgazetteer.org
10. https://pleiades.stoa.org
11. https://fastapi.tiangolo.com

veur ASGI Python Uvicorn [12]. Une page de documentation est automatiquement créée et il est possible de tester directement l'API depuis un navigateur [13].

2.3 Librairie Python

La librairie Python Perdido est disponible en open source sur GitHub [2] mais également au sein du système de gestion de paquets PIP [14]. De cette manière, il est très simple de l'installer (ainsi que toutes ses dépendances) dans un environnement Python et de l'utiliser en seulement quelques lignes de code (voir Listing 1).

```
1 from perdido.geoparser import Geoparser
2 geoparser = Geoparser()
3 doc = geoparser('Je visite la ville de Lyon, Annecy et Chamonix.')
4 for entity in doc.named_entities:
5     print(f'entity: {entity.text}\ttag: {entity.tag}')
```

Listing 1 – Import et exemple d'utilisation de la classe `Geoparser`.

La librairie fournit trois principales classes : `Geoparser` et `Geocoder` qui permettent de faire appel aux services web correspondant de l'API et `Perdido` qui permet de manipuler, visualiser et exporter les résultats. D'autres classes sont également disponibles, comme par exemple la classe `PerdidoCollection`, qui étend le rôle de la classe `Perdido` pour un ensemble de documents traités par Perdido, ou les classes `Token`, `Entity`, et `Toponym`, qui offrent divers attributs et méthodes pour la récupération et la visualisation des objets manipulés par la classe `Perdido`.

Le constructeur de la classe `Geoparser` prend plusieurs arguments optionnels en paramètre : ceux qui servent à paramétrer l'étape de geotagging et ceux utilisés par l'étape de geocoding (ces derniers correspondent à ceux du constructeur de la classe `Geocoder`). Concernant le geotagging, il y a en particulier un paramètre (`version`) qui permet de sélectionner quelle version des cascades d'annotation sera exécutée parmi les deux versions existantes actuellement : *Standard* et *Encyclopédie*. Si le paramètre n'est pas présent, la version *Standard* est exécutée par défaut. Cette version a été développée pour le geotagging de textes ayant une dimension spatiale très importante, comme par exemple des descriptions d'itinéraires ou de randonnées (Moncla et al., 2014, 2016; Gaio et Moncla, 2019). La version *Encyclopédie*, comme son nom l'indique, a été adaptée spécifiquement pour le traitement d'articles encyclopédiques et permet d'annoter certaines constructions linguistiques spécifiques au discours encyclopédique et améliore ainsi les étapes de reconnaissance et de classification des entités nommées par rapport à la version *Standard* (Vigier et al., 2020; Moncla et al., 2021). Concernant le geocoding, plusieurs paramètres peuvent être renseignés afin de filtrer les résultats et limiter les ambiguïtés lors de l'interrogation des gazetiers. Il est par exemple possible de spécifier le nombre maximum de localisations retournées pour chaque toponyme (`max_rows`), un code pays (`country_code`), ou encore une bounding box (`bbox`).

Les méthodes `parse()` de la classe `Geoparser` et `geocode()` de la classe `Geocoder` font appel aux services web de geoparsing et de geocoding de l'API et retournent un objet de type `Perdido`. Ce sont ces méthodes qui sont exécutées lorsqu'une instance des classes

12. `https://www.uvicorn.org`
13. `http://choucas.univ-pau.fr/docs`
14. `https://pypi.org/project/perdido/`

`Geoparser` ou `Geocoder` est utilisée comme une fonction (c'est par exemple le cas à la ligne 3 du Listing 1). La méthode `parse()` prend en paramètre le texte que l'on souhaite geoparser et la méthode `geocode` prend en paramètre un nom de lieu (ou une liste de noms de lieux) à géocoder. Pour la désambiguïsation, la méthode `cluster_disambiguation()` de la classe `Perdido` implémente un clustering par densité spatiale (DBSCAN) et permet de lever un grand nombre d'ambiguïtés lorsque les lieux du texte sont proches (un paramètre epsilon permet de fixer la distance maximale pour que deux points soit regroupés au sein d'un même cluster) (Moncla et al., 2014).

2.3.1 Formats de sortie, visualisation et export des résultats

La classe `Perdido` fournit différents attributs et méthodes pour accéder aux formats de sortie et proposer différents modes de visualisation des résultats du geoparsing. Par exemple, l'attribut `tei` permet de récupérer directement le format XML-TEI retourné par le service Web de geoparsing (voir Figure 1).

La méthode `tsv_format` de la classe `Token` permet de récupérer les tokens au format TSV selon le schéma d'annotation BIO (Beginning, Inside, Outside). Un token est annoté B-<tag> s'il est le premier token d'une entité de type *tag*, I-<tag> s'il appartient à une entité de type *tag* sans être son premier élément et O s'il ne fait partie d'aucune entité. Le format TSV permet de stocker un token par ligne et pour chaque token : son indice, sa forme, son lemme, sa partie du discours et sa ou ses catégories sémantiques (voir Figure 2).

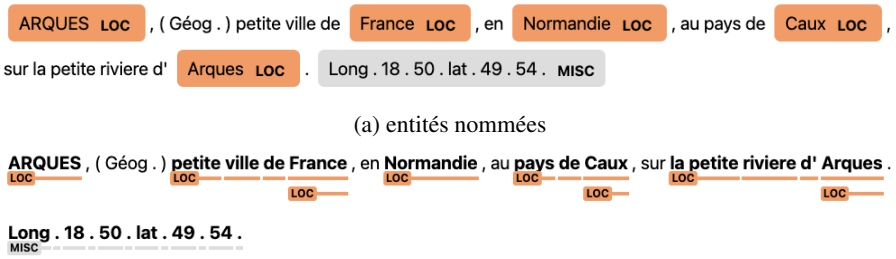
```
1       ARQUES  arquer  V       B-LOC
2       ,               PUN     O
3       (       (       PUN     O
4       Géog    géog    NPr     O
5       .               PUN     O
6       )       )       PUN     O
7       petite  petit   A       B-LOC-NNE
8       ville   ville   N       I-LOC-NNE
9       de      de      PREP    I-LOC-NNE
10      France  france  NPr     I-LOC-NNE       B-LOC
```

FIG. 2 – Exemple d'annotation BIO au format TSV.

Afin de pouvoir afficher les résultats de manière plus graphique, la classe `Perdido` propose la méthode `to_spacy_doc()`, qui transforme un objet de type `Perdido` en objet `Doc` [15] de la librairie spaCy. Cette transformation permet d'utiliser la librairie displaCy [16] qui peut par exemple s'utiliser au sein d'un notebook Jupyter. Deux modes de visualisation sont possibles, le premier n'affiche que les entités nommées au sens classique du terme (i.e., noms propres ou expressions figées) (Fig. 3a), le deuxième permet d'afficher les entités nommées imbriquées ou étendues (Fig. 3b).

Perdido permet également de visualiser les résultats sous la forme d'une carte géographique (Fig 4). Pour cela elle propose la méthode `get_folium_map()`, qui s'appuie sur la librairie Python Folium [17], afin d'afficher les marqueurs correspondants aux toponymes contenus dans l'attribut `geojson`. La librairie Folium permet ensuite de manière très simple d'exporter le résultat au format image.

15. `https://spacy.io/api/doc`
16. `https://spacy.io/universe/project/displacy`
17. `http://python-visualization.github.io/folium/`



(a) entités nommées

(b) entités nommées imbriquée ou étendues

FIG. 3 – Affichage avec displaCy des annotations produites par Perdido

Enfin, Perdido propose plusieurs méthodes pour exporter les résultats du geoparsing, comme par exemple la méthode `to_xml()`, qui enregistre le contenu de l'attribut `tei` dans un fichier XML, la méthode `to_geojson()`, qui enregistre le contenu de l'attribut `geojson` dans un fichier json ou encore la méthode `to_iob()`, qui enregistre les résultats de l'annotation des entités nommées au format TSV selon le schéma d'annotation IOB (Fig 2). Ces méthodes prennent en paramètre le chemin vers lequel l'utilisateur souhaite enregistrer les fichiers.

2.3.2 Les jeux de données

Deux jeux de données sont actuellement disponibles dans la librairie. Le premier contient 3 385 articles encyclopédiques (correspondant au volume 7 de l'Encyclopédie de Diderot et d'Alembert (1751-1772)), fournis par l'ARTFL [18] dans le cadre du projet GEODE [19]. Le deuxième contient 30 descriptions de randonnées collectées dans le cadre du projet ANR CHOUCAS [20], où chaque description est associée à sa trace GPS.

Le jeu de données *encyclopédie* est présent dans la librairie en deux versions, une version « brute » (`Dataframe`) et une version déjà annotée par Perdido (`PerdidoCollection`). Ces deux versions peuvent être récupérées grâce aux fonctions `load_edda_artfl()` et `load_edda_perdido()`. Le jeu de descriptions de randonnées est également déjà annoté (`PerdidoCollection`) et est accessible via la fonction `load_choucas_perdido()`.

3 Perspectives

Cet article présente l'architecture générale de l'outil de geoparsing Perdido ainsi que le récent développement de la librairie Python associée. Cette librairie propose deux principales fonctionnalités : le geoparsing et le geocoding à partir de textes en français. Il s'agit d'un travail toujours en cours de développement et de nombreuses améliorations sont envisagées. Une première piste concerne l'ajout d'un modèle entraîné pour l'annotation automatique des entités nominales (ou entités non nommées). Ce modèle sera intégrée en amont de la cascade d'annotation existante. Une deuxième piste s'intéresse à l'entraînement de modèles par apprentissage

18. `https://artfl-project.uchicago.edu`
19. `https://geode-project.github.io`
20. `http://choucas.ign.fr`

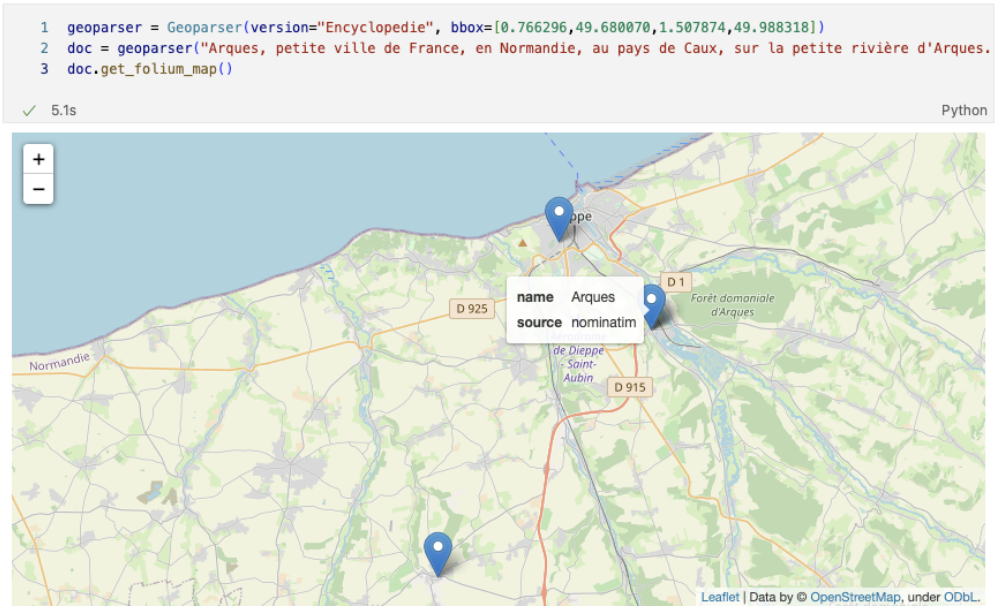


FIG. 4 – Exemple d'utilisation du geoparser et d'affichage des résultats.

automatique avec comme objectif d'être combinés à l'approche existante afin de généraliser le traitement pour des types de textes hétérogènes. Nous envisageons aussi d'adapter notre format d'annotation et les formats de sortie pour se rapprocher des standards existant comme par exemple le jeu d'étiquettes CONLL-U [21].

Plusieurs autres pistes sont également envisagées concernant l'étape de geocoding. En particulier pour la désambiguïsationn des toponymes avec l'implémentation de différentes solutions telles que le calcul de centroïdes, de distances ou encore l'interprétation du contexte spatiale extrait du texte.

Références

Fize, J., L. Moncla, et B. Martins (2021). Deep learning for toponym resolution : Geocoding based on pairs of toponyms. *ISPRS International Journal of Geo-Information 10*(12), 818.

Gaio, M. et L. Moncla (2019). 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*, 354–386.

Gritta, M., M. T. Pilehvar, N. Limsopatham, et N. Collier (2018). What's missing in geographical parsing ? *Language Resources and Evaluation 52*(2), 603–623.

21. https://universaldependencies.org

Halilali, M. S., E. Gouardères, M. Gaio, et F. Devin (2022). Geospatial web services discovery through semantic annotation of wps. *ISPRS International Journal of Geo-Information 11*(4), 254.

Hu, X., Z. Zhou, H. Li, Y. Hu, F. Gu, J. Kersten, H. Fan, et F. Klan (2022). Location reference recognition from texts : A survey and comparison. *arXiv preprint arXiv :2207.01683*.

Jones, C. B. et R. S. Purves (2008). Geographical information retrieval. *International Journal of Geographical Information Science 22*(3), 219–228.

Moncla, L. et M. Gaio (2015). A multi-layer markup language for geospatial semantic annotations. In *Proceedings of the 9th Workshop on Geographic Information Retrieval*.

Moncla, L. et M. Gaio (2018). Services web pour l'annotation sémantique d'information spatiale à partir de corpus textuels. *Revue Internationale de Géomatique 28*(4), 439–459.

Moncla, L., M. Gaio, T. Joliveau, Y.-F. Le Lay, N. Boeglin, et P.-O. Mazagol (2019). Mapping urban fingerprints of odonyms automatically extracted from french novels. *International Journal of Geographical Information Science 33*(12), 2477–2497.

Moncla, L., M. Gaio, J. Nogueras-Iso, et S. Mustière (2016). Reconstruction of itineraries from annotated text with an informed spanning tree algorithm. *International Journal of Geographical Information Science 30*(6), 1137–1160.

Moncla, L., W. Renteria-Agualimpia, J. Nogueras-Iso, et M. Gaio (2014). 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, pp. 183–192.

Moncla, L., D. Vigier, K. Mcdonough, A. Brenon, et T. Joliveau (2021). Combinaison d'approches qualitative et quantitative pour le repérage et la classification des entités nommées dans l'encyclopédie de diderot et d'alembert (1751-1772). In *Theoretical linguistics in the light of the interaction of qualitative and quantitative approaches*.

Nouvel, D., M. Ehrmann, et S. Rosset (2015). *Les entités nommées pour le traitement automatique des langues*. ISTE Group.

Vigier, D., L. Moncla, A. Brenon, K. Mcdonough, et T. Joliveau (2020). 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*.

Summary

This article presents the Perdido Python library for geoparsing and geocoding of French texts. We present the general architecture of the Perdido tool composed of three layers: back-office, API and Python library. We detail the methods used for the development of the processing chain and the different tasks (named entity recognition and classification and toponym resolution). Finally, we present the different functionalities of the Python library and how to use it. The library is developed as an overlay using the API services and allows to manipulate, visualize and export geoparsing and geocoding results. A Jupyter notebook describes, in the form of a tutorial, all the features implemented in the library.