Rapport de stage

Extraction d'entités nommées géographiques et exploitation dans le cadre du Web des données

Étudiant : Arbona Arnaud

Formation : Master 2 Informatique Fondamentale et Ingénierie

Parcours : Ingénierie Spécialité Ingénieur en Web et IA

Encadrante de stage : Madame Anne Toulet, Cirad

Tutrice enseignante : Madame Catherine Faron, Polytech Nice-Sophia

Table des matières

Remerciements

Je tiens à remercier toutes les personnes qui ont participé au bon déroulement de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

En premier lieu, je tiens à remercier mon encadrante de stage, Madame Anne Toulet pour son accueil et ses conseils au cours de ce stage. Je remercie également Monsieur Franck Michel de l'équipe Wimmics et Monsieur Andon Tchechmedjiev de l'IMT Mines Alès pour leur aide et leur appui.

Ce travail a été réalisé dans le cadre du projet Issa soutenu par le GIS Collex-Persée (https://www.collexpersee.eu/projet/issa/)

1. Introduction

1.1 Contexte et sujet du stage

La science ouverte est un mouvement international qui a pour but de rendre accessible la recherche scientifique et les données qu'elle produit. Dans cet objectif, les archives ouvertes, bases de données documentaires accessibles librement et gratuitement sur internet contenant des documents issus de la recherche scientifique, accentuent leurs efforts pour accroître l'accessibilité aux ressources dont elles disposent.

Ce stage s'inscrit dans le cadre du projet Issa (Indexation Sémantique d'une archive scientifique et Services Associés pour la science ouverte), lauréat de l'appel à projet CollEx-Persée 2019/2020.
L'objectif de ce stage est de permettre un accès et une interopérabilité accrus à des publications scientifiques proposées par une archive ouverte en adoptant des techniques d'indexation sémantique adossée à des référentiels terminologiques standards. Le recours aux techniques du Web sémantique et du traitement naturel des langues sera privilégié.

Dans ce travail, nous nous intéresserons en particulier à la question de l'indexation[1] par des mots-clés géographiques : extraction d'entités nommées[2] géographiques, alignement avec des référentiels sémantiques standards et visualisation cartographique.

Tout au long de ce projet, l'archive ouverte du cirad, Agritrop (https://agritrop.cirad.fr/), servira de cas d'usage.

Agritrop est un portail qui propose essentiellement des publications scientifiques, mais aussi un fonds de cartes et de documents anciens. Chaque document est décrit par des métadonnées riches parmi lesquelles se trouvent des descripteurs thématiques et géographiques issus du thésaurus[3] Agrovoc[4].

Figure 1 : Exemple d'une notice d'un article dans Agritrop

[1] L'indexation permet de préciser le contenu d'un document à travers des mots-clés et ainsi de retrouver dans un catalogue tous les documents qui traitent d'un sujet donné quel que soit le support.
[2] Concept d'intérêt dans un document donné
[3] Un thésaurus est un lexique structuré de mots-clés permettant l'analyse et le classement de documents
[4] Thésaurus développé et maintenu par la FAO depuis le début des années 80 (http://aims.fao.org/fr/agrovoc)

Le projet ISSA est composé de trois partenaires institutionnels qui travaillent en collaboration :

- Le Cirad (Centre de coopération internationale en recherche agronomique pour le développement), qui est un organisme français de recherche agronomique et de coopération internationale pour le développement durable des régions tropicales et méditerranéennes.
- IMT Mines Alès (Coordinateur Andon Tchechmedjiev).
- L'équipe Wimmics, qui est une équipe de recherche conjointe entre l'INRIA Sophia Antipolis–Méditerranée et I3S – CNRS et Université côte d'Azur – (Coordinateur Franck Michel).

Nous allons maintenant présenter le contenu du rapport.

1.2 Plan du rapport

Dans les chapitres suivants, nous commencerons par une présentation détaillée du sujet de stage.
Puis nous expliquerons le travail réalisé pour atteindre les objectifs explicités dans la partie précédente :

- État de l'art des référentiels géographiques
- Conception et réalisation d'une chaîne de traitement
- Analyse des résultats obtenus
- Enrichissement des données et visualisation cartographique

Dans la conclusion, nous reviendrons sur l'expérience acquise au cours de ce stage, en particulier les domaines scientifiques sur lesquelles j'ai pu me perfectionner ainsi que sur les connaissances et compétences que j'ai pu améliorer.

2. Description du travail proposé

2.1 Présentation du projet Issa

Comme indiqué dans l'introduction, mon stage s'inscrit dans le projet Issa qui se décompose en plusieurs work packages.

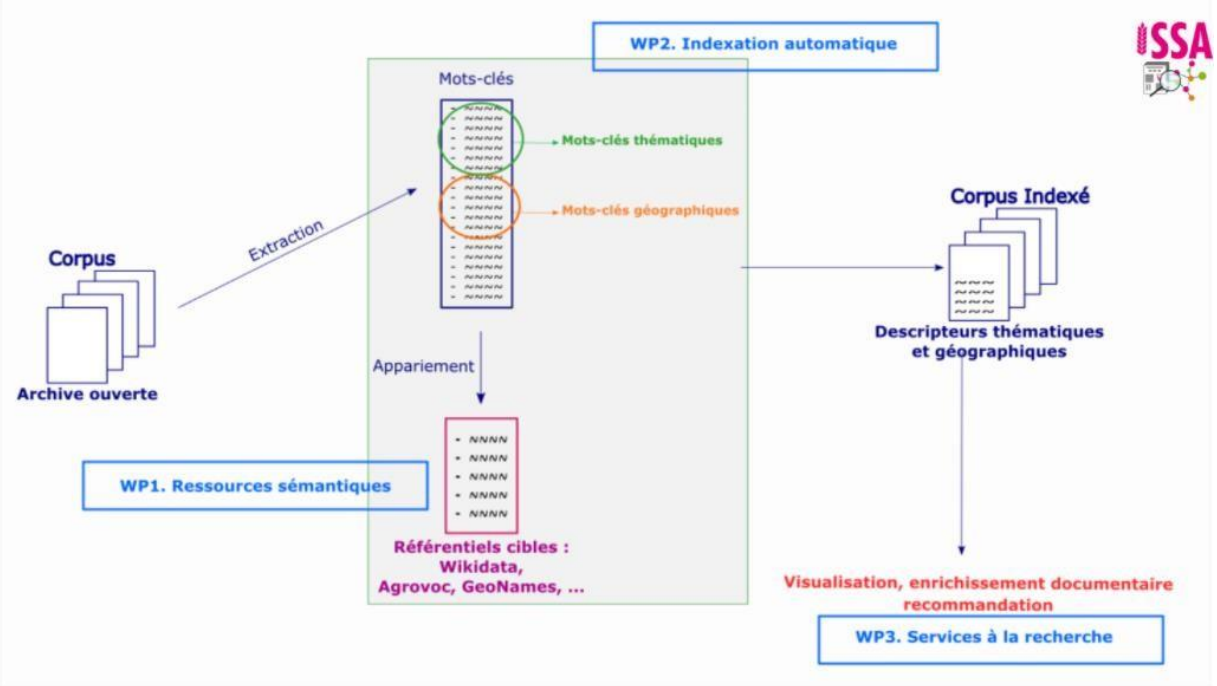


Figure 2 : Décomposition du projet issa

Le WP1 concerne les ressources sémantiques d'intérêt pour le projet dans le domaine de l'agronomie et la géographie. Cette étude sera basée sur les besoins d'indexation dans Agritrop et proposera des pointeurs vers des référentiels terminologiques adaptés à d'autres domaines pour permettre de transférer la démarche à d'autres communautés.

Le WP2 a pour but de concevoir une chaîne de traitement[5] générique permettant l'indexation automatique d'un corpus scientifique. Il faudra tout d'abord effectuer une extraction structurée complète des textes (traitement d'un pdf pour obtenir un texte structuré exploitable par les machines) puis un liage des entités nommées. Pour cela nous utiliserons des techniques de reconnaissance et désambiguïsation d'entités nommées.

Le WP3 a pour objectif d'exploiter les descripteurs sémantiques en proposant différents services aux utilisateurs, comme une visualisation enrichie. Cela sera possible grâce aux descripteurs globaux ainsi que les entités nommées obtenues à la sortie du WP2.

Dans le paragraphe suivant, je détaillerai plus précisément les tâches à mener dans le cadre de mon stage.

[5] Série de processus permettant d'extraire de l'information ou produire du savoir à partir de données brutes de manière automatisée

2.2 Descriptif du stage

La problématique de mon sujet de stage se concentre sur l'extraction d'entité nommées géographiques et leur visualisation cartographique.

Actuellement, l'indexation des documents par mots-clés géographiques est effectuée manuellement par des documentalistes.
L'objectif principal est de trouver comment automatiser cette procédure et enrichir les informations avec de la visualisation cartographique.

Pour atteindre cet objectif, plusieurs étapes seront nécessaires : 1) choisir un référentiel géographique pertinent pour le projet ; 2) extraire les entités nommées géographiques des documents et les lier avec le référentiel cible ; 3) proposer un service de visualisation cartographique de ces entités nommées.

Dans ce cadre-là, les tâches attendues sont les suivantes :

1. États de l'art
   - Dresser un inventaire détaillé des référentiels géographiques existants en comparant leurs différentes approches. On s'intéressera particulièrement à GeoNames.
   - Étudier et comparer les différents outils d'extraction d'entités nommées existants, en particulier pour les entités géographiques.

2. Conception et réalisation d'une chaîne de traitement

   - Développer et appliquer une chaîne d'extraction et de désambiguïsation des entités nommées
   - Liage des entités nommées extraites avec le référentiel géographique choisi sur la base de l'état de l'art effectué précédemment,

Cette chaîne de traitement sera testée sur un jeu de publications scientifiques issues d'Agritrop.

3. Exploitation de l'indexation sémantique

Exploiter les descripteurs géographiques obtenus en utilisant les technologies du Web sémantique et le Web de données, par exemple en permettant la visualisation géographique (cartographie) et/ou l'enrichissement encyclopédique à partir des mots-clés géographiques obtenus précédemment.

Il sera nécessaire de mettre en application des règles de bonne pratique de développement et d'industrialisation d'une application. Il s'agit de délivrer un produit fini, facilement déployable et configurable dans d'autres environnements.

3. Description du travail réalisé

3.1 Planning du stage

Mois	Tâche
Avril	Étude de la bibliographie + état de l'art
Mai	État de l'art + tests outils d'extraction
Juin	Chaîne de traitement
Juillet	Chaîne de traitement et visualisation cartographique
Août	Documentation + amélioration de la chaîne de traitement et de la visualisation

3.2 Tâches réalisées

3.2.1 État de l'art des référentiels géographiques

Tel qu'exposé précédemment, le premier objectif de mon stage est de réaliser un inventaire des référentiels géographiques.

Tout d'abord, sur la partie concernant l'état de l'art des référentiels géographiques, il a fallu déterminer quels sont les critères de sélection qui permettront de choisir un référentiel adapté aux besoins du projet. Les principales caractéristiques recherchées sont les suivantes :

- Présence des coordonnées géographiques : latitude, longitude
- Disponibilité au format du web sémantique
- Ressource en libre d'accès
- Accessible via une API Rest / Sparql endpoint
- Couverture géographique mondiale

Ci-dessous vous trouverez un tableau récapitulatif de l'analyse effectuée en fonction de ces critères.

Les liens vers ces référentiels sont indiqués dans la partie « Liens utiles » à la fin de ce document.

Nom	Développeur	Objectif Projet	Dernière date de mise à jour	Licence	API	Type Donnée	Projet utilisant ces référentiels
GeoNames	Wiki + équipe d'aide aux développement	géographique	2021	Libre de partager et de modifier Préciser que l'on utilise GeoNames	oui	Json xml/rdf	Multimap microsoft Popfly
LinkedGeoData OpenStreetMap	AKWS (Agile Knowledge Engineering and Semantic Web)	géographique	2018	Amalgame de plusieurs licences	oui	rdfs	Pas mentionné
Getty Thesaurus of Geographic Names	Plusieurs équipes (musée, parc nationaux)	historique	2018	Open Data Commons Attribution License v1.0	oui	xml/rdf	Beaucoup de projet ayant pour thème l'histoire
GeoEthno	CNRS, Université Paris Nanterre	Science social	2020	Libre de partager Préciser le que l'on utilise GeoEthno, pas d'utilisation commerciale, pas de modification	Isidore (mais ce n'est pas une API)	rdf skos	Isidore

Dans ce tableau sont présentés uniquement 4 référentiels géographiques. Beaucoup d'autres ont été étudiés mais écartés majoritairement en raison de leur couverture géographique insuffisante.

Le premier référentiel présenté, GeoNames est une base de données géographiques consultable sur internet qui contient plus de 25millions de noms géographiques et se compose de plus de 11 millions de noms uniques. Ce référentiel est de type wiki avec une équipe qui vérifie les données (pour le mode premium). Plusieurs informations concernant les lieux géographiques sont présentes, comme les coordonnées géographiques ainsi que les différents noms que peuvent porter certaines régions, les données sont mises à jour régulièrement. En ce qui concerne la licence d'utilisation, elle est libre d'accès, il faut uniquement préciser que l'on utilise GeoNames. En ce qui concerne l'accessibilité du référentiel via une API Rest cela est possible, cependant il n'y a pas de Sparql endpoint. Nous pouvons néanmoins télécharger un dump[6] de GeoNames au format rdf. Ce référentiel est utilisé par de nombreuses entreprises comme Microsoft ou Popfly.

Le second référentiel étudié est LinkedGeoData OpenStreetMap. Ce référentiel est un amalgame de plusieurs projets dont GeoNames, il est développé par AKWS (Agile Knowledge Engineering and semantic Web). En ce qui concerne la licence d'utilisation il faut vérifier chaque projet dont est composé ce référentiel. Le projet n'étant plus mis à jour depuis 2018, certaines données peuvent ne plus être exactes. Une API et un Sparql endpoint existent, mais au moment de la rédaction de ce document les services en ligne sont inutilisables. Le format de données utilisé est le rdfs et aucun projet utilisant ce référentiel n'est mentionné.

Getty Thesaurus of Geographic Names est un référentiel différent des deux précédents, car l'objectif de ce référentiel est plutôt axé sur l'histoire. Cela a un certain intérêt, car de nombreuses villes au cours de l'histoire ont changé de nom ce qui peut être un problème lors de l'analyse de documents. Ce référentiel a été développé par plusieurs musées et parc nationaux. Les données n'ont pas été mises à jour depuis 2018. En ce qui concerne la licence les données sont en accès libre et nous pouvons réutiliser ce projet. Une API est disponible ainsi qu'un Sparql endpoint. Le format des données, correspond à nos besoins et de nombreux projets ont utilisé ce référentiel.

Le dernier référentiel étudié est GeoEthno. Il s'agit d'un projet développé par le CNRS et l'université Paris Nanterre. La thématique de ce référentiel n'est pas orientée géographie, mais science sociale ce qui ne correspond pas vraiment à nos besoins. L'utilisation de ce référentiel est différente des trois autres car il faut utiliser un autre projet (Isidore), de plus lorsqu'on utilise Isidore pour interagir avec Geoethno nous avons accès à GeoNames. Là encore ce référentiel est un amalgame de plusieurs référentiels et les conditions d'utilisation ne correspondent pas à nos besoins.

En conclusion, parmi les critères recherchés, le référentiel géographique répondant le plus à nos besoins est GeoNames. Il possède une couverture mondiale, les ressources sont libres d'accès, de nombreuses données géographiques sont présentes, les données sont disponibles au format du web sémantique. Néanmoins pour pouvoir utiliser les données au format rdf il nous faut télécharger le dump de GeoNames.

[6] Copie du contenu de GeoNames sur un fichier

3.2.2 Conception et mise en place de la chaîne de traitement

a) Choix d'un outil d'extraction d'entités nommées

Après avoir analysé et sélectionné le référentiel géographique le plus adapté à notre projet, il m'a fallu étudier les différents outils d'extraction d'entités nommées. Là encore il a fallu chercher des critères d'évaluation afin de choisir le meilleur outil.

Les critères sont les suivants :

- Outil Open source
- Multilingue Anglais, Français
- Indication de la Position des entités dans le texte
- Désambiguïsation

Les deux outils testés sont Blink et entity-fishing.

Afin de pouvoir tester ces outils il m'a fallu mettre en place une machine virtuelle pour me permettre d'installer Blink et entity-fishing.

Le premier outil que j'ai testé est Blink, qui est une librairie python de liage d'entités avec Wikipédia. Blink est développé par Facebook research, il utilise une approche basée sur des architectures BERT (c'est un réseau de neurones qui permet de traiter une grande variété de problèmes du traitement automatique de la langue).
Le projet est open source sous licence MIT.
Au travers des différents tests effectué avec cet outil, les résultats sur des textes anglais sont concluants. Toutes les entités nommées sont reconnues et sont liées à un id Wikipédia.

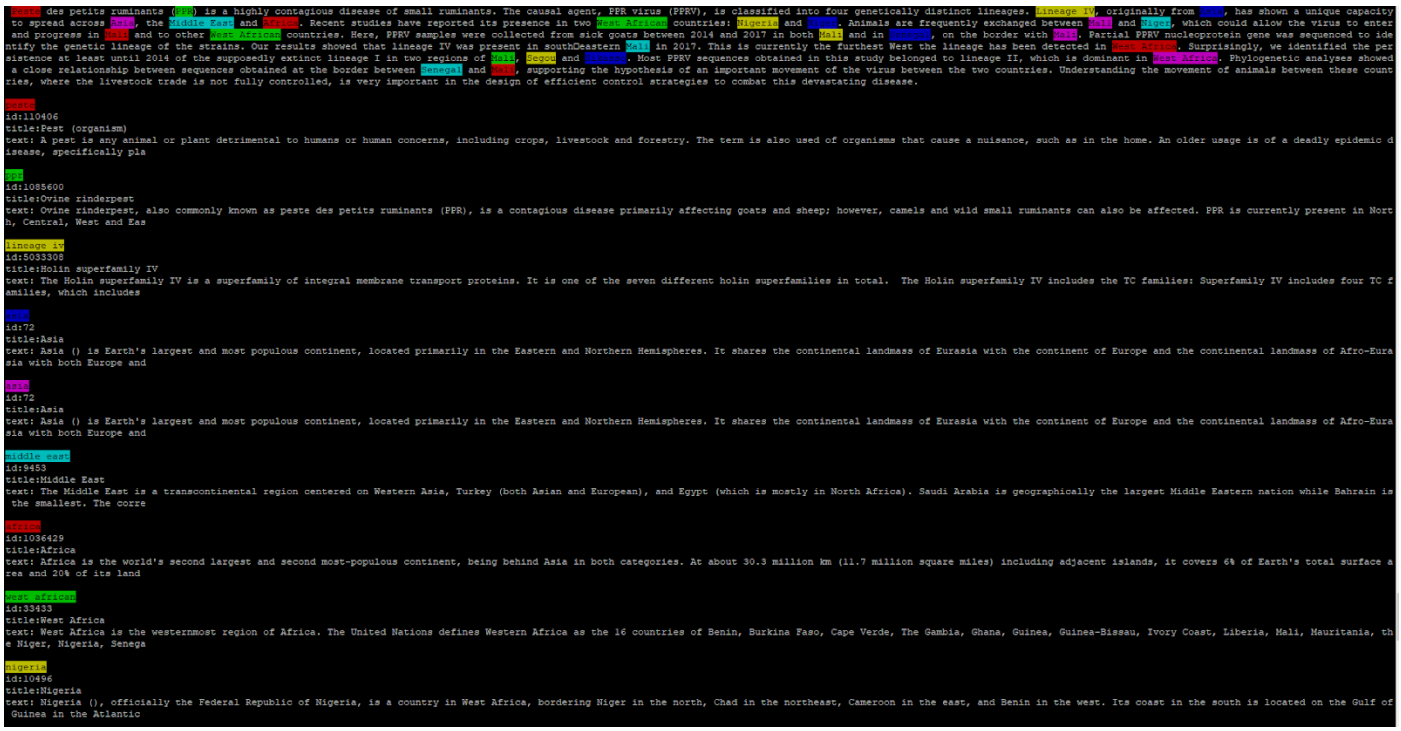


Figure 3 : Résultat Blink avec un texte issu d'agritrop

Le problème majeur de Blink est que la méthode BERT utilisée pour faire fonctionner cet outil a besoin d'un apprentissage en fonction de la langue pour permettre au réseau de neurones de désambiguïser le mieux possible. Cela a pour conséquence de ne pouvoir traiter qu'un seul type de langue alors que les textes fournis par Agritrop sont majoritairement en anglais ou français.
On s'est donc tourné vers un second outil, entity-fishing, développé par la société science-miner qui est aussi la société qui a conçu Grobid, un des logiciels utilisés pour transformer des pdf en document TEI ou txt.
Ce logiciel est open source sous licence Apache 2.
La première différence entre entity-fishing et Blink est la manière dont est fait le liage avec les entités nommées. Blink utilise Wikipedia alors que entity-fishing utilise à la fois Wikipedia, mais aussi Wikidata. De plus, lorsque celui-ci est dans l'incapacité d'effectuer le liage, un « Type » de classe est renvoyé par Grobid ner (name entity recognition). Grobid ner possède 27 types dont un type Location pour les entités géographiques.

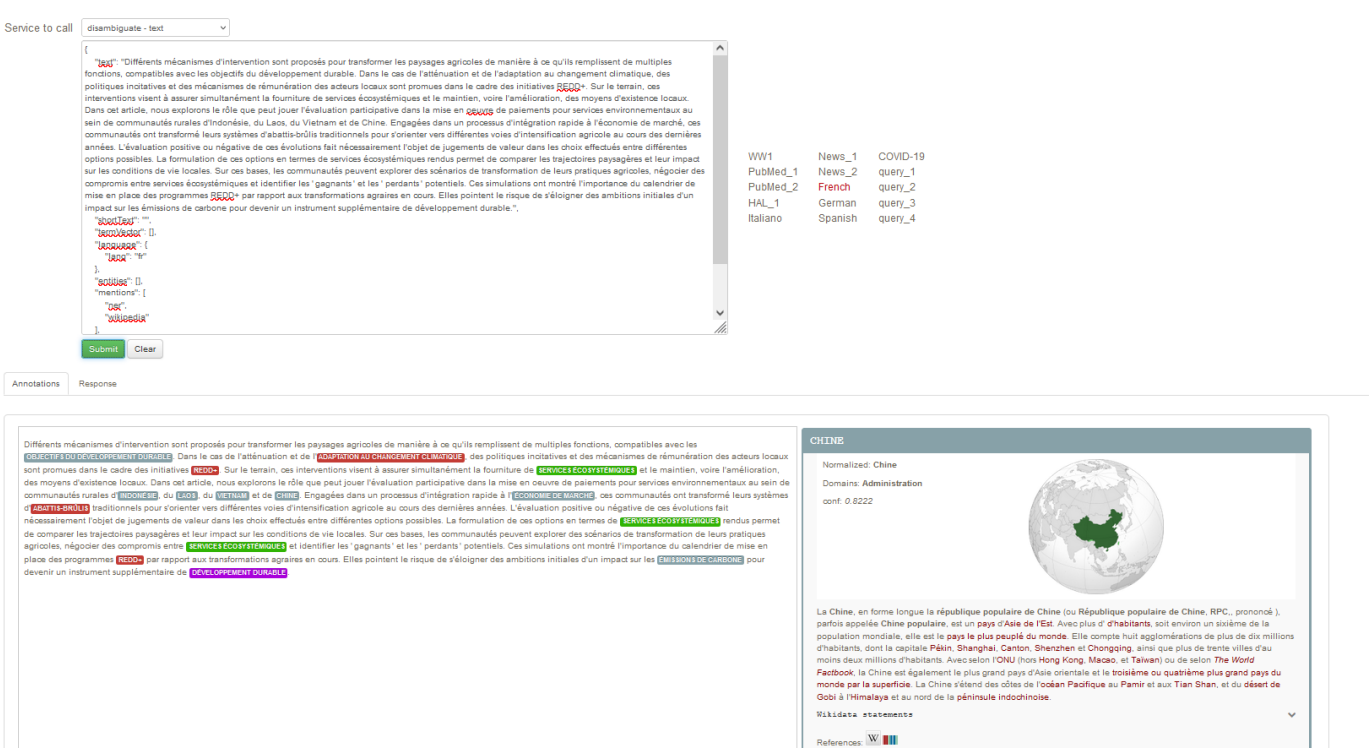


Figure 4: Résultat d'entity-fising avec un texte issu d'Agritrop

L'outil fonctionne avec plusieurs langues (Français, anglais, italien, allemand, espagnol), il n'utilise pas la méthode BERT pour fonctionner, ce qui est un avantage, car il n'y aura pas besoin de réentraîner un modèle à chaque changement de langue.
Entity-fishing possède un service en ligne qui permet de faire des tests avant de l'installer sur la machine virtuelle du projet.
Cela m'a permis d'analyser les premiers résultats et de les comparer à ceux obtenus avec Blink.
Entity-fishing renvoie l'ensemble des entités nommées sous forme d'un json avec les positions dans le texte de chaque concept.

Cet outil répondant à tous nos critères de sélection nous avons décidé de l'utiliser pour la suite du développement.

b) Extraction des entités nommées géographiques et alignement avec GeoNames

En sortie d'entity-fishing, on récupère une liste d'entités nommées de tout type : entités nommées généralistes et géographiques. L'un des objectifs est d'isoler les entités géographiques des autres.

Pour pouvoir réaliser cette chaîne de traitement j'ai travaillé en plusieurs étapes :

- Travail technique : récupérer et traiter le fichier json en sortie d'entity-fishing.
- Isoler et traiter les entités nommées géographiques pour les aligner avec GeoNames.

Travail technique

Pour récupérer et traiter le fichier json obtenu en sortie d'entity-fishing, il existe une librairie utilisable en python. Cette librairie développée par Hirmeos avec l'aide de l'équipe de science-miner et sous licence Apache 2.0 m'a permis de communiquer avec entity-fishing installé sur la machine virtuelle.
J'ai utilisé putty afin d'établir la connexion entre mon ordinateur et la machine virtuelle où est installé entity-fishing. Pour fonctionner, la librairie avait seulement besoin d'un point d'accès vers le port de la machine virtuelle associé à entity-fishing.

À la suite de cet ajustement technique, nous avons décidé de construire la chaîne de traitement en python afin de pouvoir utiliser cette librairie mais aussi car python est très utilisé pour effectuer du traitement automatique du langage

Traitement des entités nommées géographiques

Le résultat en sortie de entity-fishing est au format json et est composé de trois parties :

- Des informations concernant la version de l'outil : le texte envoyé, la date et heure du run ainsi que la langue associée au texte
- Une partie avec des « global_categories » associées à Wikipedia
- Une partie contenant toutes les entités nommées


```
{
    "software": "entity-fishing",
    "version": "0.0.4",
    "date": "2021-08-21T13:47:28.743Z",
    "runtime": 1754,
    "nbest": false,
    "text": "Edmond Ludlow (vers 1617-1692) est un parlementaire anglais, plus connu pour son implication dans l'exécution de Charles Ier, et pour ses mémoires, publiés à titre posthume et qui sont devenus une source
importante pour les historiens des Guerres des Trois Royaumes. Après avoir servi dans les guerres civiles anglaises, Ludlow a été élu membre du Long Parlement. Après la création du Commonwealth en 1649, il est nommé
adjoint de Ireton, commandant des forces du Parlement en Irlande, avant de rompre avec Oliver Cromwell lors de la création du Protectorat. Après la Restauration, Ludlow part en exil en Suisse, où il passe une grande
partie du reste de sa vie.",
    "language": {
        "lang": "fr",
        "conf": 0
    },
    "global_categories": [
        {
            "weight": 0.007751937984496138,
            "source": "wikipedia-fr",
            "category": "Langue de Saint-Christophe-et-Niévès",
            "page_id": 553481
        },
        {
            "weight": 0.007751937984496138,
            "source": "wikipedia-fr",
            "category": "Langue de Papouasie-Nouvelle-Guinée",
            "page_id": 547084
        },
        {
            "weight": 0.007751937984496138,
            "source": "wikipedia-fr",
            "category": "Langue de Sainte-Lucie",
            "page_id": 553484
        }
    ],
    "entities": [
        {
            "rawName": "Edmond Ludlow",
            "offsetStart": 0,
            "offsetEnd": 13,
            "confidence_score": 0.9921,
            "wikipediaExternalRef": 8331104,
            "wikidataId": "Q53399650",
            "domains": [
                "Administration"
```


Figure 5 : Exemple de réponse au format json de entity-fishing

Plus précisément, voici les informations d'importance pour nous :

- Au début du fichier, seule la langue nous intéresse pour pouvoir différentier correctement les textes anglais et français.
- Les « global categories » ne nous intéressent pas, car dans cette tâche on se concentre uniquement sur les entités nommées géographiques et celles-ci ne comportent aucune information utile de ce genre.
- La partie « entities » est la plus importante : elle contient toutes les entités nommées (géographiques et autres) ainsi que divers renseignements.

On peut séparer ces entités en deux parties, celles qui possèdent un id Wikidata et Wikipédia et celles qui possèdent seulement un type sans id.
Les entités nommées qui possèdent les deux id sont celles qui ont pu être liées avec Wikipedia.
Les entités qui possèdent un type NER sont celles que entity-fishing n'a pas réussi à désambiguïser. Ce type est obtenu grâce à grobid tel qu'expliqué précédemment. Dans notre cas on s'intéressera uniquement à celles qui ont le type « LOCATION »


```
{
    "rawName": "Irlande",
    "type": "LOCATION",
    "offsetStart": 477,
    "offsetEnd": 484,
    "confidence_score": 0
},
```


Figure 6 : Exemple d'une entité nommée géographique non désambiguïsée

Chaque entité correctement désambiguïsée est composée de :

- Son « rawName »
- De sa position dans le texte (début et fin)
- D'un score de confiance par rapport à la désambiguïsation, confidence_score
- De deux id (Wikidata, Wikipedia)
- D'un ou plusieurs « domains »


```
{
    "rawName": "services écosystémiques",
    "offsetStart": 493,
    "offsetEnd": 516,
    "confidence_score": 0.9948,
    "wikipediaExternalRef": 858238,
    "wikidataId": "Q295865",
    "domains": [
        "Environment",
        "Hydraulics",
        "Plants"
    ]
},
```


Figure 7 : Exemple d'une entité nommée désambiguïsée

En sortie de traitement j'obtiens donc une liste d'entités nommées :

- Soit désambiguïsée avec des id Wikipedia, Wikidata
- Soit non désambiguïsée avec un type NER

Travail préalable technique : reformater les résultats pour pouvoir les exploiter

- Création d'un programme python qui prend en entrée un répertoire contenant des fichiers texte avec les résumés de quatre documents issus d'Agritrop et qui renvoie en sortie un autre répertoire qui va contenir les quatre fichiers json.

Ce programme python contient plusieurs fonctions :

- Une fonction read qui prend en entrée un répertoire et un nom de fichier qui va me permettre de lire chaque fichier txt (en utf-8)
- Une fonction pour éliminer les parties inutiles et garder seulement la métadonnée sur la langue du texte et toutes les entités
- Une fonction d'écriture pour écrire le résultat dans un fichier json dans le répertoire de sortie en gardant un encoding utf-8 et en gardant le même nom de fichier que celui des txt, car chaque fichier txt se nomme avec id Agritrop afin de pouvoir repérer plus facilement le document concerné.
- Deux fonctions en lien avec la librairie python d'entity-fishing afin de communiquer avec l'outil
- Un programme principal « main » dans lequel on va parcourir chaque fichier du répertoire d'entrée pour obtenir un résultat dans le répertoire de sortie.

À ce stade, on a donc récupéré toutes les entités nommées et reformaté le résultat.

Problématique : Comment isoler les entités nommées géographiques des autres et les aligner avec GeoNames ?

Une première idée était d'utiliser les « domains » associés à chaque entité afin de répertorier celles qui sont géographiques, mais cela était impossible, car les « domains » n'étaient pas pertinents. Du coup, cette partie a été retirée du résultat.
Les seuls éléments pouvant nous aider restaient les id Wikidata et Wikipedia. En ce qui concerne les entités avec un type location, elles seront traitées ultérieurement.

Pour chaque EN géographique trouvée, l'id Wikipedia renvoie sur une page qui ne contient pas de lien vers GeoNames.
En revanche, on peut trouver un pointeur vers GeoNames dans Wikidata, ce qui me permet de lier les EN extraites avec GeoNames.

Travail technique pour séparer les EN géographiques des autres et les lier avec GeoNames.

Pour cela j'ai créé un second programme python afin d'effectuer des requêtes sparql sur le sparql endpoint de Wikidata de manière à récupérer les id GeoNames grâce aux id Wikidata.
La requête sparql pour interroger Wikidata n'est pas intégrée au code python, mais séparée dans un fichier .txt (bonne pratique).
Afin de pouvoir récupérer le fichier txt contenant ma requête sparql, j'utilise la librairie string puis pour interroger Wikidata, la libraire SPARQLWrapper.

Ce programme python contient les fonctions suivantes :

- Une fonction read_json pour lire le json obtenu à l'étape précédente
- Une fonction read_txt pour lire la requête Sparql
- Une fonction write afin d'écrire le résultat dans un nouveau json
- Une fonction Get_value_id_GeoNames_wrapper afin d'effectuer la requête à Wikidata
- Une fonction Traitement_Wikidata afin de récupérer le résultat de la requête sparql et de récupérer uniquement l'id GeoNames
- Un programme principal « main » pour exécuter l'ensemble.

La chaîne de traitement est volontairement divisée en deux programmes afin de pouvoir analyser chaque étape et de pouvoir effectuer des analyses sur les résultats.

Lorsqu'un id Wikidata possède un id GeoNames, on l'associe avec l'entité nommée auquel il appartient. Il est ensuite gardé et écrit dans un nouveau json situé dans un nouveau répertoire. Sinon il est supprimé de la liste des entités. En ce qui concerne les entités géographiques avec un type LOCATION, elles sont gardées dans le json sans modification.

Une fois le programme terminé, nous obtenons un json avec uniquement des entités géographiques, soit associées à un id GeoNames, soit à un type LOCATION.

Après avoir optimisé certaines parties, j'ai continué à explorer diverses solutions pour trouver un moyen d'arriver à lier les entités qui possèdent seulement un type LOCATION à une base de connaissance.

Pour pouvoir résoudre ce problème, on renvoie sans contexte de phrase les entités géographiques dans entity-fishing pour les désambiguïser avec Wikidata.

Cependant il y a deux inconvénients :

- Le premier est que le « confidence score » à moins de valeur car l'EN est désambiguïsée sans contexte
- Le second est que le risque de faux positifs est augmenté, pour cela un score de confiance d'au moins 10% est requis sinon l'entité géographique est éliminée. même si l'outil la déclare comme entité géographique

c) Analyse des résultats

Les résultats que je vais maintenant exposer ont été obtenus après traitement de deux articles scientifiques issus de l'archive ouverte Agritrop.

Le premier est un article intitulé « Persistence of the historical lineage I of West Africa against the ongoing spread of the Asian lineage of peste des petits ruminants virus » écrit en langue anglaise (id dans Agritrop : 598198).
Cet article possède 36 entités géographiques distinctes identifiées manuellement par un documentaliste et chacune d'entre elles peut être répétée plusieurs fois dans le texte (124 fois avec la répétition).
Sur ces 36 entités distinctes, le programme en trouve 26/36 avec deux entités qui sont des faux positifs, soit un pourcentage de réussite de 63%.

Ce pourcentage de réussite est à nuancer, car certaines des entités géographiques présentes dans le texte ne possèdent pas de page Wikipedia ni Wikidata (il s'agit de petits villages de l'ouest de l'Afrique). Par conséquent il est impossible d'arriver à désambiguïser ou à lier ces entités avec entity-fishing. Si l'on s'en tient aux entités nommées (EN) qui ont un id Wikipédia, alors le taux de réussite est meilleur.

Par exemple, dans cet article, sept EN ne peuvent être trouvées avec la chaîne de traitement car elles ne sont pas répertoriées dans Wikipédia. En excluant ces sept EN, on obtient alors un pourcentage de réussite de 77%.
Les 3 EN typées LOCATION n'ont pas été retenu car non répertoriées dans GeoNames.

Dans le tableau ci-dessous, on peut voir les EN de la publication avec dans la première colonne le label d'EN, dans la seconde le nombre de répétions et dans la troisième colonne une note explicative.

Africa	2 sur 2	
Asia	1 sur 1	
Bamako	1 sur 1	
Benin	2 sur 2	
Burkina Faso	3 sur 3	
Central, East, North and West Africa	0 sur 1	impossible car entité géographique composé
Dialan	0 sur 2	n'existe pas dans la base de donnée de wikidata ni geonames
East	1 fois	Est de l'afrique mais problème de désambiguisation
East Africa	2 sur 2	
France	0 sur 1	Problème même lors de la désambiguisation dans le pdf
Ghana	1 sur 1	
Ivory Coast	1 sur 1	
Kayes	3 sur 2	
Kayes region	3 sur 4	
Kedougou	3 sur 3	
Kenieba	1 sur 1	
Kolondieba	1 sur 1	
Kopropin	0 sur 1	n'existe pas dans la base de donnée de wikidata ni geonames
Krounikoto	0 sur 1	n'existe pas dans la base de donnée de wikidata ni geonames
Mali	29 sur 29	
Montpellier	0 sur 1	Problème même lors de la désambiguisation dans le pdf
Mauritania	1 sur 1	
Middle East	1 sur 1	
Mopti	3 sur 4	
Niger	11 sur 11	
Nigeria	7 sur7	
Samako	0 sur 1	n'existe pas dans la base de donnée de wikidata ni geonames
Segou	5 sur 5	
Senegal	14 sur 14	
Seroume	0 sur 1	n'existe pas dans la base de donnée de wikidata ni geonames
Sikasso	1 sur 1	
Sitakili	0 sur 1	Problème dans le fichier txt car reconnu dans le pdf
Tambacounda	1 sur 1	
united kingdom	1 sur 1	
west	1 fois	ouest de l'afrique mais problème de désambiguisation
West and East Africa	0 sur 1	impossible car entité géographique composé
West Africa	8 sur 8	
West african	6 sur 6	associe le même id que west africa
Article 598198		

Figure 8 : Analyse des résultats pour l'article 598198

À partir de l'analyse des résultats présentés précédemment, nous pouvons calculer les scores de précision[7] et de rappel[8].

598198	Entités nommées présent	Entités nommées absent
Réponse du test positive	Vrai positif : 111	Faux positif : 3
Réponse du test négative	Faux négatif : 10	Vrai négatif : /

- Précision : 111 / (111 + 3) = 97%
- Rappel : 111 / (111 + 10) = 91%

[7] La précision donne le pourcentage de réponses correctes.
[8] Le rappel donne le pourcentage des réponses correctes qui sont données.

Le deuxième est un article intitulé « Explorer l'impact environnemental des transformations agraires en Asie du Sud-Est grâce à l'évaluation participative des services écosystémiques » écrit en français (id dans Agritrop : 597393).
Cet article possède 22 entités géographiques distinctes identifiées manuellement par un documentaliste et chacune d'entre elles peut être répétée plusieurs fois dans le texte (52 fois avec la répétition).
Sur ces 22 entités distinctes, le programme en trouve 13/22 soit un pourcentage de réussite de 59%.
Ce pourcentage de réussite et aussi à nuancer car certaines entités géographiques ne sont pas présentes dans nos différents référentiels.
3 entités géographiques ne peuvent être trouvé, alors le taux de réussite est de 68%.
6 entités était typé LOCATION n'ont pas été retenu car non répertoriées dans GeoNames.

Dans le tableau ci-dessous, on peut voir les EN de la publication avec dans la première colonne le label d'EN, dans la seconde le nombre de répétions et dans la troisième colonne une note explicative.

Afrique	1 sur 1	
Amérique Latine	1 sur 1	
Asie	1 sur 1	
Asie du Sud-Est	2 sur 3	
Chine	7 sur 8	
Chinois	3 sur 3	
district de Con Cuong	0 sur 1	Problème même lors de la désambiguisation dans le pdf
district de Hiem	0 sur 1	n'existe pas dans la base de donnée de wikidata ni geonames
district de Kutai Barat	0 sur 1	Problème même lors de la désambiguisation dans le pdf
Huaphan	1 sur 1	
Indénosie	3 sur 3	
indonésiens	0 sur 3	Problème même lors de la désambiguisation dans le pdf
Kalimantan	0 sur 1	Problème dans le fichier txt car reconnu dans le pdf
Kalimantan Est	0 sur 1	Problème dans le fichier mais ne trouver pas kalimantan Est, l'outil faire le lien avec Kalimantan
Laos	7 sur 7	
laotiens	0 sur 2	Problème même lors de la désambiguisation dans le pdf
préfecture de Xishuangbanna	0 sur 1	Problème dans le fichier txt car reconnu dans le pdf
province de Nghe An	1 sur 1	
province du Yunnan	1 sur 1	
région chinoise du Xishuangbanna	0 sur 1	Problème dans le fichier txt car reconnu dans le pdf
thaïlande	1 sur 1	
Vietnam	9 sur 9	
Article 597393		

Figure 9 : Analyse des résultats pour l'article 597393

À partir de l'analyse des résultats présentés précédemment, nous pouvons à nouveau calculer les scores de précision et de rappel pour ce second article.

- Précision : 38 / (38 + 0) = 100%
- Rappel : 38 / (38 + 14) = 73%

5597393	Entités nommées présent	Entités nommées absent
Réponse du test positive	Vrai positif : 38	Faux positif : 0
Réponse du test négative	Faux négatif : 14	Vrai négatif : /

Le score de rappel obtenu précédemment pour les documents ne prennent pas en compte le fait que certaines entités nommées géographiques ne possèdent pas d'id dans Wikipedia, Wikidata et GeoNames.
Si l'on refait les calculs pour les 2 documents en enlevant en enlevant ces entités nommées là on obtient :

Pour le premier document, nous avons 119 entités nommées géographique en enlevant celles cité précédemment.

598198	Entités nommées présente	Entités nommées absente
Réponse du test positive	Vrai positif : 111	Faux positif : 3
Réponse du test négative	Faux négatif : 5	Vrai négatif : /

- Rappel : 111 / (111 + 5) = 95%

Et pour le second document, nous avons 46 entités nommées géographiques en enlevant celle cité précédemment.

599793	Entités nommées présente	Entités nommées absente
Réponse du test positive	Vrai positif : 38	Faux positif : 0
Réponse du test négative	Faux négatif : 8	Vrai négatif : /

- Rappel : 38 / (38 + 8) = 82%

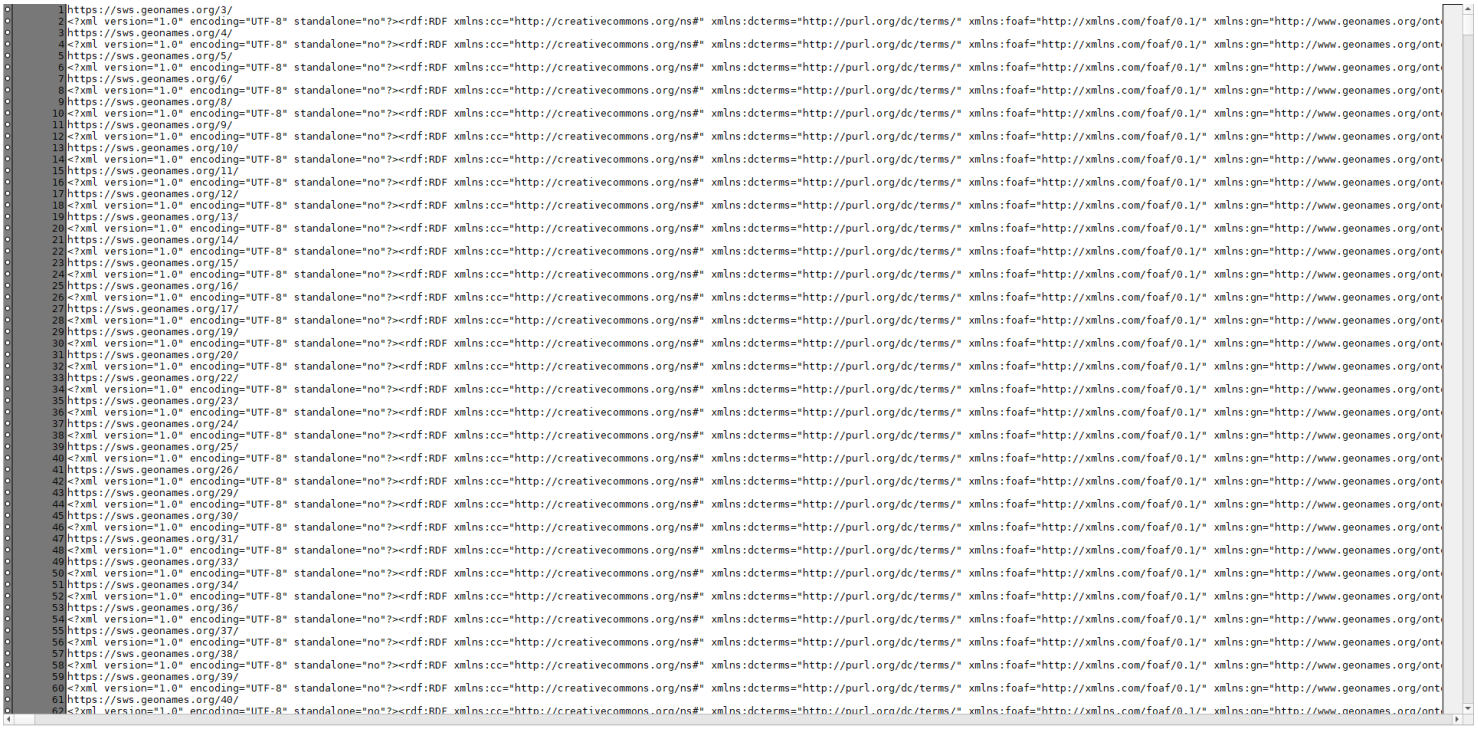
En conclusion, sur les 2 documents analysés nous obtenons un score de précision de plus de 95% nous permettant de voir que le nombre de faux positifs est extrêmement bas.
En ce qui concerne le score de rappel, nous avons aussi de très bons résultats, dans les 2 cas supérieurs à 70 %.
Lorsqu'on enlève les entités nommées qui ne se trouvent dans aucune des bases de connaissance (Wikipédia, Wikidata, GeoNames), le score de rappel est supérieur à 80% soit une augmentation de 10%.
La chaîne de traitement mise en place obtient un score de plus de 75% de réussite dans le meilleur des cas.
Le texte en anglais possède un meilleur score que celui en français, probablement parce que les outils utilisés fonctionnent mieux en langue anglaise.

3.2.3 Enrichissement des données et visualisation cartographique

Dans cette section, on explique les procédés mis en place pour visualiser les EN géographiques sur une carte.

Pour pouvoir enrichir les données obtenues en sortie de la chaîne de traitement, on utilise le dump de GeoNames. Le premier problème rencontré est que ce dump n'est pas un fichier rdf valide (voir figure ci-dessous). Il s'agit d'une concaténation de plusieurs milliards de fichier rdf.

Ce fichier contient des éléments associés à l'id GeoNames comme les coordonnées géographiques, les différents noms d'un même endroit, s'il s'agit d'un contient, d'un pays ou d'une ville.



```
1 https://sws.geonames.org/3/
2 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
3 https://sws.geonames.org/4/
4 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
5 https://sws.geonames.org/5/
6 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
7 https://sws.geonames.org/6/
8 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
9 https://sws.geonames.org/8/
10 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
11 https://sws.geonames.org/9/
12 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
13 https://sws.geonames.org/10/
14 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
15 https://sws.geonames.org/11/
16 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
17 https://sws.geonames.org/12/
18 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
19 https://sws.geonames.org/13/
20 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
21 https://sws.geonames.org/14/
22 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
23 https://sws.geonames.org/15/
24 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
25 https://sws.geonames.org/16/
26 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
27 https://sws.geonames.org/17/
28 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
29 https://sws.geonames.org/19/
30 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
31 https://sws.geonames.org/20/
32 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
33 https://sws.geonames.org/22/
34 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
35 https://sws.geonames.org/23/
36 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
37 https://sws.geonames.org/24/
38 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
39 https://sws.geonames.org/25/
40 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
41 https://sws.geonames.org/26/
42 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
43 https://sws.geonames.org/29/
44 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
45 https://sws.geonames.org/30/
46 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
47 https://sws.geonames.org/31/
48 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
49 https://sws.geonames.org/33/
50 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
51 https://sws.geonames.org/34/
52 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
53 https://sws.geonames.org/36/
54 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
55 https://sws.geonames.org/37/
56 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
57 https://sws.geonames.org/38/
58 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
59 https://sws.geonames.org/39/
60 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
61 https://sws.geonames.org/40/
62 <?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ont
```


Figure 10 : Dump de GeoNames avant mise en conformité

Travail technique

Pour résoudre ce problème, j'ai créé un troisième fichier python permettant de modifier le fichier rdf et de le rendre conforme. Pour cela, j'ai effectué des traitements en utilisant les expressions régulières.
Dans la figure ci-dessous on montre le résultat obtenu.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <rdf:RDF xmlns:cc="http://creativecommons.org/ns#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:gn="http://www.geonames.org/ontology#" xmlns:owl="http://www.w3.org/2002/07/owl#" xm
3     <gn:Feature rdf:about="https://sws.geonames.org/3/">
4         <rdfs:isDefinedBy rdf:resource="https://sws.geonames.org/3/about.rdf"/>
5         <gn:name>Zamīn Sūkhteh</gn:name>
6         <gn:featureClass rdf:resource="https://www.geonames.org/ontology#S"/>
7         <gn:featureCode rdf:resource="https://www.geonames.org/ontology#S.CRRL"/>
8         <gn:countryCode>IR</gn:countryCode>
9         <wgs84_pos:lat>32.45831</wgs84_pos:lat>
10        <wgs84_pos:long>48.96335</wgs84_pos:long>
11        <gn:parentFeature rdf:resource="https://sws.geonames.org/3202991/"/>
12        <gn:parentCountry rdf:resource="https://sws.geonames.org/130758/"/>
13        <gn:parentADM1 rdf:resource="https://sws.geonames.org/127082/"/>
14        <gn:nearbyFeatures rdf:resource="https://sws.geonames.org/3/nearby.rdf"/>
15        <gn:locationMap rdf:resource="https://www.geonames.org/3/zamin-sukhteh.html"/>
16    </gn:Feature>
17
18    <gn:Feature rdf:about="https://sws.geonames.org/4/">
19        <rdfs:isDefinedBy rdf:resource="https://sws.geonames.org/4/about.rdf"/>
20        <gn:name>Rūdkhāneh-ye Āb-e Zālek</gn:name>
21        <gn:alternateName>Rūdkhāneh-ye Āb-e Zālek</gn:alternateName>
22        <gn:alternateName>Rūdkhāneh-ye Āb-e Zālekī</gn:alternateName>
23        <gn:featureClass rdf:resource="https://www.geonames.org/ontology#H"/>
24        <gn:featureCode rdf:resource="https://www.geonames.org/ontology#H.STM"/>
25        <gn:countryCode>IR</gn:countryCode>
26        <wgs84_pos:lat>32.93273</wgs84_pos:lat>
27        <wgs84_pos:long>48.76505</wgs84_pos:long>
28        <gn:parentFeature rdf:resource="https://sws.geonames.org/127082/"/>
29        <gn:parentCountry rdf:resource="https://sws.geonames.org/130758/"/>
30        <gn:parentADM1 rdf:resource="https://sws.geonames.org/127082/"/>
31        <gn:nearbyFeatures rdf:resource="https://sws.geonames.org/4/nearby.rdf"/>
32        <gn:locationMap rdf:resource="https://www.geonames.org/4/rudkhaneh-ye-ab-e-zalek.html"/>
33    </gn:Feature>
34
35    <gn:Feature rdf:about="https://sws.geonames.org/5/">
36        <rdfs:isDefinedBy rdf:resource="https://sws.geonames.org/5/about.rdf"/>
37        <gn:name>Yekāhī</gn:name>
38        <gn:featureClass rdf:resource="https://www.geonames.org/ontology#P"/>
39        <gn:featureCode rdf:resource="https://www.geonames.org/ontology#P.PPL"/>
40        <gn:countryCode>IR</gn:countryCode>
41        <wgs84_pos:lat>32.5</wgs84_pos:lat>
42        <wgs84_pos:long>48.9</wgs84_pos:long>
43        <gn:parentFeature rdf:resource="https://sws.geonames.org/127082/"/>
44        <gn:parentCountry rdf:resource="https://sws.geonames.org/130758/"/>
45        <gn:parentADM1 rdf:resource="https://sws.geonames.org/127082/"/>
46        <gn:nearbyFeatures rdf:resource="https://sws.geonames.org/5/nearby.rdf"/>
47        <gn:locationMap rdf:resource="https://www.geonames.org/5/yekahi.html"/>
48    </gn:Feature>
49
50    <gn:Feature rdf:about="https://sws.geonames.org/6/">
51        <rdfs:isDefinedBy rdf:resource="https://sws.geonames.org/6/about.rdf"/>
52        <gn:name>Āb-e Yasi</gn:name>
53        <gn:featureClass rdf:resource="https://www.geonames.org/ontology#H"/>
54        <gn:featureCode rdf:resource="https://www.geonames.org/ontology#H.STM"/>
55        <gn:countryCode>IR</gn:countryCode>
56        <wgs84_pos:lat>32.8</wgs84_pos:lat>
57        <wgs84_pos:long>48.8</wgs84_pos:long>
58        <gn:parentFeature rdf:resource="https://sws.geonames.org/127082/"/>
59        <gn:parentCountry rdf:resource="https://sws.geonames.org/130758/"/>
60        <gn:parentADM1 rdf:resource="https://sws.geonames.org/127082/"/>
61        <gn:nearbyFeatures rdf:resource="https://sws.geonames.org/6/nearby.rdf"/>
62        <gn:locationMap rdf:resource="https://www.geonames.org/6/ab-e-yasi.html"/>
```

Figure 11 : Dump de GeoNames après mise en conformité

À ce stade, on possède deux sources de données à exploiter : le fichier json obtenu après les traitements en sortie de entity-fishing et le fichier reformaté du dump GeoNames. Dans la section suivante on va expliquer comment lier les deux et exploiter leurs informations.

Travail technique

Un triplestore Virtuoso a été déployé sur la VM du projet de manière à : stocker les données et déployer un Sparql endpoint permettant de les interroger .

Une fois l'installation terminée, j'ai pu commencer à créer une requête sparql afin de pouvoir récupérer l'ensemble des données en fonction d'un id GeoNames.
Ces données (surtout les coordonnées géographiques) vont me permettre de créer une visualisation cartographique pour situer correctement toutes les entités géographiques d'un document.

Pour réussir cette visualisation, je me suis appuyé sur mes connaissances acquises lors de mon premier semestre : librairie React javascript, requêtes Sparql.

Travail technique

J'ai développé mon application en react en utilisant la librairie react leaflet et en utilisant une carte open source de openstreetmap.
Pour pouvoir appeler mes fichiers json récupérés en sortie de la chaîne de traitement, j'ai créé un serveur web dans un docker et je les place sur le serveur.

Lorsque mon application react s'exécute, je choisis quel fichier json va être lu. En fonction des id GeoNames présent dans mon json, j'exécute une requête sparql vers le Virtuoso afin de récupérer plusieurs informations comme les coordonnées géographiques.

Outre la visualisation cartographique recherchée, les traitements précédents permettent d'enrichir les données à partir des informations obtenues dans Wikipédia, Wikidata et GeoNames

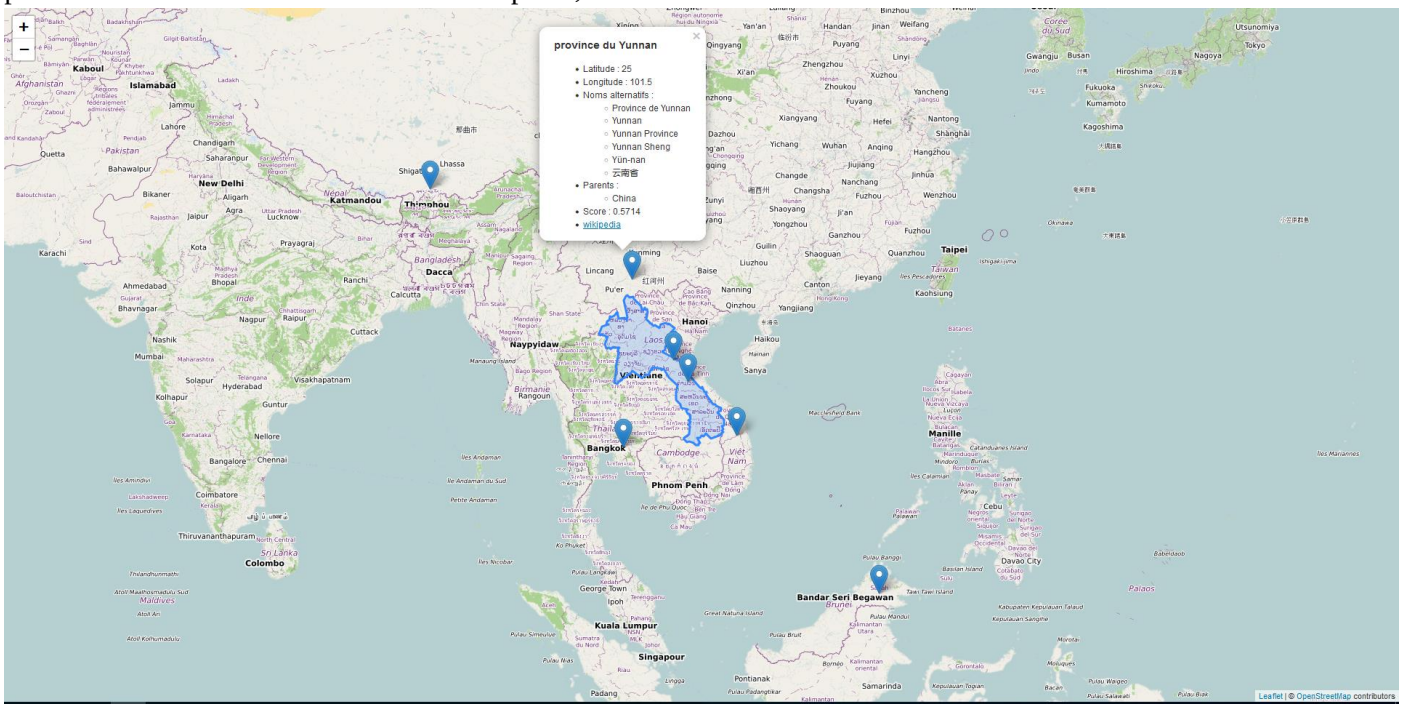


Figure 12 : Visualisation cartographique avec bulle d'information

Dans la bulle d'information, nous pouvons donc trouver un lien vers la page Wikipedia de l'entité géographique pour obtenir plus d'informations.
Dans cet exemple, on peut remarquer que le pays du Laos ne possède pas un marqueur simple mais bien la délimitation de tout le pays (colorié en bleu).
Cela est possible grâce à des fichiers GeoJson qui possèdent un id GeoNames associé au contour d'un pays.
Ces fichiers GeoJSON sont obtenus sur le site de GeoNames.
Pour obtenir ce contour pour des villes, cela n'a pas été possible dans le cadre de ce projet qui utilise prioritairement des données et outils libres et gratuits. Or ce type de données est payant (1500€ par an sur le site de GeoNames).

Conclusion et perspectives

Les objectifs de ce stage étaient de réaliser une chaîne de traitement permettant d'extraire d'un article scientifique les entités nommées géographiques, de les aligner avec un référentiel standard (ici, GeoNames) puis de les visualiser sur une carte.

Ces objectifs font appel à des connaissances en Web sémantique et en traitement automatique des langues. Cela m'a permis de renforcer mes compétences scientifiques et techniques dans ces domaines. J'ai pu aussi mettre en application les bonnes pratiques de développement. Ce dernier point était d'importance pour le projet : le code développé doit pouvoir être réutilisé et intégré à la chaine de traitement globale.

J'ai été confronté à différents problèmes tout au long de ce stage, à la fois techniques et conceptuels. J'ai appris à les analyser pour trouver des solutions avec l'aide de mes encadrants.
Par ailleurs, j'ai énormément renforcé mes compétences transversales : prise de recul par rapport au sujet, synthèse et rédaction d'un document, relations/communication avec les membres d'un projet.

Si je devais poursuivre mon travail sur ce projet, il y a au moins deux points que j'aimerais approfondir et améliorer :

- Tester ma chaîne de traitement sur un plus grand nombre de textes pour avoir des résultats plus pertinents.
- Trouver une solution (gratuite) pour les fichiers GEOJson afin d'avoir un id GeoNames associé à un contour de ville, pays ou de région.

Dans tous les cas, l'expérience acquise au cours de ce stage me servira grandement dans mes futurs emplois.

Bibliographie

Compréhension générale du sujet

[1] Arnaud, J. Cataloguer, rechercher des cartes. Le référencement géographique en question. https://www.cairn.info/revue-documentaliste-sciences-de-l-information-2014-3-page-68.htm#

[2] Ghislain Auguste Atemezing, Raphaël Troncy. Vers une meilleure interopérabilité des données géographiques françaises sur le Web de données. ».

https://hal.archives-ouvertes.fr/hal-00716149v1/

[3] Pascal Cuxac, Alain Collignon, Stéphanie Gegoro, François Parmentier, Des bases de données massives au Web de données : désambiguïsation et alignement d'entitésgéographiques dans les textes scientifiques.
https://hal.archives-ouvertes.fr/hal-02307577

[4]Geographic Ontologies: Survey and ChallengesRobert Laurini (Lyon, France) & Okba Kazar (Biskra,Algeria) https://perso.liris.cnrs.fr/robert.laurini/text/Kazar.pdf

[5]W3C Geospatial ontologies https://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/

Liens utiles

Index geographiques (Gazetteers)

- **GeoNames** : http://www.GeoNames.org/
  Pour avoir des infos sur l'API : https://www.programmableweb.com/api/GeoNames
- **OpenStreetMap & LinkedGeoData**
  La base de données associée à OpenStreetMap se trouve sur linkedgeodata.org : "LinkedGeoData is an effort to add a spatial dimension to the Web of Data / Semantic Web. LinkedGeoData uses the information collected by the OpenStreetMap project and makes it available as an RDF knowledge base according to the Linked Data principles. It interlinks this data with other knowledge bases in the Linking Open Data initiative." Visualisation : http://browser.linkedgeodata.org/
- **Getty Thesaurus of Geographic Names**
  https://www.getty.edu/research/tools/vocabularies/tgn/
- **GeoEthno**, un thésaurus géographique pour l'ethnologie

http://www.mae.u-paris10.fr/dbtw-wpd/bed/index-lesc.html

Entity linking avec Wikidata

http://www.semantic-web-journal.net/system/files/swj2670.pdf

Documentation et outils

Entity-fishing : https://nerd.readthedocs.io/en/latest/index.html

GROBID Named Entity Recognition Documentation : https://grobid-ner.readthedocs.io/en/latest/Voir aussi : https://grobid.s3.amazonaws.com/presentations/29-10-2017.pdf

GitHub entity-fishing : https://github.com/kermitt2/entity-fishing
Service en ligne entity-fishing (API REST) : http://nerd.huma-num.fr/nerd/
Client Python: pip install git+https://github.com/issa-project/entity-fishing-client-python.git

BLINK: https://github.com/facebookresearch/BLINK
Babelfy: http://babelfy.org

Docker Virtuso :
https://hub.docker.com/r/tenforce/virtuoso/

React Leaflet :
https://react-leaflet.js.org/

GeoJSON :
https://en.Wikipedia.org/wiki/GeoJSON

Projet Issa :
https://issa.cirad.fr/

Table des figures

Annexes

Config.ini

```
[ENTITY_FISHING]
DOSSIER_ENTREE = C:\Users\Luci\Desktop\Test\txt
DOSSIER_SORTIE = C:\Users\Luci\Desktop\Test\Not_propre
MIN_CONFIDENCE_SCORE = 0.1
API_BASE = http://localhost:5500/service/

[REQUETE]
DOSSIER_ENTREE = C:\Users\Luci\Desktop\Test\Not_propre
DOSSIER_SORTIE = C:\Users\Luci\Desktop\Test\Propre
```

Config.ini qui permet de modifier directement les paramètres importants du code

Entity-fishing.py

```
from nerd import nerd_client
import json
import os
import os.path
import sys
import re
import configparser

config = configparser.ConfigParser()
config.read('config.ini')

#url api
api_base = config.get('ENTITY_FISHING', 'API_BASE')
min_confidence_score = float(config.get('ENTITY_FISHING', 'MIN_CONFIDENCE_SCORE'))
```

1 : Import + config parser

```
#Takes a directory and a file as input and reads the contents of the file
def read(rep, file_name):
    result= None
    with open(os.path.join(rep,file_name),'r',encoding = 'utf-8') as file:
        result = file.read()

    return result
```

2 : Fonction de lecture d'un fichier texte

```python
#Depending on the result of the wikidata query and if the type of the entity is LOCATION, keep only the geographical entities and add the Id_geonames
def Traitement_Type_LOCATION(json_,lang):
    result = []

    for data in json_:
        print('        ', data['rawName'])
        if 'wikidataId' in data:
            result.append(data)
        elif data['type'] == 'LOCATION':
            short = entity_fishing_short(data['rawName'],lang)[0]
            if short != None and 'entities' in short:
                data.pop('type', None)
                data.pop('confidence_score', None)
                data['confidence_score'] = short['entities'][0]['confidence_score']
                if 'wikipediaExternalRef' in short['entities'][0] :
                    data['wikipediaExternalRef'] = short['entities'][0]['wikipediaExternalRef']
                if 'wikidataId' in short['entities'][0] :
                    data['wikidataId'] = short['entities'][0]['wikidataId']
                result.append(data)

    return result
```

3 : Traitement des entités nommées avec un type LOCATION

```python
#Conforming the json to keep only geographical entities and relevant information. Modifies json display for start and end of words compared to entity-fishing for duplicates
def clean_json(json_):
    result = []
    entities_only = json_[0]['entities']
    for data in entities_only:
        if not any(obj['rawName'] == data['rawName'] for obj in result):
            data.pop('domains', None)
            data['text_location'] = []
            data['text_location'].append({'offsetStart':data['offsetStart'], 'offsetEnd':data['offsetEnd']})
            del data['offsetStart']
            del data['offsetEnd']
            result.append(data)
        else :
            next(obj for obj in result if obj['rawName'] == data['rawName'])['text_location'].append({'offsetStart':data['offsetStart'], 'offsetEnd':data['offsetEnd']})
    return result
```

4 : Suppression des données non importante + modification pour les doublons + sélection des entités nommées géographique (avec id et avec type Location)

```python
#Take a directory, a file and write the data contained in the payload
def write(rep, file_name, payload):
    with open(os.path.join(rep, file_name), 'w', encoding = 'utf-8') as file:
        json.dump(payload,file, indent=4, ensure_ascii=False)
```

5 : Fonction pour écrire dans un fichier

```python
#allows you to connect to entity-fishing and disambiguate a text
def entity_fishing(t,lang):
    client = nerd_client.NerdClient(api_base)
    test = client.disambiguate_text(t, language=lang)

    return test
```

6 : Fonction qui permet de se connecter à entity-fishing

```python
#Allows you to retrieve the language of a text
def entity_fishing_get_language(t):
    client = nerd_client.NerdClient(api_base)
    test = client.get_language(t)

    return test[0]['lang']
```

7 : Permet de récupérer le langage du texte

```
#Allows you to send a single word to disambiguate and link without context, the function needs to know the language of the word to allow disambiguation and linking correctly
def entity_fishing_short(t,lang):
    client = nerd_client.NerdClient(api_base)
    test = client.disambiguate_query(t, language=lang)

    return test
```

8 : Permet d'envoyer une entité nommée sans contexte à entity-fishing

```
def main(argv):
    dossier_entree = config.get('ENTITY_FISHING', 'DOSSIER_ENTREE')
    dossier_sortie = config.get('ENTITY_FISHING', 'DOSSIER_SORTIE')
    if not os.path.isdir(dossier_entree):
        print("Créer le dossier d'entrée car il n'existe pas encore")
        return
    if not os.path.isdir(dossier_sortie):
        os.mkdir(dossier_sortie)


    for file_name in os.listdir(dossier_entree):
        print(file_name)
        txt = read(dossier_entree, file_name)
        lang = entity_fishing_get_language(txt)
        json_ = entity_fishing(txt,lang)
        json_ = clean_json(json_)
        json_ = Traitement_Type_LOCATION(json_,lang)
        json_ = list(filter(lambda x : x['confidence_score'] >= min_confidence_score, json_))
        print(file_name)
        transform = re.sub(r'.txt$', '.json', file_name)
        json_= {'lang' : lang, 'entities' : json_}
        write(dossier_sortie, transform, json_)

if __name__ == "__main__":
    main(sys.argv[1:])
```

9 : Le main qui applique toutes les fonctions précédentes + applique le min_confidence_score + vérifie la l'existence des répertoires

Requête.py

```
import json
import os
import os.path
import sys
from string import Template
from SPARQLWrapper import SPARQLWrapper, JSON
from time import sleep
import configparser

config = configparser.ConfigParser()
config.read('config.ini')

#url to make sparql queries on wikidata
url = 'https://query.wikidata.org/sparql'
```

1 : Import + config parser

```python
#Takes a directory and a file as input and reads the contents of the file
def read_text(rep, file_name):
    result= None
    with open(os.path.join(rep,file_name),'r',encoding = 'utf-8') as file:
        result = file.read()

    return result
```

2 : Fonction de lecture d'un fichier texte

```python
#Takes a directory and a json file as input, reads the contents of the file
def read_json(rep, file_name):
    result= None
    with open(os.path.join(rep,file_name),'r',encoding = 'utf-8') as file:
        result = json.load(file)

    return result
```

3 : Fonction de lecture d'un fichier json

```python
#Take a directory, a file and write the data contained in the payload
def write(rep, file_name, payload):
    with open(os.path.join(rep, file_name), 'w',encoding = 'utf-8') as file:
        json.dump(payload,file, indent=4, ensure_ascii=False)
```

4 : Fonction qui permet d'écrire dans un fichier

```python
#Makes a query to wikidata. The query is in the txt file "Requete_sparql_wikidata".
def Get_value_id_geonames_wrapper(n, query):
    q = query.substitute(valeur = n)
    sparql = SPARQLWrapper(url)
    sparql.setReturnFormat(JSON)

    sparql.setQuery(q)

    attempt = 0
    MAX_ATTEMPTS = 5

    while attempt <= MAX_ATTEMPTS:
        try:
            ret = sparql.query().convert()
            return ret['results']['bindings'][0]['id_geonames']['value'] if len(ret['results']['bindings']) > 0 else None
        except:
            if attempt < MAX_ATTEMPTS:
                sleep(0.5)
        finally:
            attempt = attempt + 1
    print("            Abbort")
    return None
```

5 : Fonction pour envoyer une requête a Wikidata et récupérer le résultat

```python
#Depending on the result of the wikidata query, keep only the geographical entities and add the Id_geonames
def Traitement_wikidata(json_, query):
    result = []

    for data in json_:
        print('      ', data['rawName'])
        if 'wikidataId' in data:
            wikidata_id = Get_value_id_geonames_wrapper(data['wikidataId'], query)
            if wikidata_id != None:
                data['Id_geonames'] = wikidata_id
                result.append(data)

    return result
```

6 : Fonction pour garder uniquement les entités nommées géographiques et rajouter l'id GeoNames obtenue

```python
def main(argv):
    dossier_entree = config.get('REQUETE', 'DOSSIER_ENTREE')
    dossier_sortie = config.get('REQUETE', 'DOSSIER_SORTIE')
    query_wikidata= Template(read_text('.', 'Requete_sparql_wikidata.txt'))

    if not os.path.isdir(dossier_entree):
        print("Créer le dossier d'entrée car il n'existe pas encore")
        return
    if not os.path.isdir(dossier_sortie):
        os.mkdir(dossier_sortie)

    for file_name in os.listdir(dossier_entree):
        print(file_name)
        json_ = read_json(dossier_entree, file_name)
        print(' Traitement_wikidata')
        json_['entities'] = Traitement_wikidata(json_['entities'],query_wikidata)
        json_['entities'] = sorted(json_['entities'], key = lambda x : x['confidence_score'], reverse = True)
        print(' Ecriture')
        write(dossier_sortie, file_name, json_)




if __name__ == "__main__":
    main(sys.argv[1:])
```

7 : Le main qui applique toutes les fonctions précédentes + applique le min_confidence_score + vérifie la l'existence des répertoires

Mise_en_confimité_GeoNames_rdf.py

```
import os.path
import os
import re

#Nettoyage d'une ligne
def Clean(l) :
    #Remplace les espaces entre les balises par un saut de ligne, puis les espaces
    l = re.sub("\>( *)\<", r">\n\1<", l)
    #Retire les alternanteName, officialName et shortName  hors fr et en
    l = re.sub("\<gn:(alternateName|officialName|shortName) xml:lang=\"(?!fr|en).*\"\>.*\</gn:(alternateName|officialName|shortName)\>", "", l)
    #Retire les lignes vides
    l = re.sub("\n\s*\n", "\n", l, flags=re.MULTILINE)
        #Retine balise rdf fermante
    l = re.sub("\</rdf:RDF\>", "", l)
    return l

with open("all-geonames-rdf.txt", "r", encoding='utf-8') as r :
    with open("all-geonames-rdf-propre.txt", "w", True, encoding='utf-8') as w :

        #Première ligne ignorée
        r.readline()

        #Deuxième ligne
        l = r.readline()
        l = Clean(l)
        w.write(l)

        passe = True
        #Reste du fichier
        for line in r:

            #Saute une ligne sur 2 (les urls)
            if(not passe):
                #On retire la balise xml
                line = re.sub("\<\?xml version=\"1\.0\" encoding=\"UTF-8\" standalone=\"no\"\?\>\<rdf\:RDF xmlns\:cc=\"http\://creativecommons\.o
                line = Clean(line)
                w.write(line)

            passe = not passe

        w.write("</rdf:RDF>")
```

Mise en conformité du dumps GeoNames

Requête Sparql

```
SELECT ?id_geonames
WHERE
        {
    wd:$valeur wdt:P1566 ?id_geonames
    }
```

Requête pour récupérer les id GeoNames a partie du Sparql endpoint de Wikidata

```
prefix gn: <http://www.geonames.org/ontology#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>

select ?name ?officialName ?latitude ?longitude ?altitude ?alternateName ?nameParentFeature ?nameParentCountry
where
{
  <https://sws.geonames.org/1655842/> gn:name ?name;
                                      wgs84_pos:lat ?latitude;
                                      wgs84_pos:long ?longitude;
                                      gn:featureCode ?type.
  OPTIONAL { <https://sws.geonames.org/1655842/> gn:officialName ?officialName}.
  OPTIONAL { <https://sws.geonames.org/1655842/> wgs84_pos:alt ?altitude}.
  OPTIONAL { <https://sws.geonames.org/1655842/> gn:alternateName ?alternateName}.
  OPTIONAL { <https://sws.geonames.org/1655842/> gn:parentCountry ?parentCountry.
             ?parentCountry gn:name ?nameParentCountry}
   OPTIONAL { <https://sws.geonames.org/1655842/> gn:parentFeature ?parentFeature.
             ?parentFeature gn:name ?nameParentFeature}.
}
```

Requête sparql pour extraire les informations du dump GeoNames dans virtuoso

Résumé

Les objectifs de ce stage étaient de réaliser une chaîne de traitement permettant d'extraire d'un article scientifique les entités nommées géographiques, de les aligner avec un référentiel standard (ici, GeoNames) puis de les visualiser sur une carte.

Les tâches réalisées font appel à des connaissances en Web sémantique et en traitement automatique des langues.

Dans ce rapport, nous expliquons comment nous avons conçu et développé une chaîne de traitement permettant d'atteindre ces objectifs : état de l'art des référentiels géographiques, analyse d'outil d'extraction d'entités nommées et visualisation cartographique.

Mots clés : Web sémantique, Web de données, extraction d'entités nommées, traitement automatique des langues, indexation, descripteurs géographiques.

Abstract

The objectives of this internship were to create a processing chain allowing to extract named geographical entities from a scientific article, to align them with a standard reference system (here, GeoNames) and then to visualize them on a map.

The tasks performed call for knowledge in semantic web and automatic language processing.

In this report, we explain how we designed and developed a processing chain to achieve these objectives: state of the art of geographic repositories, analysis of named entity extraction tools and map visualisation.

Keywords: Semantic Web, Web of data, named feature extraction, automatic language processing, indexing, geographic descriptors indexing, geographical descriptors