Conception d'une méthode hybride d'extraction d'informations géographiques à partir de données textuelles

Christopher Gizard

Encadré par Ludovic Moncla (INSA Lyon) et
Thierry Joliveau (Université Jean Monnet, Saint-Etienne)

Université Claude Bernard Lyon 1, France

Résumé Notre projet consiste à apporter des améliorations à une plateforme capable d'extraire et d'analyser les occurrences de noms de lieux à partir de romans publiés entre 1800 et 1914. Nous décrirons plusieurs méthodes pour désambiguïser ces noms de lieux. Puis nous étudierons différents systèmes de reconnaissances d'entités nommées (NER) et entraînerons des modèles à reconnaître automatiquement les noms de lieux. Ce rapport décrit les résultats obtenus pour la désambiguïsation et la reconnaissance automatique d'entités nommées étendues de lieux.
Mots-clés: Geoparsing, Reconnaissance d'entités nommées, Geocoding, Désambiguïsation, Entité Nommée Etendue

Abstract. Our project involves making improvements to a platform able to extract and analyze the occurences of place names from fictional novels published between 1800 and 1914. We will describe several methods for disambiguating these place names. Then, we will study different NER systems and will train models to automatically recognize place names. This report describes the results obtained for disambiguation and automatic recognition of extended spatial named entities.

Keywords: Geoparsing, Named Entity Recognition, Toponym resolution, Extended Named Entity

1 Introduction

Ce projet s'inscrit dans le cadre d'une collaboration pluridisciplinaire (informatique et géomatique) ayant comme problématique principale l'extraction d'informations géographiques pour l'analyse et la représentation cartographique de documents textuels. L'objectif de ce projet est de proposer une nouvelle méthode pour modéliser et extraire de manière automatique des informations géographiques issues de données textuelles non structurées afin de les intégrer dans des systèmes d'information utilisant des modèles de raisonnement.

Pour répondre à cette problématique, nous nous appuierons sur les travaux antérieurs de M. Moncla et M. Joliveau [9] afin d'enrichir les fonctionnalités de web services existants au sein de le plateforme Perdido.

Le travail a été décomposé en deux tâches principales. La première tâche concerne le géocodage des informations géographiques, c'est-à-dire, le processus qui permet d'associer des coordonnées GPS à un nom de lieu. La deuxième tâche, s'intéresse à l'expérimentation et l'évaluation de différentes méthodes d'apprentissage automatique pour rendre les services plus modulaires et interopérables (nature du document, langues, etc.). Les expérimentations et évaluations ont été réalisées sur un corpus de 32 romans français publiés entre 1800 et 1914 et dont l'action se déroulent principalement à Paris (Table1).

Titre	Auteur	Date	Titre	Auteur	Date
La maison du Chat-qui-pelote	Balzac	1830	L'Assommoir	Zola	1877
Ferragus	Balzac	1833	Nana	Zola	1880
La fille aux yeux d'or	Balzac	1835	Une belle journée	Céard	1881
Le père Goriot	Balzac	1835	Pot-Bouille	Zola	1882
Grandeur et décadence de César Birotteau	Balzac	1837	Au Bonheur des dames	Zola	1883
Les mystères de Paris	Sue	1842	Le vingtième siècle	Robida	1883
Sans Cravate ou les Commissionnaires	Kock	1844	Sapho	Daudet	1884
L'envers de l'histoire contemporaine	Balzac	1848	Bel-ami	Maupassant	1885
M. Choublanc à la recherche de sa femme	Kock	1856	L'oeuvre	Zola	1876
Les misérables	Hugo	1862	La vie électrique	Robida	1892
Les demoiselles de magasin	Kock	1863	Paris	Zola	1897
Paris au XXème siècle	Verne	1863	La charpente	Rosny jeune	1900
L'éducation sentimentale	Flaubert	1869	Mr. Bergeret à Paris	France	1901
La Curée	Zola	1871	La Maternelle	Frapié	1904
Le ventre de Paris	Zola	1873	La Vague rouge	Rosny aîné	1910
Jack	Daudet	1876	Dans les rues	Rosny aîné	1913

Table 1: Liste de 32 romans du corpus.

La suite de ce rapport se compose de la manière suivante : la section 2 présente les différentes définitions en lien avec le sujet, la section 3 décrit les tâches de géocoding et de désambiguïsation. La section 4 présente les méthodes d'apprentissage automatique testées et leur expérimentations. Enfin, la section 5 conclut ce rapport.

2 Définitions des concepts et travaux existants

Ce projet se situe dans un contexte de recherche d'information (RI) et de traitement automatique de la langue (TAL) dont une des tâches principales est la reconnaissance des entités nommées. Cette tâche consiste à repérer les différentes entités d'un texte et à les catégoriser selon une typologie pré-définie (lieu, personne, organisation, etc.). En particulier, nous nous concentrons dans ce travail

sur les entités géographiques, c'est-à-dire les noms de lieux faisant référence à une localisation (e.g., nom de ville, nom de rue, etc.). On parle alors de geoparsing ou de résolution des toponymes [5]. De nombreux travaux dans la littérature traitent de cette problématique nécessaire pour interpréter les informations géographiques exprimées dans les textes [9,10,11].

2.1 Entité Nommée Étendue

Dans ce projet nous nous intéresserons à un type d'information géographique appelé Entité Nommée Étendue de lieu [3]. C'est une structure construite à partir d'un nom propre (entité nommée) et d'un ou plusieurs concepts. Une entité nommée étendue (ENE) peut posséder plusieurs niveaux d'encapsulation en fonction du nombre de concepts qui la composent. Dans le cas d'une entité de lieu, ces concepts font références à des objets de nature géographique (ville, rue, lac, etc.). Le concept d'ENE encapsule également les relations spatiales associées à l'entité décrite (e.g., 'au sud de Lyon', 'à 10 km du lac d'Annecy') on parle alors d'ENE relatives (ENER) que l'on différencie des ENE absolues (ENEA).

$$
\text{ENER}\left[\begin{array}{ll}
\textit{level} & 2 \\
\textit{type} & \textit{place name} \\
\textit{comp.} & \text{NP,IN,OFFSET,} \\
\text{ENEA} & \left[\begin{array}{ll}
\textit{level} & 1 \\
\textit{type} & \textit{location} \\
\textit{cat.} & \textit{descriptive} \\
\textit{comp.} & \text{NN,IN,} \\
\text{ENEA} & \left[\begin{array}{ll}
\textit{level} & 0 \\
\textit{type} & \textit{location} \\
\textit{cat.} & \textit{pure} \\
\textit{comp.} & \text{NNP} \\
\textit{lex.} & \textit{Rivoli}
\end{array}\right] \\
\textit{lex.} & \textit{rue de Rivoli}
\end{array}\right] \\
\textit{lex.} & \textit{angle de la rue de Rivoli}
\end{array}\right]
$$

Fig. 1: Description de l'ENE 'angle de la rue de Rivoli'

Dans l'exemple (Fig. 1), Rivoli est le nom propre à partir duquel on construit l'ENE. C'est l'ENE de niveau 0. Puis, on a l'entité de niveau 1 : rue de Rivoli. Enfin, on a l'entité de niveau 2 : l'angle de la rue de Rivoli.
Les informations contenues dans les différents niveaux d'encapsulation permettent d'aider lors de la phase de désambiguïsation.

2.2 Geoparsing

Le geoparsing est un processus qui se décompose en deux étapes principales :

– geotagging
– geocoding

Le geotagging consiste à identifier les références géographiques dans un texte (noms de lieux, relations spatiales, etc). Le but de cette première étape est de reconnaître et extraire les lieux mentionnés dans le texte. Cette partie peut être difficile car l'auteur a différentes façon d'évoquer un lieu: il peut y faire allusion directement ou bien de façon détournée. Parfois ces références sont déguisées (e.g "la ville des Parisiens") ou bien complètement imaginaires. Cette étape appelée geotagging a déjà été implémentée lors de précédents travaux [8,9].

La deuxieme étape (geocoding) consiste à assigner des coordonnées GPS aux références trouvées dans le texte. Il existe des index géographiques (gazetiers) qui permettent de retrouver les coordonnées d'un lieu à partir de son nom. Cette étape peut s'avérer complexe car ces ressources ne sont pas exhaustives et parfois les entités identifiées dans le texte font référence à des lieux imaginaires. Il sera alors relativement peu probable de trouver des coordonnées pour un tel lieu dans ces index. Cependant une entité pour laquelle on ne trouve rien n'est pas forcement imaginaire. Il peut s'agir d'une entité, dont le nom actuel a changé, qui a disparu avec le temps ou qui n'est tout simplement pas référencé dans les index.

Un des problèmes de cette deuxième étape est qu'il est possible d'obtenir plusieurs localisations pour une même entité. Par exemple "Paris" peut faire référence à la capitale française ou à une des 22 villes américaines portant ce nom. En se servant de toutes les informations dont nos disposons, il s'agira de retrouver le bon résultat parmi tout ceux disponibles. C'est ce qu'on appelle la désambiguïsation.

2.3 Repérage des ENE avec l'outil PERDIDO

Ce travail s'appuie sur la chaîne de traitement PERDIDO [3,8] qui a pour rôle l'annotation géo-sémantique des entités nommées étendues et d'informations géographiques (relations spatiales et expressions de déplacements). Les différents traitements disponibles au sein de la plateforme PERDIDO[1] sont disponibles sous forme de services Web. La tâche de geotagging est implémentée dans PERDIDO avec un système basé sur un ensemble de règles. Il s'agit de règles linguistiques utilisant des patrons morpho-syntaxiques implémentées par une cascade de transducteurs [2] au sein de la plateforme Unitex[2].

Le repérage automatique des entités nommées avec PERDIDO va nous servir dans les deux tâches que nous décrivons dans ce rapport. Dans un premier temps, la tâche de géocoding et de désambiguïsation (voir section 3) s'appuie sur le résultat de l'annotation automatique réalisée avec PERDIDO afin d'associer des coordonnées géographiques (latitude, longitude) aux noms de lieux repérés dans les textes. Dans un deuxième temps, l'annotation automatique des ENE produite

[1] `http://erig.univ-pau.fr/PERDIDO/`
[2] `https://unitexgramlab.org/fr`

par PERDIDO servira à construire les bases d'apprentissage et d'évaluation pour l'expérimentation des méthodes d'apprentissage automatique (voir section 4). En effet, la constitution d'un corpus d'apprentissage annoté manuellement est une étape qui prend énormément de temps et très fastidieuse. Nous souhaitons donc évaluer la capacités des méthodes d'apprentissage à généraliser à partir d'une entrée bruitée et contenant des erreurs.

3 Geocoding et désambiguïsation

Un première tâche de ce travail a été de concevoir une méthode permettant de géolocaliser les ENE repérées dans les textes. Les textes utilisées pour les expérimentations sont ceux du corpus de 32 romans précédemment décrit (Table 1).

3.1 Géocoding

Une fois les ENE repérées à l'aide de la chaîne de traitement PERDIDO (voir section 2.3), celles-ci doivent être géolocalisées. Pour cela nous allons faire appel à des index géographiques. Certains sont requêtables gratuitement via des API, les autres devront être importés dans une base de données locale. Ainsi pour chaque ENE, on enverra des requêtes à ces index pour récupérer sa latitude et sa longitude. En fonction du type d'entité ("street" ou "others") que l'on souhaite géolocaliser, on utilisera seulement certains index. Pour obtenir cette information, nous réaliserons des tests de performance sur chaque index afin d'obtenir le rappel, la précision et le F-score. La précision est obtenue en calculant la proportion de réponses pertinentes obtenues parmi les réponses et le rappel est obtenue en calculant la proportion de réponses pertinentes obtenues sur le nombre total de réponses pertinentes attendues. Enfin la moyenne harmonique des deux nous donne le F-Score. Chaque index sera évalué individuellement en géocodant les ENE sur une sélection de 7 romans pris aléatoirement, représentant un total de 392 entités de type "street" et 274 entités de type "others". Chaque résultat sera vérifié manuellement..
Nous verrons, dans cette partie, les différents index utilisés ainsi que leurs particularités.

GeoHistorical [3]

Cette base de données contient des références géographiques sur Paris et ses environs sur une période de 1789 à 1950. C'est un index très efficace sur les rues de Paris car il permet de retrouver presque 90 % des entités de type rue mais il n'est pas très efficace sur les autres entités. Cependant il possède une très bonne précision dans n'importe quel cas et provoquera par conséquent peu de bruit dans les réponses (Table 2). On va donc l'utiliser pour requêter les rues ("Street") mais aussi les autres entités ("Others").

[3] `http://www.geohistoricaldata.org/`

	Total	Street	Other
Rappel (%)	54.52	89.94	14.29
Précision (%)	99.39	100	95.24
F-Score (%)	70.41	94.7	24.85

Table 2: Performance de l'index géographique GeoHistorical sur 7 textes du corpus

IGN [4] Une base de données proposée par l'Institut National de l'information géographique et forestière qui contient des lieux français. C'est un index assez performant sur les rues puisqu'il retrouve plus de 70 % des rues requêtées mais bien moins efficace sur les autres entités avec seulement 20 % de rappel et 82% de précision (Table 3). On l'utilisera donc seulement pour les rues ("Street").

	Total	Street	Other
Rappel (%)	47.99	72.61	20.57
Précision (%)	92.86	95.8	82.86
F-Score (%)	63.28	82.61	32.96

Table 3: Performance de l'index géographique IGN sur 7 textes du corpus

Nominatim [5] Une API de MapQuest qui permet de requêter sur les données d'OpenStreetMap. C'est l'index le plus complet au niveau des performances. Il dispose d'un bon taux de précision pour n'importe quel type d'entité : si il a trouvé une réponse, il y a plus de 90 % de chance qu'elle soit correcte. Il possède un rappel correct pour les rues et le meilleur rappel parmi les index pour les autres entités (Table 4). On l'utilisera donc pour requêter n'importe quel type d'ENE de lieux.

	Total	Street	Other
Rappel (%)	65.19	76.43	52.21
Précision (%)	95.98	99.17	91.03
F-Score (%)	77.64	86.33	66.36

Table 4: Performance de l'index géographique Nominatim sur 7 textes du corpus

[4] `https://geoservices.ign.fr/documentation/geoservices/geocodage.html`

[5] `https://developer.mapquest.com/documentation/open/nominatim-search/search/`

Geonames [6] Un index téléchargeable ou requêtable via une API qui contient des informations sur des lieux connus du globe.
Il possède un taux de précision correct pour les deux types d'ENE de lieux mais il retrouve seulement 1 rue sur 10 (Table 5). On va donc limiter son utilisation aux autres entités ("Others").

	Total	Street	Other
Rappel (%)	23.83	10.19	39.01
Précision (%)	94.67	94.12	94.83
F-Score (%)	38.08	18.39	55.28

Table 5: Performance de l'index géographique Geonames sur 7 textes du corpus

Wikipedia [7] Une API de Geonames qui permet de requêter sur les entrées géolocalisées de Wikipédia.
C'est un index qui possède moins d'entrées que les autres mais son bon taux de précision et son rappel correct pour les deux types d'entités font de lui un bon complément aux autres index (Table 6). On l'utilisera donc pour n'importe quel type d'ENE de lieux.

	Total	Street	Other
Rappel (%)	36.33	31.88	41.43
Précision (%)	96.46	100	93.55
F-Score (%)	52.78	48.35	57.43

Table 6: Performance de l'index géographique Wikipédia(Geonames) sur 7 textes du corpus

Format des réponses Chacun des index précédemment cités possède un modèle de réponse différent. Pour procéder à la phase de désambiguïsation, on va devoir formaliser et normaliser ces réponses. Pour chaque réponse, nous aurons besoin des informations suivantes :

- les attributs :
  - le type de lieu : rue, boulevard, musée, hôtel, bois, ...
  - le pays : le pays au format ISO-3166
  - l'état : l'état ou la région si l'information est disponible

[6] `http://www.geonames.org/export/geonames-search.html`
[7] `http://www.geonames.org/export/wikipedia-webservice.html#wikipediaSearch`

  - la ville : la ville quand elle est disponible
- les éléments de la réponse :
  - le nom court : le nom du lieu sans l'adresse complète. C'est ce qui se rapproche le plus du nom de l'entité.
    Lorsque l'index qui retourne la réponse gère les noms alternatifs, le nom court utilisé sera celui qui est le plus proche du nom spécifié dans la requête.
  - le nom complet : le nom du lieu avec plus de détails. En fonction de l'index, le nom complet peut être plus ou moins long.
  - les coordonnées : ce sont les informations les plus importantes car c'est ce qu'on cherche à obtenir lors de la phase de geocoding. Les coordonnées sont décrites sous la forme de coordonnées GPS exprimées en degrés décimaux.
  - des distances orthographiques : ces distances vont servir à exprimer la pertinence de la réponse par rapport à la requête. Elles seront comprises entre 0.0 et 1.0, 0.0 indiquant une correspondance parfaite et 1.0 une inégalité complète. Deux distances différentes seront utilisées :
    * distance de Damerau-Leveshtein normalisée : c'est une mesure de similarité entre chaîne de caractères. Dans sa version classique, elle est égale à la somme minimale des opérations élémentaires qu'il faut effectuer pour passer d'une chaîne à une autre (ajout, suppression, substitution, transposition). On normalise cette valeur en la divisant par la longueur de la plus grande des deux chaînes comparées.
    * coefficient de Dice : c'est une mesure de similarité entre deux échantillons de valeurs. Elle peut néanmoins s'appliquer à des chaînes de caractères en utilisant l'ensemble des bigrammes de chacune des chaînes comme échantillon de valeur. Elle se définit par la formule suivante :

      $$s = \frac{2 * |X \cap Y|}{|X| + |Y|} \quad (1)$$

      Nous prendrons $1 - s$ comme valeur car $s = 1.0$ quand les deux chaînes sont identiques et $s = 0.0$ quand elles sont différentes.
      Comme la distance de Damerau-Levenshtein est très sensible à l'ordre des mots, il est possible que certains résultats pourtant très pertinents obtiennent un mauvais score. Le coefficient de Dice n'y est pas sensible. Il permettra donc de s'assurer de la non-pertinence d'une réponse.
    * la moyenne des deux : à titre indicatif

Pour chaque texte étudié, on va créer un fichier XML de réponses (Fig. 2) qui regroupe les résultats renvoyés par les index pour chaque entité avant de procéder à la phase de désambiguïsation. L'élément racine du fichier possède deux attributs :

- le nombre d'entité extraites lors de la phase de geoparsing
- le nombre de mots contenus dans le document

L'élément racine possédera un élément par entité recherchée. Chacun de ces éléments possédera les attributs suivants :

- locationName : le nom de l'entité extraite lors de la phase de geoparsing
- shortLocationName : le nom court de l'entité
- locationCity : la ville dans laquelle chercher si elle est connue
- locationType : quand l'entité a été identifié comme étant une rue, cet attribut vaut "street", sinon il vaut "others"
- locationFeature : le "feature" de l'entité, c'est à dire ce qui est contenu dans la balise geogFeat de l'entité.
  Par exemple, dans "rue de Rivoli" et "bois de Boulogne", "rue" et "bois" sont les features.
- locationProperName : le nom propre à partir duquel a été identifié l'entité.
  Par exemple, dans "rue de Rivoli" et "bois de Boulogne", "Rivoli" et "Boulogne" sont les noms propres.
- textPosition : la position de l'entité dans le texte représentée par le numéro du premier mot de l'entité
- totalResponses : le nombre de réponses reçu par tous les index pour cette entité.

```
<?xml version="1.0" encoding="UTF-8"?>
<assommoir totalEsne="255" nbWords="169644">
  <esne1 locationName="boulevard de la Chapelle" shortLocationName="boulevard de la Chapelle"
    locationCity="Paris" locationType="street" locationFeature="boulevard"
    locationProperName="Chapelle" textPosition="985" totalResponse="173">
    <responses src="GeoHistorical" poids="2" numberResponse="10">
      <response id="1" type="Unknown" country="FR" city="PARIS" state="ÎLE-DE-FRANCE">
        <locationFound>boulevard DE LA CHAPELLE</locationFound>
        <completeName>boulevard DE LA CHAPELLE, Paris</completeName>
        <coordinates>
          <longitude>2.35552690147662</longitude>
          <latitude>48.8840506423837</latitude>
        </coordinates>
        <distance type="Levenshtein">0.0</distance>
        <distance type="Dice">0.0</distance>
        <distance type="AVG">0.0</distance>
      </response>
```

Fig. 2: Extrait du fichier de réponse du texte L'Assommoir avant désambiguïsation

Pour chacune des entités, on a un élément par index requêté. Cet élément, qui possède un attribut *numberResponse* pour indiquer le nombre de réponses qu'a renvoyé l'index, contient toutes les réponses dont le format a été expliqué précédemment

3.2 Filtrage des résultats du geocoding

Après interrogation des index, on obtient pour chacune des entités : zéro, une ou plusieurs réponses. La pertinence de ces réponses est variée et cela est surtout

lié à la façon dont la recherche textuelle de l'entité est faite. Lorsqu'on a aucune réponse, cela signifie qu'aucun index n'a trouvé de références à l'entité requêtée : il est donc probable que l'entité ne soit pas un lieu, n'existe plus, ou ne soit pas référencée.

Quand on obtient plusieurs réponses, cela pose un problème car on cherche à obtenir une seule géolocalisation pour notre entité. On va donc devoir faire un tri sur les résultats retournés. On va donc appliquer plusieurs filtres sur nos résultats pour essayer de supprimer les réponses non pertinentes.

Cette étape de filtrage a plusieurs fonctionnalités, elle permet de supprimer les mauvaises réponses que les index ont renvoyées et, par conséquent, pour les entités qui sont des lieux imaginaires ou qui ne sont pas des lieux, il ne subsistera aucune réponse. Pour les autres entités, ce filtrage permettra de réduire au maximum le nombre de réponses pertinentes avant de commencer à désambiguïser.

Filtre de distance orthographique La première chose sur laquelle on commence à filtrer, c'est la similarité entre la chaîne retournée et la chaîne requêtée. On va donc utiliser des mesures de similarité sur le nom des lieux. Une des mesures de similarité la plus utilisée est la distance de Damerau-Levenshtein qui compte le nombre d'opérations élémentaires (ajout, suppression, substitution, transposition) pour passer d'une chaîne à une autre. Comme toutes nos chaînes de caractères sont de tailles différentes, on va normaliser cette distance en la divisant par la longueur de la chaîne de plus grande taille entre la chaîne de la requête et celle de la réponse. On obtient alors, pour 2 chaînes comparées, une valeur entre 0 et 1, 0 signifiant que les deux chaînes sont identiques et 1 qu'elles sont complètement différentes.

Le problème de cette distance de Damerau-Levenshtein est qu'elle est très sensible à l'ordre des mots. Cela va poser problème pour certaines langues car on va supprimer des résultats pertinents juste parce que les mots sont inversés. Par exemple "Moor of Rannoch" et "Rannoch Moor" désignent le même lieu mais obtiennent une distance de Damerau-Levenshtein très mauvaise. Pour pallier à ce problème on va rajouter une deuxième mesure de similarité appelée indice de Dice. Cette mesure, moins sensible à l'ordre des mots que la première, nous servira de vérification avant la suppression d'un résultat. Si un résultat obtient une distance de Damerau-Levenshtein normalisée supérieure à 0.5 et un indice de Dice supérieure à 0.45, alors on supprime ce résultat car on le considère trop différent de ce qui a été demandé.

Filtre à doublons (nom complet) Certains index retournent des doublons exactes dans leur réponses. Ce filtre va permettre de les supprimer. Pour cela on va comparer le nom complet de chaque réponse. Si on en trouve deux identiques, on vérifie qu'ils font référence à la même position géographique : si la distance géographique qui sépare ces deux entités est inférieure à 1 km, on supprime une des deux réponses.

Filtre de feature On appelle *feature* le groupe de mot qui permet de définir le type du lieu représenté par l'entité. Par exemple, dans "boulevard Magenta", "boulevard" est le *feature*. Parfois aucun *feature* n'est identifié lors de la phase de geotagging. Dans ce cas ce filtre ne s'applique pas.
L'intérêt de ce filtre c'est de supprimer les résultats dont le feature ne correspond pas. Par exemple si on cherche "rue de Montplaisir", on peut supprimer le résultat "boulevard de Montplaisir" car on cherche une "rue" et non un "boulevard". Si un feature est présent dans les annotations, on va regarder s'il est contenu dans la chaîne de réponse. Si ce n'est pas le cas, on peut supprimer le résultat.

Filtre de ville Lorsqu'on dispose d'un peu de contexte, il est plus facile de retrouver la localisation d'une entité. C'est dans ce but qu'a été créé ce filtre. En effet, si on dispose du nom de la ville dans laquelle rechercher, on peut supprimer tous les résultats qui ne se situe pas dans cette ville. Après plusieurs expérimentations, il apparaît que ce filtre n'est pas pertinent pour tout ce qui n'a pas été classé comme une rue (c-à-d *others*). Il ne sera donc appliqué que sur les rues pour ne pas faire baisser les résultats. Bien évidemment ce filtre ne fait rien si aucune ville n'a été spécifiée.

Filtre de nom propre Après le passage des filtres cités précédemment, les résultats restants sont déjà parmi les plus pertinents. Cependant il arrive que certaines mauvaises réponses subsistent à ce stade du filtrage. Par exemple, lors de la recherche de "cimetière de Montparnasse", on obtient le résultat "cimetière de Montmartre" que les précédents filtres n'ont pas réussi à supprimer. C'est pour cela que le filtre de nom propre a été créé. Ce filtre utilise le nom propre qui a été annoté lors de la phase de geotagging et le compare aux différentes réponses. Si une réponse ne contient pas ce nom propre, alors elle est jugée non pertinente et supprimée. Il arrive parfois que le nom propre désignant une rue ait légèrement changé avec le temps. C'est par exemple le cas pour "la rue Marboeuf" devenue "la rue Marbeuf". On tolère donc une faute de frappe lorsque l'on regarde si le nom propre est contenu dans la réponse.

Filtre à doublons (nom court) Le premier filtre à doublons permet de filtrer par rapport aux doublons purs d'un même index, c'est à dire les réponses qui ont des noms complets identiques. Ce filtre va chercher les doublons par rapport aux noms courts. Avec certains index on obtient parfois plusieurs fois la même rue avec seulement le numéro de rue qui change. On supprime les résultats dont le nom court est distant de 0.3 (distance de Damerau-Levenshtein normalisée) et géolocalisés à moins de 1km de distance.

Filtre à doublons inter-index Jusqu'à présent, les filtres à doublons procédaient seulement sur les réponses d'un index. Cependant quand un résultat est pertinent et/ou connu, il est généralement retourné par plusieurs index. Chaque index

est différent et par conséquent un index peut parfois renvoyer une information qu'un autre ne possédait pas. Le rôle de ce filtre c'est de retrouver les doublons entre index pour les fusionner en un seul résultat plus complet.

Filtre matching parfait Après passage des premiers filtres, on a réussi à séparer les bons résultats des résultats moyens. Cependant il est possible qu'un résultat excellent (c'est à dire parfaitement identique à ce qu'on cherchait) se retrouve au milieu de bons résultats. Si cela arrive, ce filtre permet de garder seulement les résultats excellents.

3.3 Désambiguïsation des entités spatiales

Lorsque plusieurs résultats sont disponibles pour une entité de lieu, il faut retrouver le bon, s'il fait partie des résultats. Le filtrage nous a permis de réduire le nombre de réponses au maximum mais il ne nous permet pas tout le temps de supprimer l'ambiguïté. C'est pourquoi il nous faut une méthode pour choisir, parmi les réponses restantes, laquelle a le plus de chance d'être celle qu'on recherche. Pour cela, nous avons testé plusieurs méthodes que nous décrivons ici.

Centroïde La première méthode que nous avons essayée est sûrement la plus basique. Le principe est simple : on calcule un centroïde à partir des points qui sont non-ambiguës et pour chaque point ambiguë, on prend la réponse qui est la plus proche du centroïde en terme de distance.

C'est une méthode qui fonctionne plutôt bien lorsqu'on sait que tous les lieux sont regroupés dans une même zone mais qui est limitée quand ce n'est pas le cas. De plus, elle est très sensible aux erreurs. Il suffit qu'un point non-ambiguë soit erroné pour que le centroïde soit complètement décalé. Par exemple, avec les romans situés à Paris, un point erroné placé aux États-Unis va provoquer un décalage de plus de 100 km du centroïde.

Cluster + Centroïde La première méthode n'était pas adaptée pour les textes avec plusieurs lieux dispersés. C'est à partir de ce constat que nous avons réfléchi à une deuxième méthode.

Nous allons partir du principe que dans un roman, les entités qui sont proches géographiquement sont aussi proches dans le texte. En regroupant les entités proches dans le texte sous la forme de clusters, on devrait obtenir, en théorie, des clusters représentants les différentes régions géographiques citées dans le texte.

Cependant, en analysant la répartition des entités dans le texte, on s'aperçoit qu'elles sont uniformément réparties dans le récit (Fig. 3). Dans cet exemple, une partie de l'action se déroule à Paris et une autre à Cannes. Les entités qui désignent des lieux dans la région de Cannes sont situées entre le 65 000eme mot et le 85 000eme mot mais sur le graphique, on n'observe pas de cluster particulier

dans cette zone. Il apparaît donc que les clusters textuels ne correspondent pas forcément à des zones géographiques. Cette méthode a donc été abandonnée.

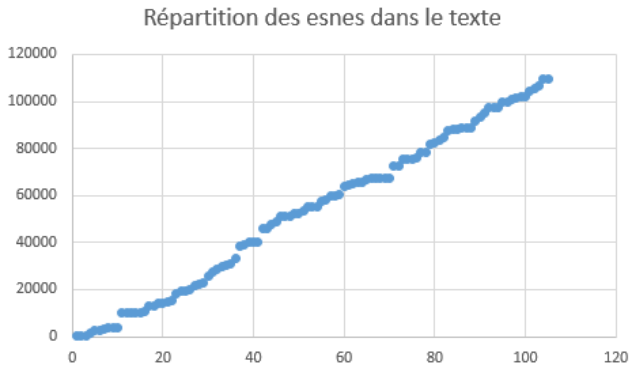

Fig. 3: Répartition des ESNE dans le texte "Belami" le numéro de l'entité en abscisse et la position de l'entité dans le texte en ordonnée

Voisins textuels Une autre méthode pour désambiguïser est d'utiliser la proximité textuelle d'une autre façon en utilisant pour chaque entité ambiguë son plus proche voisin dans le texte et de choisir le résultat le plus proche de ce dernier. Pour que cette technique soit la plus précise possible, il faut commencer à désambiguïser l'entité la plus proche textuellement d'un point non-ambigu, puis recommencer le procédé jusqu'à ce que toutes les entités soient désambiguïsées.

C'est la méthode la plus efficace pour le moment puisqu'on respecte le principe du "proche dans le texte, proche sur la carte" tout en évitant une propagation des erreurs trop importante. Sur ce dernier point, on remarque, en expérimentant, que si il y a une erreur sur une des entités, cette erreur se propagera rarement au delà d'un ou deux voisins.

Groupe de points les moins distants Jusqu'à présent, les techniques utilisées s'appuyaient sur les entités non-ambiguës pour désambiguïser les autres. Cependant, il arrive que sur de petits textes, toutes les entités soient ambiguës. Il faut donc trouver une méthode qui ne soit pas dépendante des entités déjà valides.

Dans ce type de situation, on se retrouve avec plusieurs combinaisons de réponses parmi lesquelles il faut choisir. Comme vu précédemment, les lieux cités dans un texte appartiennent au même contexte et donc ils sont souvent assez proches. On va donc sélectionner la combinaison de points qui minimise la distance. C'est une méthode assez efficace mais qui possède un gros désavantage : il faut calculer la distance pour chaque combinaison et si le nombre de points est trop grand, le nombre de combinaison à calculer explose.

Combinaison de méthodes. On va maintenant essayer de combiner plusieurs critères pour la désambiguïsation et nous calculerons un score en fonction de ces critères. Après le calcul des points, seule l'entité avec le meilleur score est gardée.

Avec cette méthode, nous allons procéder à une désambiguïsation entité par entité jusqu'à ce qu'il ne reste plus aucun ambigu. Avant de commencer à attribuer des points, il faut donc commencer par sélectionner l'entité par laquelle démarrer le cycle. Pour plus de clarté nous appellerons l'entité sélectionnée pour la désambiguïsation entité$_S$. Nous allons réutilisé la méthode des voisins textuelles et, par conséquent, l'entité$_S$ sera celle qui est la plus proche textuellement d'une entité non-ambiguë.

Une fois que l'entité$_S$ a été désignée, nous allons utiliser des critères de sélection pour calculer un score. La liste de ces critères de sélections est la suivante (Fig.4) :

Fig. 4: Le cycle de désambiguïsation de la méthode par score

– voisins textuels : on va réutiliser la méthode des voisins textuelles vu précédemment. Parmi les réponses de l'entité$_S$, on choisi celle qui est la moins distante géographiquement de son plus proche voisin textuel non-ambigu et on lui attribut 1.25 point.

- centroïde : on reprend la méthode des centroïde vu précédemment. On calcule un centroïde à partir de tous les points valides. Les entités déjà désambiguïsées seront considérées comme valides et donc on doit calculer le centroïde à chaque itération du cycle. Après calcul du centroïde on attribut 1 point à la réponse de l'entité$_S$ la plus proche du centroïde.
- référence : en tant qu'humain, je désambiguïse une entité en observant les autres entités. Par exemple, lorsque je cherche à désambiguïser "Glencoe" et que je m'aperçois qu'une autre entité est "Ecosse", il y a de fortes chances que la réponse "Glencoe, Ecosse" soit la bonne. De plus, lorsque l'auteur d'un récit nous décrit une rue, il est plus probable qu'il commence par décrire le pays dans lequel elle se situe, puis la région, puis la ville avant de commencer à décrire la rue.
  Sur ce principe, on ajoutera 1 point par référence à chaque réponse de l'entité$_S$ qui possédera dans son nom complet une référence à une autre entité.
- pays, état, ville : un constat que l'on peut faire en observant les différents résultats, c'est que comme ils sont proches sur la carte, on retrouve souvent les mêmes pays, états et villes dans les réponses. Cela correspond à une zone de recherche pertinente puisqu'elle est obtenue en croisant les réponses des différentes entités. On va donc chercher à favoriser les réponses qui sont dans cette zone de recherche.
  Pour cela on va calculer les pays, états et villes qui reviennent le plus fréquemment dans les réponses et les proportions qu'ils représentent par rapport à tous les pays, états et villes cités. Pour chaque réponse de entité$_S$, on ajoutera 1 point par attribut de pays, état ou ville fréquent dans les réponses.
- ville ciblée : si on connaît la ville dans laquelle le texte prend place, on favorisera naturellement les réponses qui sont situées dans cette ville.
  On ajoutera donc 1 point à chaque réponse situé dans la bonne ville.
  Bien évidemment, ce critère ne fonctionne que si une ville a été spécifiée au préalable.
- distance orthographique : plus le nom de la réponse est proche du nom de l'entité$_S$, plus il y a de chance que ce soit la bonne réponse. On va donc attribuer 1 point aux réponses dont la distance orthographique est la plus faible.

A la fin du cycle on désambiguïse en sélectionnant la réponse qui a accumulé le plus de points. Il arrive parfois que plusieurs réponses aient le même nombre maximum de points. Dans ce cas on supprime toutes les autres réponses, on laisse l'ambiguïté sur l'entité$_S$ et on le marque pour le traiter à la fin quand on aura plus d'informations valides.

Le fichier, une fois désambiguïsé, est transformé en géojson, un format exploitable par un système d'information géographique (SIG). On peut alors visualiser chaque ENE de lieux directement sur la carte (Fig. 5). Sur cet exemple, on peut voir le centroïde (en blanc) qu'on a construit à partir des entités non-ambiguës (en noir), ainsi que les réponses d'une entité ambiguë qui sont de la même couleur.

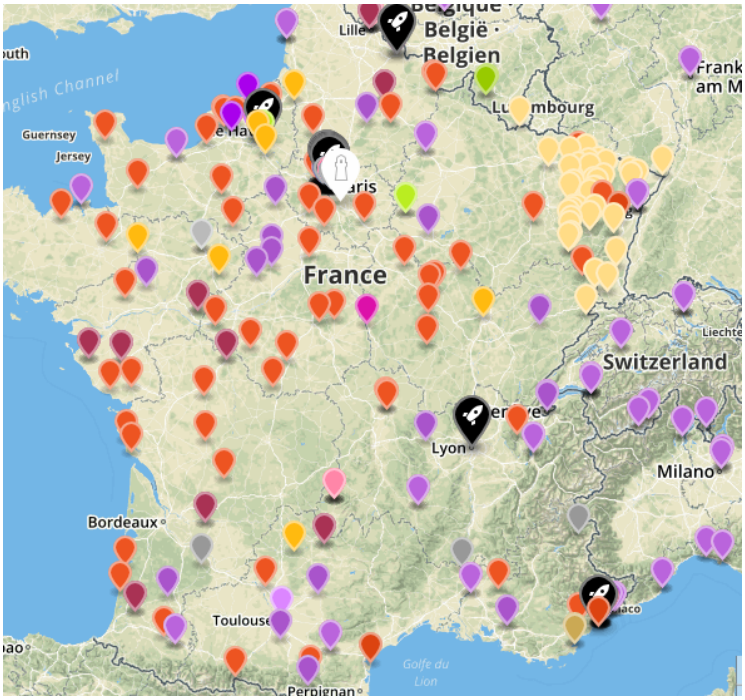

Fig. 5: Affichage des entités nommées étendues de lieux du texte "Belami" dans *geojson.io*

4 Repérage des ENE par apprentissage supervisé

Une deuxième tâche de notre travail s'est intéressée au repérage des ENE grâce à des méthodes d'apprentissage automatique supervisées.
Nous allons donc nous intéresser à plusieurs outils de reconnaissance d'entités nommés que nous adapterons pour essayer de trouver une méthode performante pour apprendre à reconnaître les entités nommées étendues de lieux. Il faudra pour cela transformer les fichiers annotés du corpus de romans (Fig. 6) en un format adapté à chaque outil. Le corpus sera ensuite séparé en deux parties : une partie sur laquelle le modèle apprendra et une partie sur laquelle on l'évaluera. Pour chacun des outils, nous entraînerons plusieurs modèles en variant la taille du corpus d'apprentissage, afin de déterminer le nombre de données nécessaires pour obtenir un modèle avec des performances satisfaisantes.

Nous chercherons aussi à évaluer la capacité de généralisation de chaque modèle, c'est à dire sa capacité à faire de bonnes prédictions sur des données qu'il n'a jamais rencontrées. Pour cela, nous supprimerons certains types d'entités nommées du corpus d'apprentissage lors de l'entraînement du modèle et nous observerons la capacité de ce dernier à identifier correctement ces entités dans la phase de test.

```
<w lemma="il" type="PRO" xml:id="w53">il</w>
<phr type="motion">
  <motion type="median">
    <w lemma="prendre" type="V" xml:id="w54">prit</w>
  </motion>
  <w lemma="sa" type="DET" xml:id="w55">sa</w>
  <term type="N">
    <w lemma="course" type="N" xml:id="w56">course</w>
  </term>
</phr> ,<w lemma="galoper" type="V" xml:id="w57">galopa</w>
<w lemma="dégingander" type="V" xml:id="w58">dégingandé</w> ,
<w lemma="éperdre" type="V" xml:id="w59">éperdu</w> ,
<w lemma="le" type="DET" xml:id="w60">le</w>
<w lemma="long" type="A" xml:id="w61">long</w>
<w lemma="de" type="PREPDET" xml:id="w62">du</w>
<placeName n="1" xml:id="esne1">
  <geogName type="R" subtype="ST">
    <geogFeat>
      <w lemma="quai" type="N" xml:id="w63">quai</w>
    </geogFeat>
    <w lemma="de" type="PREP" xml:id="w64">de</w>
    <w lemma="la" type="DET" xml:id="w65">la</w>
    <rs type="unknown">
      <name type="unknown">
        <w type="NPr" lemma="" xml:id="w66">Grève</w>
      </name>
    </rs>
  </geogName>
</placeName> .
```

Fig. 6: Extrait du fichier XML d'annotation par Perdido pour le roman "L'Oeuvre"

Pour entraîner et tester chaque modèle, nous utiliserons le corpus de romans (Table 1) que nous annoterons automatiquement avec Perdido. De ce fait, dans les fichiers utilisés pour l'apprentissage, certaines entités seront partiellement identifiées ou erronées, comme par exemple :

$$\boxed{\text{rue de la Tour}} - d'Auvergne \tag{2}$$

$$\boxed{\text{rue Notre}} - Dame - des - Champs \tag{3}$$

$$\boxed{\text{rue de la Barrière}}\ des\ Gobelins \tag{4}$$

Nous étudierons aussi la faculté du modèle à corriger ces erreurs. Nous utiliserons pour cela une version annotée corrigée manuellement des textes "L'oeuvre"

d'Emile Zola et "L'education sentimentale" de Gustave Flaubert pour effectuer les tests.
Les outils de reconnaissance d'entités nommées ont été choisi parmi les plus performants pour la reconnaissance d'entités nommées de lieux. Nous présenterons le principe de fonctionnement de chacun de ces outils avant d'évaluer leur efficacité sur des entités nommées étendues de lieux.

4.1 Stanford-NLP

Stanford-NLP[8] [7] est un outil de traitement automatique du langage naturel qui permet d'effectuer, entre autres, de l'étiquetage morpho-syntaxique, de l'opinion mining ou de la détection de co-référence. C'est un outil développé par le groupe du traitement du langage naturel de l'Université de Stanford.

Cet outil possède un module de reconnaissance d'entités nommées qui permet de labelliser plusieurs types d'entité comme des lieux, des personnes ou encore des organisations. Avec ce module, il est possible d'entraîner son propre modèle. Nous allons donc l'entraîner à reconnaître nos entités nommées étendues de lieux. Pour entraîner un modèle avec ce NER, il faut transformer le fichier XML créé par Perdido en un fichier txt compatible. La figure 7 donne un exemple de ce format pour l'extrait précédent (Fig. 6) : chaque mot est représenté sur une ligne avec son type grammatical, son label et son lemme. Chaque champ d'une ligne est séparé par une tabulation et on marque le changement d'une phrase par une ligne vide. Les labels sont encodés selon le format BIO, c'est à dire qu'on marque le début d'une entité par un "B-"[9], un morceau d'entité par un "I-"[10] et un mot qui n'est pas une entité est labellisé par un "O"[11].

L'outil de reconnaissance est basée sur les champs aléatoire conditionnels linéaires dont nous comparerons le fonctionnement avec celui des modèles de Markov cachés.

Champs aléatoires conditionnels Les champs aléatoires conditionnels (CAC) [12] présentés en 2001 par Lafferty et al. [4] ont ouvert de nouvelles portes pour l'analyse de séquences. Jusque là, les modèles de Markov cachés (MMC) étaient la méthode la plus utilisée pour ce type d'analyse.

Un CAC est un processus stochastique qui modélise les dépendances entre un ensemble d'observations discrètes réalisées sur une séquence discrète (à l'origine, une séquence de mots) et un ensemble d'étiquettes.

Le L-CAC, pour champ aléatoire conditionnel linéaire, est un modèle discriminant qui ne fait pas l'hypothèse qu'une observation conditionnée par son étiquette est indépendante des observations voisines.

[8] https://nlp.stanford.edu/software/CRF-NER.html
[9] B pour "Begin"
[10] I pour "Inside"
[11] O pour "Outside"
[12] Conditional Random Fields -CRF- dans la littérature anglaise

```
il      PRO     O       il
prit    V       O       prendre
sa      DET     O       sa
course  N       O       course
,       PUNC    O       ,
galopa  V       O       galoper
dégingandé      V       O       dégingander
,       PUNC    O       ,
éperdu  V       O       éperdre
,       PUNC    O       ,
le      DET     O       le
long    A       O       long
du      PREPDET O       de
quai    N       B-STREET        quai
de      PREP    I-STREET        de
la      DET     I-STREET        la
Grève   NPr     I-STREET        Grève
.       PUNC    O       .
```

Fig. 7: Extrait du fichier d'entrée de Stanford-NLP pour le roman "L'Oeuvre"

Définissons quelques notations pour la compréhension de la suite de cette partie :

- X = x_1,x_2,...,x_T est une séquence de T observations discrètes
- Y = y_1,y_2,...,y_T est la séquence des T étiquettes associées aux observations de la séquence X
- L est l'ensemble des étiquettes possible (les valeurs possibles pour les y_t)
- O est l'ensemble des observations (les valeurs possibles pour les x_t)

Un L-CAC est définit par :

$$p(X|Y) = \frac{1}{Z(X)} exp(\sum_{t}^{T} \sum_{k}^{K} \lambda_k f_k(y_{t-1}, y_t, x, t)) \tag{5}$$

Comme nous le voyons avec (5), la probabilité qu'une séquence d'étiquettes particulière Y soit associée à la séquence d'observations X est obtenue par une combinaison linéaire de poids λ_k associés à des fonctions f_k sur la séquence d'observations. Les poids λ_k sont les paramètres du modèle et peuvent être interprétés comme la fiabilité de l'information apporté par la fonction f_k. Les fonctions f_k rendent compte de l'occurrence d'une combinaison d'observations et de labels particulière. Elles sont définies par l'utilisateur lors de l'entraînement du modèle. Dans le cas de la reconnaissance d'entité nommées, comme dans ce projet, les fonctions caractéristiques peuvent être les suivantes :

- le mot courant
- le mot précédent
- le mot suivant
- la forme du mot

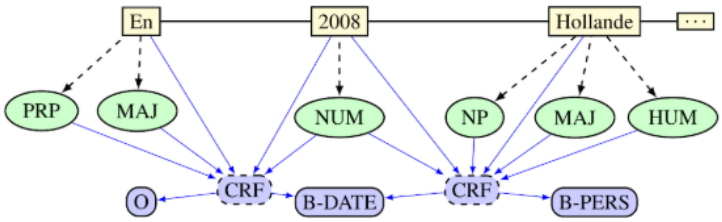

Fig. 8: Principe de fonctionnement des CRF

- la forme des mots voisins
- le type grammatical [13]
- le lemme du mot
- la position du mot dans la phrase
- les n-grammes du mot

Entraîner un CAC signifie optimiser ses paramètres λ_k sur un ensemble d'exemple.

Pour retrouver la meilleure séquence d'étiquettes pouvant être associée à une séquence d'observation, on utilise un processus de décodage qui consiste à retrouver la séquence d'étiquettes y* qui maximise $p(y|x)$:

$$y^* = argmax_y(p(y|x)) \tag{6}$$

La recherche de cette séquence est réalisée par un algorithme de programmation linéaire dynamique de type Viterbi.

4.2 MITIE

MITIE[14] est un outil de reconnaissance d'entités nommées développé à l'Institut de technologie du Massachusetts.
Pour utiliser ce NER, il n'est pas nécessaire de transformer le fichier XML en format d'entrée compatible avec MITIE. En effet, pour entraîner un modèle, on lui donne directement les phrases une par une, en indiquant les positions et les types des entités . Pour faciliter cette opération, nous allons quand même créer un format de fichier dans lequel on placera une phrase par ligne avec les mots de la phrase annotée comme ceci :
La_O rue_B-STREET de_I-STREET Rivoli_I-STREET est_O très_O souvent_O cité_O ._O
Il suffit ensuite de traiter le fichier ligne par ligne pour donner au modèle ce dont il a besoin.
MITIE utilise un algorithme de *word embedding* pour vectoriser les mots issus d'un texte et se sert ensuite de ces vecteurs pour entraîner une machine à vecteur de support.

[13] Part-Of-Speech Tag en anglais
[14] https://github.com/mit-nlp/MITIE

Word Embedding Le word embedding [15] est un algorithme qui permet de représenter des mots sous la forme de vecteurs de nombres réels. L'hypothèse principale de cette méthode de représentation c'est de prendre en compte le contexte dans lequel le mot a été trouvé, c'est à dire les mots avec lesquels il est souvent utilisé.

La particularité d'une telle méthode, c'est qu'on va représenter des mots apparaissant dans des contextes similaires par des vecteurs relativement proches. Il est même possible de retrouver certaines régularités linguistiques en effectuant simplement des translations linéaires dans ce nouvel espace de représentation. Par exemple :

$$vec("Paris") - vec("France") + vec("Pologne") \approx vect("Varsovie") \quad (7)$$

L'intérêt de cette représentation est de pouvoir réduire le nombre de dimensions utilisées pour représenter les mots, afin de faciliter les tâches d'apprentissage impliquant ces mots. On peut ensuite se servir de nos vecteurs caractéristiques des mots pour alimenter des algorithmes, comme celui du SVM dans le cas de la reconnaissance d'entités nommées fourni par le MIT.

Machine à vecteurs de support Les Machines à Vecteurs de Support (SVM) [16] sont un ensemble de techniques d'apprentissage supervisé destinés à résoudre des problèmes de régression ou de classification. Le SVM appartient à la famille des classificateurs linéaires dont le but est de retrouver la frontière optimale qui sépare deux catégories d'objet. L'intérêt de trouver une telle frontière c'est de pouvoir prédire la catégorie d'un objet dont on ne connaît que la position sur le plan.

L'algorithme commence par estimer l'emplacement le plus plausible de cette frontière, appelée hyperplan, à partir de données d'entraînement. Pour obtenir la meilleure classification possible, on va chercher à maximiser la marge, c'est à dire la distance minimale entre les données d'entraînement et l'hyperplan séparateur. Si on se place dans $\mathbb{R}^n$ muni de la base $(e_1, ..., e_n)$, on calculera un hyperplan vectorielle d'équation :

$$w_1.e_1 + ... + w_n.e_n = 0 \text{ ainsi qu'un scalaire b} \quad (8)$$

avec w le vecteur de poids
Étant donné un ensemble de point x_k et leur label associé l_k, la marge est donnée par :

$$\min_k(\frac{l_k(w^T.x_k + b)}{||w||}) \quad (9)$$

On cherche w et b tel qu'ils maximisent la marge :

$$\underset{(w,b)}{\text{argmax}} \min_k(\frac{l_k(w^T.x_k + b)}{||w||} \quad (10)$$

[15] Le plongement des mots en français
[16] Support Vector Machine en anglais

```
O          0       38      O       PRO     il      x       x       0
O          0       39      O       V       prit    x       x       0
O          0       40      O       DET     sa      x       x       0
O          0       41      O       N       course  x       x       0
O          0       42      O       PUNC    ,       x       x       0
O          0       43      O       V       galopa  x       x       0
O          0       44      O       V       dégingandé      x       x       0
O          0       45      O       PUNC    ,       x       x       0
O          0       46      O       V       éperdu  x       x       0
O          0       47      O       PUNC    ,       x       x       0
O          0       48      O       DET     le      x       x       0
O          0       49      O       A       long    x       x       0
O          0       50      O       PREPDET du      x       x       0
B-STREET   0       51      O       N       quai    x       x       0
I-STREET   0       52      O       PREP    de      x       x       0
I-STREET   0       53      O       DET     la      x       x       0
I-STREET   0       54      O       NPr     Grève   x       x       0
O          0       55      O       PUNC    .       x       x       0
```

Fig. 9: Extrait du fichier d'entrée de CogComp pour le roman "L'Oeuvre"

On peut résoudre ce problème d'optimisation en utilisant les multiplicateurs de Lagrange. Une fois entraîné, c'est à dire les paramètres w et b trouvés on peut classer une entrée x en regardant le résultat de $h(x) = w^T.x + b$

4.3 CogComp

Le troisième outil[17] utilisé pour apprendre à reconnaître nos entités nommées de lieux a été développé par le groupe de calcul cognitif de l'université de Pennsylvanie [12]. Pour entraîner un modèle avec cet outil, on doit d'abord créer un fichier selon le bon format. C'est un format qui ressemble beaucoup à celui de Stanford-NLP : chaque mot est écrit sur une ligne avec son label au format BIO, son lemme, son type grammatical et sa position, et une ligne vide marque un changement de phrase(Fig. 9).

CogComp utilise un "averaged Perceptron" pour apprendre à reconnaître des entités.

Averaged Perceptron Le perceptron[1] est un algorithme d'apprentissage supervisé qui permet de décider si une entrée, représentée par un vecteur de nombres, appartient à une classe. C'est un cas particulier de réseau de neurone normalement constitué d'une couche de neurone portant le signal d'entrée, d'une couche de sortie indiquant la prédiction faite par le réseau et d'une ou plusieurs couches cachées. Chaque neurone du réseau est relié aux neurones des couches voisines par des liaisons auxquelles des poids sont attribués. L'entraînement du perceptron se fait en rétro-propageant les erreurs de classification sur les différents poids du réseaux. Autrement dit, à chaque fois que le perceptron se

[17] https://github.com/CogComp/cogcomp-nlp/tree/master/ner

trompe, il calcule son erreur et ajuste les poids de chaque couche en faisant "remonter" l'erreur jusqu'à la couche d'entrées.

Le problème avec ce type de perceptron est qu'il apprend mieux sur ses dernières entrées que sur les premières. En effet s'il avait classé correctement 999 exemples et qu'il se trompe sur le dernier exemple, il va modifier les poids qui avaient pourtant bien performés sur 99% des données. Une variante du perceptron appelé "averaged perceptron" permet de corriger ce problème et d'obtenir ainsi une meilleure généralisation. Pour chaque vecteur de poids, on va noter son temps de survie, c'est à dire le nombre d'itérations que le l'algorithme a effectué sans faire d'erreur de classification. Lors de la phase de test, on se servira de ces informations (11) pour faire les prédictions contrairement à l'algorithme classique du Perceptron (12) :

$$\hat{y} = sign(\sum_{k=1}^{K} c^{(k)}(w^{(k)}.\hat{x} + b^{(k)})) \tag{11}$$

avec $\hat{x}$ le vecteur en entrée, $w^{(k)}$ le k^{eme} vecteur de poids, $c^{(k)}$ la variable du temps de survie du k^{eme} vecteur de poids et $b^{(k)}$ le k^{eme} biais

$$\hat{y} = sign(w.\hat{x} + b) \tag{12}$$

avec $\hat{x}$ le vecteur en entrée, w le vecteur de poids et b le biais

4.4 Expérimentations et Résultats

Dans cette partie, nous allons évaluer les différents systèmes de reconnaissance d'entités nommées sur notre corpus de romans annoté automatiquement. Nous entraînerons plusieurs modèles en faisant varier la taille du corpus d'apprentissage, puis nous évaluerons leur performances.
Pour cela, nous allons comparer les annotations automatiques de Perdido avec les annotations faites par les différents modèles. Pour chaque entité, nous distinguerons 3 types d'erreur :

- une entité est annotée par le NER mais pas par Perdido
- une entité est annotée par Perdido mais pas par le NER
- une entité est annotée par Perdido mais les balises de l'annotation du NER sont mal placées

Nous utiliserons une variante du slot error rate [6] qui permet de prendre en compte ces différentes erreurs dans l'évaluation. Nous aurons donc, pour chaque modèle, les mesures suivantes :

- le *slot error rate*(SER) : le taux d'erreur total.
- le rappel : la proportion d'entités correctement annotées par rapport à toutes les entités qu'il fallait annoter.
- la précision : la proportion d'entités correctement annotées sur toutes les entités annotées

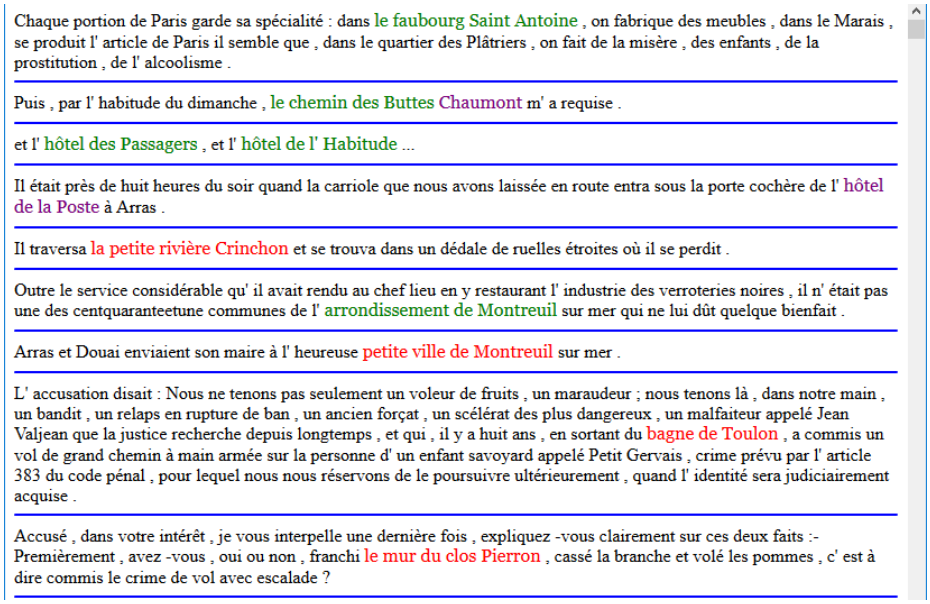

Fig. 10: Extrait du fichier html de résultat pour la reconnaissance des ENE de type "Others" avec un modèle Stanford

– la précision du balisage : une mesure de précision du balisage

L'évaluation des performances se fait automatiquement en comparant les entités identifiées par le NER avec celles qui étaient attendues mais il est aussi possible de les comparer manuellement. Pour cela, on dispose d'un fichier html dans lequel toutes les entités à annoter ou annotées sont présentes (Fig. 10). Dans ce fichier, les mots qui ont été correctement annotés sont en vert, les mots qui n'ont pas été correctement annotés sont en rouge et les mots qui ont été annotés par le NER mais pas par Perdido sont en violet.

Dans un premier temps, nous évaluerons les modèles en les entraînant sur une partie du corpus de romans annotée automatiquement par Perdido , puis en les testant sur la partie restante du corpus.
Puis nous testerons ces mêmes modèles sur deux textes annotés automatiquement par Perdido puis corrigés manuellement. Enfin nous entraînerons les modèles à partir du corpus annoté automatiquement en supprimant un type d'entité de l'apprentissage.

Le tableau 7 décrit les performances de Stanford-NLP, CogComp, et MITIE dans la reconnaissance d'entités nommées étendues de lieux de type "street" en prenant de 30 à 70% du corpus de romans pour l'apprentissage. Le nombre d'entités à annoter est donné en fonction de la taille du corpus de test. On re-

marque que Stanford obtient les meilleures performances avec presque 96% de rappel et un faible taux d'erreur (SER ≈ 8%). CogComp atteint les 90% de rappel mais il semble moins précis (SER ≈ 15%). MITIE est beaucoup moins performant avec presque 65% de rappel et un taux d'erreur de plus de 40% . Stanford semble donc le plus adapté pour reconnaître les entités nommées étendues de lieux de type "street" .

		Slot Error Rate			Rappel			Précision			Précision balisage		
Taux apprentissage	Nombre d'entités	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE
30	2364	15.69	21.62	59.75	89.59	86.63	64.74	98.65	94.95	77.05	89.74	86.97	64.67
40	2097	11.45	19.22	48.26	93.42	88.32	67.54	98.69	96.11	85.88	91.03	87.49	73.94
50	1459	8.95	16.00	56.79	95.75	90.88	63.13	98.52	96.51	80.51	91.81	88.86	68.71
60	1177	8.46	16.19	48.26	95.75	89.72	58.62	98.86	97.51	93.88	92.45	89.66	84.08
70	1039	8.86	15.88	43.21	95.66	90.47	63.72	98.81	97.61	95.39	91.84	88.68	83.86

Table 7: Performance de Stanford-NLP, CogComp et MITIE dans la reconnaissance des entités nommées étendues de lieux de type "street"

Le tableau 8 décrit les performances de Stanford-NLP, CogComp, et MITIE dans la reconnaissance d'entités nommées étendues de lieux de type "others" en prenant de 30 à 70% du corpus de romans pour l'apprentissage. Sur ce type d'entités les résultats sont moins bons que sur les "street". En effet, Stanford obtient les meilleures performances avec presque 75% de rappel et un taux d'erreur moyen (SER ≈ 30%). CogComp atteint les 50% de rappel mais il semble aussi moins précis (SER ≈ 65%). MITIE est beaucoup moins performant avec à peine 10% de rappel dans le meilleur des cas et un taux d'erreur de plus de 90% . La baisse de performance par rapport aux entités de type "street" s'explique par la faible complexité des entités de type "street" comparées aux entités de type "others". Les ENE de type "street" sont souvent sous la forme suivante : *feature préposition nomPropre* ; les ENE de type "others", à l'inverse, ont une structure moins stable et par conséquent moins facile à identifier.

Le tableau 9 décrit les performances de Stanford-NLP, CogComp, et MITIE dans la reconnaissance d'entités nommées étendues de lieux, sans distinction sur son type, en prenant de 30 à 70% du corpus de romans pour l'apprentissage. On observe encore de bons résultats pour Stanford avec presque 90% de rappel. CogComp approche les 80% et MITIE retrouve seulement une ESNE sur deux. Les résultats pour MITIE semble plutôt mauvais, cependant on n'observe pas une nette amélioration des résultats en augmentant la taille du corpus d'apprentissage. Le problème semble venir du procédé de Word Embedding : le corpus de 30 romans est probablement trop petit pour réellement apprendre la structure de la langue française.
Concernant Stanford et CogComp, les résultats semblent bons. Malgré cela, nous

		Slot Error Rate			Rappel			Précision			Précision balisage		
Taux apprentissage	Nombre d'entités	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE
30	1475	53.83	77.47	93.33	50.07	30.01	10.50	96.22	84.06	84.70	88.79	74.00	53.55
40	1278	48.98	73.42	94.44	55.13	31.98	8.14	96.57	90.29	86.67	89.03	79.25	58.33
50	900	36.17	61.22	97.06	67.78	53.67	3.67	96.52	81.31	94.29	92.25	73.57	68.57
60	666	29.35	62.01	95.19	74.78	51.95	6.46	96.33	81.22	93.48	93.04	75.12	58.70
70	517	29.98	65.86	93.71	74.08	50.87	8.51	96.47	78.27	83.02	92.95	70.24	73.58

Table 8: Performance de Stanford-NLP, CogComp et MITIE dans la reconnaissance des entités nommées étendues de lieux de type "others"

ne pouvons pas affirmer que l'un de ces systèmes est efficace et généralise bien car ces tests de performance ont été effectués sur des fichiers annotés par Perdido et donc qui contiennent des erreurs.

		Slot Error Rate			Rappel			Précision			Précision balisage		
Taux apprentissage	Nombre d'entités	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE	Stanford	CogComp	MITIE
30	3712	29.88	44.46	73.07	77.23	69.42	49.58	97.58	90.74	75.05	84.44	72.98	56.38
40	3265	25.21	41.04	64.13	81.19	72.10	52.57	98.18	92.97	82.55	86.33	73.14	65
50	2276	19.64	38.05	75.29	86.38	73.95	45.78	97.57	93.76	74.27	88.83	75.82	57.38
60	1782	17.26	35.80	57.24	88.38	77.27	52.36	97.46	92.48	93.39	90.10	76.23	72.37
70	1509	16.47	35.39	56.33	89.00	78.53	53.02	97.74	92.36	93.57	90.25	74.90	73.45

Table 9: Performance de Stanford-NLP, CogComp et MITIE dans la reconnaissance des entités nommées étendues de lieux

Nous allons donc réaliser une deuxième série de test sur Stanford et CogComp dans laquelle nous allons utiliser nos modèles sur deux romans annotés par Perdido (L'Oeuvre de Zola et Éducation sentimentale de Flaubert), puis sur une version corrigée manuellement de ces deux mêmes romans. On devrait observer de meilleurs résultats sur les versions corrigées si le modèle généralise bien.

		Slot Error Rate	
NER	Taux apprentissage	Perdido	Corrigé
Stanford	70	12.50	18.41
CogComp	70	27.9	33.44

Table 10: Performance de Stanford-NLP dans la reconnaissance des entités nommées étendues de lieux sur deux textes annotés (477 ESNE) par Perdido et sur ces deux textes annotés par Perdido et corrigés manuellement

On voit à travers ce tableau de résultats (Table 10) que les deux systèmes NER qui s'entraînent sur des entrées bruitées n'arrivent pas à bien généraliser. En effet, avec la version bruitée du test, on obtient un taux d'erreur plus faible qu'avec la version corrigée. Cela signifie que nos deux systèmes ne parviennent pas à corriger les erreurs d'annotation du Perdido.

Nous allons maintenant réaliser un dernier test dans lequel nous comparerons les résultats de deux modèles. Nous donnerons au premier modèle, une base d'apprentissage à laquelle nous enlèverons les occurrences des entités constituées du *feature* "quai". Nous donnerons au deuxième modèle la même base d'apprentissage que le premier modèle sans supprimer d'entités. Pour chacun de ces modèles, nous compterons le nombre d'entités construites avec le *feature* "quai" qu'ils ont réussies à identifier . Nous réaliserons ces tests sur les deux fichiers annotés par Perdido et corrigés manuellement et qui contiennent 43 occurrences d'entités "quai" parmi lesquelles 27 sont exclusifs au test.

	Nombre ESNE "quai" détectées		Nombre ESNE "quai" détectées exclusifs au test	
NER	0 exemple	plusieurs exemples	0 exemple	plusieurs exemples
Stanford	0	42	0	26
CogComp	0	43	0	27

Table 11: Comparaison du nombre d'entités, détectées dans le corpus corrigé, dont le *feature* est "quai" en fonction du nombre d'entités "quai" vue dans l'apprentissage pour un taux d'apprentissage de 70 %

D'après la table 11, nous voyons que, sans exemple dans l'apprentissage, les deux modèles ne retrouvent aucune des entités recherchées. Avec plusieurs exemples en entrée, on voit que les deux systèmes retrouvent toutes les entités recherchées et même des entités qu'ils n'avaient jamais vues, ce qui signifie qu'ils ont été capables de généraliser à partir de plusieurs exemples.

5 Conclusion

Dans ce projet, nous avons étudié plusieurs techniques pour améliorer une plateforme d'extraction et d'analyse d'entités nommées étendues de lieux.
Dans un premier temps, des essais ont été faits pour trouver une méthode correcte pour désambiguïser les entités, c'est à dire, une méthode qui retrouve les lieux auxquels les entités font référence sans faire trop d'erreurs. Nous avons pu remarquer que chaque méthode n'était pas adaptée à tous les types de document. A partir de ce constat et de notre corpus de texte, nous avons créé une méthode qui fonctionne bien sur des romans.
Finalement, nous avons essayé plusieurs méthodes de reconnaissance d'entités nommées, parmi les plus performantes dans la reconnaissance d'entités nommées de lieux, en les entraînant sur des entités nommées étendues de lieux. Les

résultats de ces observations nous ont montré que des systèmes de reconnaissance d'entités nommées pouvaient obtenir des résultats encourageants dans la détection d'entités nommées étendues de lieux sur un corpus de romans mais qu'ils parvenaient difficilement à généraliser. Sur ce dernier point, nous avons pu observer l'importance du *feature* lors de la détection d'entités. En effet, il apparaît qu'en supprimant toutes les occurrences d'un des *feature* à l'apprentissage, le modèle ne parvient pas à retrouver ces mêmes occurrences dans le test en généralisant à partir d'autres *features*. Néanmoins si le modèle dispose de plusieurs occurrences d'une entité composée de ce *feature*, il est capable de retrouver d'autres exemples d'entités composées de ce *feature* qu'il n'a jamais vu dans le test.Il semble donc possible d'enrichir le système Perdido avec des techniques d'apprentissage supervisé.
L'automatisation du processus de géocodage rend possible l'analyse et la représentation cartographique de documents textuels. Il est désormais possible de suivre le parcours d'un personnage dans les rues de Paris. Les travaux sur les méthodes d'apprentissage supervisé ont mis en évidence la difficulté de ces méthodes à généraliser sur des entités nommées étendues, mais aussi le grand intérêt que peuvent avoir ces méthodes. Il apparaît possible d'utiliser ces techniques, dans de futurs travaux, pour permettre au système de traiter d'autres types de documents ou d'autres langages.

References

1. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. Machine Learning 37(3), 277–296 (Dec 1999), https://doi.org/10.1023/A:1007662407062
2. Friburger, N., Maurel, D.: Finite-state transducer cascades to extract named entities in texts. Theoretical Computer Science 313(1), 93–104 (2004)
3. Gaio, M., Moncla, L.: Extended Named Entity Recognition Using Finite-State Transducers: An Application to Place Names. In: 9th International Conference on Advanced Geographic Information Systems, Applications, and Services. Nice, France (2017)
4. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. pp. 282–289. ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001), http://dl.acm.org/citation.cfm?id=645530.655813
5. Leidner, J.L.: Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding. SIGIR Forum 41(2), 124–126 (Dec 2007)
6. Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R.: Performance measures for information extraction. In: In Proceedings of DARPA Broadcast News Workshop. pp. 249–252 (1999)
7. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014), http://www.aclweb.org/anthology/P/P14/P14-5010

8. Moncla, L., Gaio, M.: Services web pour l'annotation sémantique d'information spatiale à partir de corpus textuels. Revue Internationale de Géomatique 28(4), 439–459 (2018)
9. Moncla, L., Gaio, M., Joliveau, T., Lay, Y.F.L., Boeglin, N., Mazagol, P.O.: Mapping urban fingerprints of odonyms automatically extracted from french novels. International Journal of Geographical Information Science 0(0), 1–21 (2019)
10. Moncla, L., Gaio, M., Nogueras-Iso, J., Mustière, S.: Reconstruction of itineraries from annotated text with an informed spanning tree algorithm. International Journal of Geographical Information Science 30(2) (2016)
11. Murrieta-Flores, P., Gregory, I.: Further Frontiers in GIS: Extending Spatial Analysis to Textual Sources in Archaeology. Open Archaeology 1(1) (2015)
12. Ratinov, L., Roth, D.: Design challenges and misconceptions in named entity recognition. In: CoNLL (6 2009), http://cogcomp.org/papers/RatinovRo09.pdf