Présentation du corpus de travail

Table des matières

[Introduction 1](#_Toc511078584)

[I. Les données brutes 1](#_Toc511078585)

[I.1 Obtention des données brutes auprès de HAL 1](#_Toc511078586)

[I.2 Chargement en mémoire et premières requêtes 2](#_Toc511078587)

[I.3 Constitution d’un lexique 3](#_Toc511078588)

[I.4 Suppression des titres en anglais 3](#_Toc511078589)

[I.5 Sérialisation 3](#_Toc511078590)

[I.6 Segmentation et étiquetage 4](#_Toc511078591)

[II. Le corpus de travail 4](#_Toc511078592)

[II.1 Format du fichier XML 4](#_Toc511078593)

[II.2 Chargement du corpus en mémoire 5](#_Toc511078594)

[II.3 Première requête 6](#_Toc511078595)

[Annexe A : informations sur le code 6](#_Toc511078596)

[A.1 Le dépôt 6](#_Toc511078597)

[A.2 Les bibliothèques 7](#_Toc511078598)

[Annexe B : interrogations et projections 7](#_Toc511078599)

# Introduction

Ce document présente les étapes nécessaires à l’obtention du corpus de travail, en donnant à chaque fois le but des opérations effectuées avec le minimum nécessaire de détails techniques. L’annexe A approfondit ceux-ci. Partant des données brutes obtenues à partir de l’archive en ligne HAL[[1]](#footnote-1), nous aboutissons à un corpus de travail sur lequel notre futur mémoire se basera. Sur notre méthode, nous utilisons deux types de requêtes. Celles d’exploration nous permettent d’obtenir rapidement un résultat sur une facette de nos données brutes ou de notre corpus. Intégrées à un cycle itératif rapide, elles permettent de passer très rapidement d’une hypothèse ou d’une question à des réponses et de rebondir sur ceux-ci. Les requêtes de traitement sont plus longues et s’envisagent après réflexion. Leurs résultats sont sauvegardés pour ne pas devoir les refaire à chaque fois. Nous alternons bien sûr les deux dans notre cheminement des données brutes à notre corpus de travail.

# I. Les données brutes

## I.1 Obtention des données brutes auprès de HAL

HAL permet la récupération des notices de ses documents au format JSON. Le premier programme conçu récupère celles-ci par lots de 100 et sauvegarde directement chaque lot reçu sur le disque dur dans un fichier propre. Ces fichiers sont encodés avec l’encodage utf-8, le seul que nous utiliserons dans notre travail. Le programme obtient ainsi 3 046 fichiers, soit 304 600 notices, en plusieurs heures. Ces données brutes prennent 190 Mo et seulement 26 Mo une fois compressées au format zip.

## I.2 Chargement en mémoire et premières requêtes

Lire l’ensemble des fichiers et les transformer en données en mémoire est une chose assez simple dans ce cas-là car l’algorithme procède par lots de 100. Une fois les données en mémoire, une salve de requêtes d’interrogation est lancée pour essayer de dessiner leurs contours et leurs points saillants.

Ces premières requêtes nous permettent de commencer à travailler les données brutes et à écarter certaines d’entre elles pour les raisons suivantes :

1. Simplifications de certaines notices ayant plusieurs titres en n'en prenant plus qu'un.   
   Toutes celles dans ce cas vérifiées à la main sont des notices avec 2 titres dont le second est une traduction en anglais du premier qui est en français. Après ce traitement, nous avons donc à présent 1 notice = 1 titre.
2. Vérification sur la longueur en caractère des titres : les 37 notices avec des titres inférieurs à 4 caractères sont gardées car elles sont diluées dans la masse.
3. Suppression de certaines notices et donc du titre associé pour d'autres raisons :

* 33 car elles n'ont pas d'auteurs
* 6448 car elles n'ont pas de domaines associés
* 1 car elle n'a pas de type de document associé

On se retrouve alors avec un corpus de 298 118 titres, 194 378 auteurs et 19 domaines.

Sur les titres des notices conservées, on compte le premier mot de chaque titre et le premier mot après chaque « : ». On calcule aussi la présence de caractères spéciaux et on en tire les enseignements suivants :

1. Les notices ont toutes pour valeur de leur attribut langue le français. Néanmoins, avec cette métadonnée, on ne sait pas si c'est le titre qui est en français ou si c'est le document référencé par la notice. La bibliothèque langdetect permet de vérifier la véritable langue du titre, mais son temps de traitement est très long. Nous prenons pour l'instant comme hypothèse que si le document référencé par la notice est en français, le titre l'est aussi.
2. Si la plage des années est assez grande (de 1921 à 2018), la surreprésentation de 2018 (63%) ne permettra pas d'études diachroniques représentatives.
3. Si on réduit notre corpus aux articles, communications, chapitres d'ouvrage, thèses et mémoires, on couvre 261 985 titres, soit 87% de notre corpus total mais nous choisissons de ne pas nous restreindre à ceux-ci pour l’instant.
4. Nous avons une moyenne pondérée de 87 caractères par titre.
5. Nous avons une moyenne pondérée de 13 mots par titre.
6. 86% des notices concernent la SHS.
7. 31% des titres comportent un « : ».

La segmentation en mots a été faite à l’aide d’un algorithme simpliste pour ces premières étapes. C’est à ce moment que j’ai envoyé mon premier fichier de résultats au format Excel. Ce fichier a été produit par programmation. L’ensemble des résultats d’une requête est enregistré dans un onglet spécifique, permettant une consultation simple et de poursuivre le travail sur ces données avec un tableur après. Plus tard est venu le besoin de construire un lexique des mots utilisés.

## I.3 Constitution d’un lexique

La constitution du lexique de tous les mots utilisés dans les titres est partie d’une volonté de mieux percevoir le contenu de ceux-ci. Pour chaque mot, nous comptons son nombre d’occurrences, en ne faisant pas de distinction entre les majuscules et les minuscules. Les mots les plus fréquents sont les articles définis et indéfinis. Les deux premiers noms à apparaître sont « étude » et « analyse » ce qui est tout à fait logique au vu de l’origine de nos données brutes. Le troisième nom le plus fréquent est un nom propre, France, ce qui tend à faire penser que de nombreux sujets d’études s’élaborent dans ce cadre. L’ensemble des résultats se trouvent dans le fichier :

<https://github.com/Xitog/tal/raw/master/output_lexicon/lexicon.xlsx>

Mais il s’agit de la version corrigée du lexique, car cette étape a également révélé un problème dans les données brutes.

## I.4 Suppression des titres en anglais

Les premières versions du fichier Excel produit contenant le lexique faisaient état d’une fréquence importante des articles du français mais aussi de mots suspects comme 'on', 'in' ou 'a', pouvant être utilisés aussi bien en français qu’en anglais mais également de mots très fréquents en anglais comme 'and', 'the', ou en allemand comme 'und'.

Après des vérifications manuelles, il apparaît qu’il existe des titres en anglais et des titres mixtes qui contiennent le titre français puis sa traduction en anglais, avec un séparateur entre les deux qui peut être '[' ou '/', ou encore 'Titre en anglais : '. Enfin il existe des titres en français mais qui utilisent quelques mots anglais pour une citation ou une référence.

L'hypothèse faite auparavant que, si la langue du document référencé par la notice était le français, alors son titre était en français, est donc fausse. Pour les filtrer efficacement, chaque titre a tout d’abord été noté. À chaque fois que l’algorithme de notation rencontre un mot énoncé précédemment, il ajoute un point à la note du titre. Cette mesure grossière sert juste à délimiter l'espace de traitement pour la seconde étape.

Sur les 18 071 titres dont la note était supérieure à 0, le test de reconnaissance du langage de la bibliothèque langdetect a été utilisé. Trop lent pour être effectué sur l’ensemble des données brutes, sur ce sous-ensemble cela n’a pris que 30 minutes. Le résultat n'est pas parfait, la langue de certains titres est mal identifiée, notamment dans les titres multilingues. Cela permet néanmoins de supprimer 12 206 titres supplémentaires non écrits en français, réduisant le nombre de titres sélectionnés à 285 913, ce qui reste un ensemble assez conséquent pour notre étude.

## I.5 Sérialisation

Les données brutes étant de plus en plus travaillées en mémoire, une fonctionnalité de sérialisation est ajoutée à notre code. Celle-ci permet de sauvegarder des données en mémoire sur un disque dur, pour une conservation pérenne.

Cette sauvegarde est produite sous deux formats JSON et XML, pour un contenu qui est sémantiquement identiques mais syntaxiquement différents. Ce dernier sera ensuite privilégié. Lors de la sauvegarde, il doit être fait attention à respecter le format pour produire un fichier valide. Les titres, et plus rarement les auteurs, peuvent comporter des caractères comme '<', '>' ou '&' qui font partie de la syntaxe XML. Pour éviter qu’ils soient interprétés de cette manière, ils sont échappés, c’est-à-dire remplacés par un caractère neutre. De la même façon pour le caractère guillemet double en JSON.

Le fichier XML fait 450 Mo et compressé au format zip il n’en fait plus que 40, ce qui est avantageux pour le transporter. Armé de cette fonctionnalité, nous pouvons à présent envisager des traitements très longs sur l’ensemble des titres. Nous sauvegarderons ensuite le résultat de ces traitements à l’aide de cette fonctionnalité pour ne pas avoir à les répéter à chaque fois.

## I.6 Segmentation et étiquetage

Le traitement long le plus important est une véritable segmentation des mots (*tokenization*) et un étiquetage en partie du discours (*pos-tagging*). Il existe beaucoup de logiciels et de bibliothèques pour effectuer ce travail. Nous avons écarté Polyglot[[2]](#footnote-2) et SpyCy[[3]](#footnote-3) car ils demandaient dans notre environnement Windows l’installation de Visual Studio, qui est une énorme application.

Treetagger[[4]](#footnote-4), TXM[[5]](#footnote-5) et NTLK[[6]](#footnote-6) ont été ignorés car j’ai trouvé directement un outil similaire qui dispose de plusieurs interfaces avec Python et ne demandait que Java 8 : il s’agit de Stanford Core NLP[[7]](#footnote-7). Disponible sous la forme d’un serveur web qui reçoit des requêtes et renvoie les réponses qui conviennent, il dispose pour le français des fonctions de segmentation des mots et de catégorisations en partie du discours (appelées *annotators* dans le vocabulaire du logiciel). Notons que Stanford Core NLP ne gère pas l’association d’un mot à son lemme (*lemmatization*) pour le français.

Ces traitements ont donc été effectués sur le corpus en mémoire. Cela a pris trois heures. Le résultat a été sauvegardé pour constituer le corpus de travail.

# II. Le corpus de travail

Nous avons à présent notre corpus de travail sous la forme d’un fichier. Nous devons à présent le charger en mémoire dans un modèle de données pour effectuer des requêtes rapides dessus.

## II.1 Format du fichier XML

Le fichier sérialisé au format XML du corpus de travail a la structure explicitée par l’exemple ci-dessous. Les parties <words> et <pos\_tags> sont issues du travail de Stanford Core NLP, de la segmentation des mots et de l’étiquetage en catégorie du discours respectivement.

<notices>

<notice>

<id>**1712921**</id>

<type>**ART**</type>

<date>**2018**</date>

<title>**La logique de l'action de Michael Quante**</title>

<words>

<word>**La**</word>

<word>**logique**</word>

<word>**de**</word>

<word>**l'**</word>

<word>**action**</word>

<word>**de**</word>

<word>**Michael**</word>

<word>**Quante**</word>

</words>

<pos\_tags>

<pos\_tag>**DET**</pos\_tag>

<pos\_tag>**NOUN**</pos\_tag>

<pos\_tag>**ADP**</pos\_tag>

<pos\_tag>**DET**</pos\_tag>

<pos\_tag>**NOUN**</pos\_tag>

<pos\_tag>**ADP**</pos\_tag>

<pos\_tag>**PROPN**</pos\_tag>

<pos\_tag>**ADJ**</pos\_tag>

</pos\_tags>

<authors>

<author>**Alain Patrick Olivier**</author>

</authors>

<domains>

<domain>**0.shs**</domain>

<domain>**1.shs.phil**</domain>

</domains>

</notice>  
 <!-- notices suivantes -->  
</notices>

Le fichier comporte 285 913 notices et donc autant de titres.

## II.2 Chargement du corpus en mémoire

Du fait de la taille du fichier, un chargement progressif en mémoire est effectué à l’aide de la bibliothèque standard Python. Notre modèle de données est pour l’instant simplifié à cette étape : nous avons un corpus stocké dans la structure de données *Corpus* qui contient les titres, chacun stocké dans la structure de données *Title*. Cette étape est appelée désérialisation, c’est l’inverse de la sérialisation. Les caractères échappés sont restitués normalement.

Nous reprenons ci-dessous la structure d’un titre en mémoire :

|  |  |  |
| --- | --- | --- |
| **Nom attribut** | **Type Attribut** | **Signification** |
| title | Chaîne de caractères | Le titre en lui-même |
| words | Liste de chaîne de caractères | Liste des mots |
| pos\_tags | Liste de chaîne de caractères fixées par Stanford Core NLP | Liste des étiquettes de catégorie du discours |
| authors | Liste de chaîne de caractères | Liste des auteurs |
| docid | Entier | Identifiant de la notice |
| kind | Chaîne de caractères fixées par HAL | Type du document référencé |
| date | Chaîne de caractères | Année de dernière modification |

*Les attributs de la structure de données « Title » en mémoire*

Cette structure est moins évoluée pour l’instant que celle utilisée pour les données brutes : pour celles-ci nous avions une liste d’objets « Authors » dans l’attribut authors, ceux-ci possédant des références vers leurs publications et non pas une simple liste de chaîne de caractères. Cette structure de données est amenée à évoluer pour avoir le même niveau de fonctionnalités que dans l’étape précédente. De même, certains calculs devraient être refaits, comme la moyenne de longueur en mots des titres ou le lexique des mots utilisés.

Une fois en mémoire, nous pouvons faire des requêtes avec un temps de traitement rapide, entre 1 et 2 minutes pour continuer notre exploration, avant d’envisager éventuellement d’autres traitements plus longs. Nous commençons par un premier travail autour du « : ».

## II.3 Première requête

Nous voulons savoir, dans une fenêtre de 4 mots après le « : », quels mots il y a et quels sont leurs catégories du discours. Pour cela, il suffit de parcourir le corpus en mémoire, et compter les différents quatuors que nous rencontrons, pour les mots et les étiquettes. Ci-dessous ce trouve le tableau des résultats. La première ligne signifie que 68% des titres de notre corpus ne possède pas de « : ». Cela signifie que 32% en possède (31% pour les données brutes, l’élimination de certains titres expliquant cet écart). La valeur \_\_EMPTY\_\_ signifie qu’il n’y a pas de mots : soit le titre n’a pas de « : », soit le titre possède moins de 4 mots après le « : ». Le premier pourcentage se fonde sur la totalité des titres, le second à celui des titres avec « : ». Ne sont affichés que ceux ayant pour ce dernier pourcentage une valeur de 2% ou plus. Cela permet de voir les suites d’étiquettes les plus fréquentes après un « : ».

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **POSTAG1** | **POSTAG2** | **POSTAG3** | **POSTAG3** | **COUNT** | **% of total** | **% of titles with ":"** |
| \_\_EMPTY\_\_ | \_\_EMPTY\_\_ | \_\_EMPTY\_\_ | \_\_EMPTY\_\_ | 195846 | 68% |  |
| DET | NOUN | ADP | DET | 9028 | 3% | 10% |
| NOUN | ADP | DET | NOUN | 8348 | 3% | 9% |
| DET | NOUN | ADP | NOUN | 6106 | 2% | 7% |
| DET | NOUN | ADJ | ADP | 3522 | 1% | 4% |
| ADP | DET | NOUN | ADP | 2530 | 1% | 3% |
| NOUN | CONJ | NOUN | ADP | 2399 | 1% | 3% |
| NOUN | ADP | NOUN | ADP | 1830 | 1% | 2% |
| NOUN | ADJ | CONJ | NOUN | 1467 | 1% | 2% |
| DET | NOUN | ADP | PROPN | 1455 | 1% | 2% |
| DET | NOUN | ADJ | \_\_EMPTY\_\_ | 1359 | 0% | 2% |

*Résultats de la requête*

Le tableau ci-dessous explique la signification des différentes étiquettes rencontrées dans ces suites les plus fréquentes :

|  |  |
| --- | --- |
| DET | Déterminant |
| ADJ | Adjectif |
| ADP | Adposition (préposition ici) |
| CONJ | Conjonction de coordination |
| NOUN | Nom |
| PROPN | Nom propre |
| \_\_EMPTY\_\_ | Vide (pas d'élément) : ceci n’est pas une étiquette de Stanford Core NLP. |

*Légendes des étiquettes présentes*

# Annexe A : informations sur le code

## A.1 Le dépôt

Tout est le code Python est public et stocké sur la plateforme GitHub à cette adresse :

<https://github.com/Xitog/tal>

Il est composé de 3 fichiers qui correspondent chacun à une étape :

* corpus1.py : récupération des données brutes de HAL (correspond au chapitre I.1)
* corpus2.py : traitement des données brutes pour obtenir le corpus de travail (correspond aux chapitres I.2 à I.6).
* corpus3.py : traitement du corpus de travail (correspond au chapitre II).

corpus3.py utilise certains éléments de corpus2.py.

Le code n’a pas été contrôlé par un outil de mesure de qualité automatique comme PyLint.

<https://www.pylint.org/>

Ce dépôt contient également le corpus de travail au format zip :

<https://github.com/Xitog/tal/raw/master/output_dump_repo/dump.zip>

Ainsi que le lexique obtenu à partir des données brutes :

<https://github.com/Xitog/tal/blob/master/output_lexicon/lexicon.xlsx>

Et les différents documents de présentation et liaison.

## A.2 Les bibliothèques

Nous utilisons la bibliothèque openpyxl pour la gestion du format Excel XLSX. Celui comporte l’avantage par rapport au format Excel XLS de pouvoir dépasser plus de 65 535 lignes par onglet.

<https://openpyxl.readthedocs.io/en/stable/>

Elle est plus puissante que la première que j’avais utilisée, l’ensemble XLRD et XLWT qui gère le format XLS. Mais la documentation d’openpyxl est néanmoins plus complexe.

Pour le chargement progressif du XML, j’utilise la bibliothèque standard, plus particulièrement :

xml.etree.ElementTree

Pour la connexion avec Stanford Core NLP, j’utilise cette bibliothèque, qui permet depuis Python de choisir la langue cible ainsi que la quantité de mémoire :

<https://github.com/Lynten/stanford-corenlp>

Le site officiel de Stanford Core NLP est : <https://stanfordnlp.github.io/CoreNLP/>

La bibliothèque langdetect a également été utilisé pour détecter la langue des titres suspectés de n’être pas écrits en français.

# Annexe B : interrogations et projections

Dans cette partie, nous listons, en premier lieu pour nous même, des pistes de points à étudier plus avant :

* Calcul d’un écart-type pondéré
* Comment corrélé deux variables : nombre d’auteurs et longueurs en nombres de mots ?
* Construire une structure de données qui est un dictionnaire avec pour clé des suites d’étiquettes mais pour valeurs les différentes instances de ces suites d’étiquettes en mots. Cela permettra d’avoir des exemples pour les suites d’étiquettes, voire de grouper les exemples.
* Reconstruire des objets Author avec des références vers leurs publications
* De même pour les Domaines (sauf si inutile)
* Refaire le lexique
* Refaire la moyenne pondérée de la longueur en mots
* Obtenir le lemme pour chaque mot, grâce au pos-tag et à la base Lexique 3  
  <http://www.lexique.org/>
* Passer PyLint ?

1. <https://hal.archives-ouvertes.fr/> [↑](#footnote-ref-1)
2. <http://polyglot.readthedocs.io/en/latest/> [↑](#footnote-ref-2)
3. <https://spacy.io/> [↑](#footnote-ref-3)
4. <http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/> [↑](#footnote-ref-4)
5. <http://textometrie.ens-lyon.fr/> [↑](#footnote-ref-5)
6. <https://www.nltk.org/> [↑](#footnote-ref-6)
7. <https://stanfordnlp.github.io/CoreNLP/index.html> [↑](#footnote-ref-7)