Conception et réalisation d'une chaîne de traitement automatique des langues adaptée à des projets littéraires

Julien Fagot

Conception et réalisation d'une chaîne de traitement automatique des langues adaptée à des projets littéraires

FAGOT

Julien

Sous la direction de Olivier KRAIF

UFR LLASIC

Département Sciences du Langage

Section Industrie de la Langue

Rapport de stage de master 2 mention professionnelle - 20 crédits

Parcours : Industrie de la Langue

Année universitaire 2019-2020

Université Grenoble Alpes

Déclaration anti-plagiat
Document **à scanner** après signature
et **à intégrer** au mémoire électronique

DÉCLARATION

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

NOM : Fagot

PRENOM : Julien

DATE : 24 août 2020

[redacted]

Remerciements Arnaud Bey, pour avoir dirigé mon stage et m'avoir assisté durant tout mon stage.
Ainsi que **toute l'équipe d'ÉLAN**, pour m'avoir accueilli en tant que stagiaire et m'avoir aussi aidé durant toute la durée de mon stage.

Table des matières

Introduction

Ce rapport décrit mon stage effectué entre le 2 mars et le 27 juillet au sein de l'équipe ELAN de l'UMR Litt&Arts de l'Université Grenoble Alpes. Le but de ce stage était de concevoir une chaîne de traitement en TAL adaptée aux projets entrepris par l'équipe ELAN. Ce rapport décrit le cheminement que j'ai entrepris afin de satisfaire cette demande, d'abord en décrivant les conditions qui ont été spécifiées, puis en expliquant comment une solution a été conçue, quelles méthodes de travail ont été employées afin de mettre en place cette solution, et enfin, comment la phase de développement de la solution s'est déroulée.

Chapitre 1

Définition du besoin

1.1 Résumé de la demande initiale

Le but final de ce stage était de concevoir un outil capable d'appliquer plusieurs types de traitements textuels automatiques sur des textes tirés des projets sur lesquels travaille l'équipe de recherche ELAN. Il devait prendre la forme d'une chaîne de traitement automatique capable d'appliquer un ou plusieurs traitements textuels sur un ou plusieurs fichiers fournis comme entrées. Pour ce faire, il était nécessaire de concevoir une solution capable de traiter plusieurs formats d'entrées textuelles. Il était aussi nécessaire de s'assurer de la rendre aussi ergonomique que possible afin de faciliter toute éventuelle utilisation par des chercheurs qui ne seraient pas habitués à l'utilisation d'outils informatiques. Enfin, il était nécessaire d'écrire une documentation détaillée afin de permettre à tout utilisateur de comprendre le fonctionnement de l'outil et de le prendre en main rapidement. Cette documentation est reproduite dans l'Annexe A (Documentation de l'outil) de ce rapport.

1.2 Enjeux du projet

La réalisation de ce projet soulevait un certain nombre d'enjeux techniques ainsi que linguistiques. L'enjeu principal était la simplification de l'utilisation de programmes qui peuvent parfois être assez difficiles à utili-

ser pour toute personne non formée à l'utilisation de l'outil informatique. Ceci aurait pour but final d'exposer ces outils à un public scientifique plus vaste, notamment dans le domaine de la linguistique et de la littérature. De plus, réunir l'analyse morphosyntaxique et l'analyse textométrique en un seul outil pourrait permettre de faciliter la réalisation de projets ultérieurs qui nécessiteraient l'utilisation de ces deux traitements, notamment dans les projets en littérature, qui nécessitent des outils d'analyse de textes avancés.

Faciliter l'utilisation de ces différents programmes possède aussi des enjeux linguistiques. En effet, l'étiquetage morphosyntaxique permet d'extraire des informations sur la structure du texte qui sont utiles à d'autres projets de recherche qui se concentrent sur des textes littéraires. Par exemple, il est utile pour analyser le style d'un auteur et voir quels effets ce style a sur la structure morphosyntaxique du texte. De même, l'analyse textométrique permet d'effectuer un grand nombre de traitements statistiques qui sont eux aussi très utiles pour l'analyse de textes littéraires, par exemple en comparant la taille du vocabulaire de plusieurs textes. Faciliter l'accès à ces programmes permettrait donc de faciliter la recherche sur les textes littéraires.

Ce projet soulève aussi des enjeux techniques en faisant appel à de nombreux programmes de TAL. Ces enjeux sont ceux de la tokénisation, de la lemmatisation et de l'étiquetage morphosyntaxique. En effet, ce projet avait pour but de créer un outil capable d'appliquer les trois étapes de l'annotation morphosyntaxique, ce qui veut dire que la question de ces trois tâches avait une place importante dans le développement du projet. Néanmoins, le but principal de ce projet n'était pas la création d'un nouveau système capable d'effectuer ces tâches, donc je me suis concentré sur l'évaluation d'outils existants qui possédait déjà la capacité d'appliquer ces trois types de traitement ; je me suis cependant assuré, lors de l'évaluation des programmes déjà existants d'étiquetage morphosyntaxique, d'évaluer séparément ces trois tâches pour chacun des programmes.

Cet enjeu des trois tâches de l'étiquetage morphosyntaxique a aussi entraîné une autre question, celle de la normalisation des textes. En effet, il est possible que certaines irrégularités dans l'écriture d'un texte puissent fausser cette étape. Ce problème se rencontre le plus souvent dans le cas des

apostrophes, qui peuvent être différentes d'un texte à l'autre. Ce problème ne m'est pas apparu à temps lors du développement du projet, et n'a donc malheureusement pas pu être pris en compte comme il aurait dû l'être.

Enfin, le développement d'une solution de traitement du XML pourrait s'avérer très utile pour tout projet en linguistique qui nécessite du travail sur des fichiers déjà formatés en XML. En effet, le traitement du XML est un problème assez compliqué, où toute assistance pourrait être utile. Certaines solutions existaient déjà pour le traitement du XML, mais ce problème a quand même été une part importante du projet, puisque son but final était de traiter des fichiers TXT tout comme des fichiers XML.

être utile.

1.3 Choix techniques

L'outil à créer avait pour but d'être utile à l'équipe d'ELAN dans leurs projets. Par conséquent, l'équipe d'ELAN m'a fait plusieurs recommandations sur des programmes qui pourraient être utiles à la réalisation du projet, ainsi que sur des principes de développement qui pourraient être utiles. Les principales recommandations ont été de s'assurer que l'outil à concevoir fonctionnait sur le plus grand nombre de systèmes d'exploitation possible, dont le système d'exploitation libre Linux, et de tenter autant que possible d'utiliser des logiciels libres au lieu de logiciels fermés. De plus, afin de faciliter sa création, il était plus simple de concevoir le résultat final sous un seul langage de programmation. Le langage Python a été choisi pour remplir ce rôle, ce qui signifiait que les programmes que l'outil allait utiliser devaient être compatibles au moins partiellement avec Python.

1.4 Entrées fournies

1.4.1 Types d'entrées

Les entrées que l'outil devait être capable de traiter était des entrées textuelles extraites des projets sur lesquels travaille l'équipe ELAN. Ces entrées

s'agissaient donc principalement de textes littéraires écrits dans différentes langues : principalement en français, mais aussi parfois en latin, en grec antique ou bien en ancien français. Par conséquent, il devait être capable de traiter des textes littéraires non seulement en langue française, mais aussi dans chacune des langues qui apparaît dans les projets auxquels participe l'équipe ELAN. Il devait aussi être capable de traiter plusieurs fichiers en un seul lancement afin de permettre son utilisation sur un corpus de textes.

1.4.2 Formats d'entrée

L'outil devait être capable de traiter deux formats d'entrée différents. Il devait être capable de traiter du texte brut, fourni au format de texte *.txt*, mais aussi de traiter des fichiers de texte qui avaient déjà été pré-traités dans le cadre d'autres projets. Dans ce cas de figure, les fichiers à traiter étaient des fichiers qui avaient été retranscrits sous la forme de fichiers *.xml* selon le formalisme XML-TEI. Il était donc nécessaire que l'outil soit capable de traiter les deux formats de fichiers d'entrée afin de pouvoir être utile à l'équipe ELAN.

1.5 Sorties à produire

1.5.1 Traitements à appliquer

Le but de l'outil était d'appliquer deux types de traitements successifs sur les textes fournis comme entrées : d'abord, un étiquetage morphosyntaxique du texte, puis ensuite, une analyse textométrique du texte, dans le but d'obtenir des données statistiques sur la composition du texte. Ces données incluent des mesures comme le nombre de tokens ou la taille du vocabulaire du texte, c'est-à-dire le nombre de mots différents présents dans le texte. Il était nécessaire de s'assurer que ces deux traitements puissent être lancés l'un après l'autre, afin de faciliter leur utilisation en réduisant le nombre d'étapes qui seraient autrement nécessaires au lancement de ces deux traitements.

Annotation morphosyntaxique

Pour effectuer l'annotation morphosyntaxique, l'outil devait être capable de faire appel à un ou plusieurs étiqueteurs morphosyntaxiques, qui devaient tous être capable d'effectuer les trois traitements morphosyntaxiques principaux expliqués ci-dessous : La tokenisation du texte, la lemmatisation des tokens, et l'étiquetage morphosyntaxique des tokens.

La tokenisation d'un texte consiste en la séparation de chacune des unités lexicales du texte. Dans la majorité des cas, ces unités lexicales sont constituées par un seul mot, mais il existe des exceptions qui nécessitent un traitement différent, ce qui rend la tokenisation plus compliquée qu'une simple séparation en mots. L'étape suivante est la lemmatisation. Le but de cette étape est de déterminer le lemme, c'est-à-dire la racine de chacun des tokens qui ont été séparés auparavant. Enfin, l'étape d'étiquetage morphosyntaxique a pour objectif de déterminer la catégorie grammaticale à laquelle appartient chaque token.

Les résultats de cette opération peuvent être exploités tels quels sans traitement successif, mais ils servent aussi d'entrées pour le second type de traitement.

Traitement textométrique

Le second traitement à effectuer était une analyse textométrique, dont le but était d'extraire un grand nombre de mesures statistiques sur le texte à partir des données extraites lors de l'analyse morphosyntaxique effectuée au préalable, ainsi que de présenter ces mesures sous une forme facilement accessible à tout type d'utilisateur.

1.5.2 Format de sortie

L'outil devait être capable de produire différents types de sortie en fonction du besoin de l'utilisateur. Par exemple, pour rendre les résultats de l'analyse morphosyntaxique, l'outil devait être capable non seulement de produire les résultats sous la forme de texte brut, mais il était aussi nécessaire

qu'il soit capable de traiter des fichiers XML sans modifier leur structure, afin qu'il soient immédiatement utilisables dans les projets de l'équipe ELAN sans traitement supplémentaire ni conversion. Quant à l'analyse textométrique, il devait être capable de rendre les résultats sous une forme facilement accessible, mais le format de sortie n'était pas restreint, car la grande diversité des programmes de textométrie fait qu'on ne pouvait attendre un type de sortie particulier.

Chapitre 2

Méthodes de travail

Ce chapitre résume les méthodes de travail que j'ai utilisées durant le stage. Il comprend une courte description des outils utilisés au sein de l'équipe d'ELAN, ainsi qu'une description de l'organisation du travail, d'abord en présentiel, puis en télétravail.

2.1 Outils utilisés

2.1.1 Ubuntu

L'équipe ELAN utilise principalement des ordinateurs utilisant la distribution Ubuntu du système d'exploitation Linux, y compris sur le poste de travail qui m'a été prêté durant le stage. Par conséquent, une partie de mon stage a été consacrée à ma familiarisation avec l'utilisation de ce système d'exploitation, ce qui m'a permis de m'habituer à l'utilisation de nouveaux outils de développement tels que Docker auxquels je n'avais pas eu accès avant car Windows est un environnement qui se prête moins à l'utilisation de ces outils.

2.1.2 Python

Le langage de programmation Python a été choisi pour le développement de l'outil notamment pour s'assurer qu'il serait compatible avec le plus grand

nombre de systèmes d'exploitation possible. De plus, l'équipe d'ELAN et moi avons décidé qu'il serait plus simple d'utiliser un langage de programmation dans lequel j'avais déjà été formé afin de ne pas nécessiter une formation à un nouveau langage. La version utilisée pour le développement de l'outil était la version la plus récente au début de mon stage, à savoir la version 3.8.2.

2.1.3 Git

Git est un outil de gestion de version très utilisé dans le domaine informatique qui est aussi utilisé par l'équipe d'ELAN afin de gérer le code source de tous leurs projets, ainsi que pour collaborer autour de ces projets et les déployer. Par conséquent, il m'a semblé utile d'apprendre à utiliser cet outil afin de garder des traces de toutes les versions de l'outil, et de le mettre en ligne à l'aide de la version en ligne de Git offerte par le site Internet GitLab afin permettre au reste de l'équipe d'ELAN de le consulter et de le modifier facilement si nécessaire.

2.1.4 Docker

Docker est un outil qui permet de créer des environnements de travail isolés nommés conteneurs, qui sont construits à partir d'une liste d'instructions. Cet outil permet de créer un environnement isolé spécifique pour un programme, qui restera identique pour tout utilisateur tant qu'ils utilisent la même liste d'instructions. Dans le cas de ce projet, Docker a été très utile pour pouvoir créer un environnement de travail pour l'outil qui installe automatiquement les logiciels de traitement nécessaires à son bon fonctionnement, et qui les paramètre automatiquement sans nécessiter d'intervention de la part de l'utilisateur. De plus, le fait que l'environnement de travail crée par Docker reste identique sur tout système l'utilisant permet de développer l'outil sur la base d'une seule structure unifiée qui ne changera pas d'un utilisateur à l'autre, ce qui rend le résultat final plus stable. De même, Docker permet d'éviter tout conflit si l'utilisateur avait déjà installé une autre version de l'un des programmes auxquels le projet fait appel, ainsi que d'éviter d'éventuels problèmes de compatibilité entre les logiciels utilisés et l'ordinateur de

l'utilisateur. Cet outil n'a pas été utilisé depuis le début du projet, mais a été ajouté au projet durant son développement à la suite d'une évolution des objectifs du projet.

2.1.5 PHP

PHP est un langage de programmation orienté Web. Tout comme Docker, ce langage a été ajouté au projet plus tard suite à son évolution. Ce langage à été choisi car c'était le plus adapté à la création d'une interface Web pour l'outil, dans le but de faciliter son utilisation.

2.2 Organisation du travail

Au début du stage, et pendant environ un mois, j'ai pu travailler en coopération très étroite avec l'équipe d'ELAN, et ai donc pu recevoir leur assistance dans les cas où cela s'avérait nécessaire. De plus, des réunions hebdomadaires étaient organisées afin que tout les membres de l'équipe puissent rendre compte de la progression de leur tâche actuelle, ce qui me permettait de recevoir un retour de la part de l'équipe chaque semaine sur l'état actuel du projet. Néanmoins, l'organisation de mon stage a du être adaptée à partir de la mi-Mars en raison de la crise sanitaire. De ce fait, le fonctionnement de mon stage a été revu afin d'être adapté au télétravail. Bien que la communication entre les membres de l'équipe aie été rendue plus difficile, des moyens de communication à distance ont été rapidement mis en place non seulement afin de permettre la coopération une fois de plus, mais aussi pour continuer à organiser les réunions hebdomadaires comme avant la période de quarantaine.

Chapitre 3

Analyses préliminaires

Après m'être habitué aux méthodes et aux outils de travail de l'équipe ELAN, ma première tâche a été de mener deux analyses préliminaires des solutions de traitement de données textuelles déjà existantes qui pourraient éventuellement ensuite être adaptées à l'outil.

3.1 Analyse des outils morphosyntaxiques

Afin d'effectuer l'analyse morphosyntaxique, il est nécessaire d'utiliser un analyseur morphosyntaxique. La création d'un analyseur étant une tâche trop importante pour le stage, il était nécessaire d'utiliser une solution déjà existante. Le choix de ou des programmes à utiliser pouvait avoir un effet assez conséquent sur le fonctionnement de l'outil, donc il fallait choisir le ou lesquels utiliser avant de commencer sa phase de développement. Pour ce faire, j'ai constitué, avec l'aide de l'équipe d'ELAN, une liste de logiciels de traitement morphosyntaxique qui pourraient potentiellement être intégrés à l'outil, puis j'ai réuni des informations sur chacun d'entre eux avant de tester leur fonctionnalités, ainsi que leurs performances. Ces informations réunies sont présentées ci-dessous.

3.1.1 TreeTagger

TreeTagger est un logiciel d'étiquetage morphosyntaxique développé par Helmut Schmid dans le cadre d'un projet de l'institut de linguistique informatique de l'université de Stuttgart. Son fonctionnement a été décrit par son créateur dans deux articles : *Probabilistic Part-of-Speech Tagging Using Decision Trees* [Schmid, 1994] et *Improvements in Part-of-Speech Tagging with an Application to German* [Schmid, 1995]. Un de ses principaux avantages est le très grand nombre de langages qu'il est capable de traiter : Son site Internet propose des modèles pour plus de vingt langues différentes, dont des langues anciennes telles que le latin ou le grec ancien. Malgré son âge, TreeTagger reste un outil très utilisé dans le domaine du TAL, ce qui explique sa présence dans cette liste.

3.1.2 SpaCy

SpaCy est une bibliothèque Python qui propose un grand nombre de fonctionnalités utiles au traitement de la langue dont des outils d'étiquetage morphosyntaxique. Elle a été développée par un entreprise nommée Explosion. Bien que le nombre de langages qu'elle est capable de traiter soit inférieur au nombre de langages pris en compte par TreeTagger, cette bibliothèque permet à un utilisateur d'entraîner ses propres modèles. Tout comme TreeTagger, SpaCy est aussi un outil utilisé dans le domaine du TAL, et il est donc pertinent de l'inclure dans cette liste.

3.1.3 CoreNLP

CoreNLP est un logiciel de traitement linguistique fonctionnant sur le langage de programmation Java qui inclut un grand nombre de fonctionnalités, dont un étiqueteur morphosyntaxique. Il a été développé par le département de TAL de l'université de Stanford, et a été décrit par ses créateurs dans un article intitulé *The Stanford CoreNLP Natural Language Processing Toolkit* [Manning et al., 2014]. Il est capable de traiter six langues : Le français, l'anglais, l'espagnol, l'allemand, l'arabe et le chinois. Il possède de nombreuses

fonctions au delà du traitement morphosyntaxique, mais ces fonctionnalités ne seront pas utiles dans le cadre du projet. Néanmoins, il reste un programme très utile à ajouter à cette liste.

3.1.4 UDPipe

UDPipe réunit lui aussi un grand nombre de fonctionnalités, y compris une fonctionnalité d'étiquetage morphosyntaxique. Il a été conçu par Milan Straka et Jana Straková, qui ont aussi écrit un article intitulé *Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe*[Straka and Straková, 2017] détaillant son fonctionnement. Cet outil permet à son utilisateur d'entraîner des modèles de langue lui même, mais il possède aussi un grand nombre de modèles pré-entraînés, disponibles depuis son site Internet. Tout comme TreeTagger, ces modèles par défaut incluent des modèles couvrant des langues anciennes telles que le latin ou le français du Moyen-Âge. La grande diversité de langages traités rendent cet outil très utile dans le cadre de la réalisation de l'outil demandé.

3.1.5 NLTK

NLTK(Natural Language ToolKit) est une bibliothèque Python de traitement du langage. Elle a pour objectif de rassembler un grand nombre d'outils utiles pour le traitement de la langue, et inclut aussi un étiqueteur morphosyntaxique. Elle a été mise au point par l'équipe du NLTK Project, qui a détaillé son fonctionnement dans un livre intitulé *Natural Language Processing with Python*[Bird and Klein, 2009]. Comme SpaCy, cette bibliothèque ne peut fonctionner qu'à l'aide du langage de programmation Python, mais elle reste un outil très utilisé dans le domaine du TAL qu'il faut donc noter.

3.1.6 Talismane

Talismane (Traitement Automatique des Langues par Inférence Statistique Moyennant l'Annotation de Nombreux Exemples) est un outil d'analyse automatique qui permet entre autres d'effectuer un étiquetage morphosyn-

taxique. Il a été développé par Assaf Urieli dans le cadre de sa thèse, intitulée *Robust French syntax analysis : reconciling statistical methods and linguistic knowledge in the Talismane toolkit*[Urieli, 2013], puis a fait l'objet d'un article intitulé *L'apport du faisceau dans l'analyse syntaxique en dépendances par transitions : études de cas avec l'analyseur Talismane*[Urieli and Tanguy, 2013]. Contrairement aux autres programmes présentés ci-dessus, Talismane ne peut traiter qu'un nombre très limité de langages. Néanmoins, il est inclus dans cette liste car il est capable de traiter deux des langues les plus présentes dans les projets actuels de l'équipe ELAN.

3.1.7 Test et résultats

Une fois la liste des logiciels établie, il a été nécessaire de tester leurs performances afin de choisir le ou lesquels inclure à l'outil final. Ce test consistait en deux parties : Premièrement, j'ai effectué un test basique de chacun des logiciels afin de tester leur fonctionnement et pour me familiariser avec leur fonctionnement. A l'issue de ce test préliminaire, j'ai créé un tableau représentant les fonctionnalités de chacun d'entre eux, ainsi que d'autres informations importantes comme les langues traitées et leur licence. Ce tableau étant trop grand pour être reproduit ici, il est donc reproduit dans l'annexe B. A partir de ce tableau, j'ai pu vérifier lesquels pourraient remplir les besoins du projet, ce qui a mené à l'exclusion de NLTK qui ne remplissait pas toutes les conditions requises. Pour le second test, j'ai utilisé un texte littéraire tiré de l'un des projets de l'équipe ELAN, à savoir un extrait des cahiers de Henri de Régnier, récupéré dans le cadre du projet ENCHRE. Ce texte est reproduit dans l'annexe H. J'ai d'abord étiqueté ce texte manuellement en utilisant le jeu d'étiquettes UD. Cette version étiquettée manuellement est reproduite dans l'annexe I. J'ai ensuite converti les résultats des étiqueteurs pour qu'ils utilisent tous les étiquettes UD si nécessaire : en effet, la plupart des logiciels évalués utilisaient les étiquettes UD par défaut, donc cette conversion n'a été nécessaire que pour TreeTagger et Talismane. Les deux tableaux ci-dessous précisent quelles conversions ont été appliquées aux jeux d'étiquettes de TreeTagger et Talismane.

Etiquette TreeTagger	Etiquette UD
ABR	X
ADJ	ADJ
ADV	ADV
DET :ART	DET
DET :POS	DET
INT	INTJ
KON	CONJ
NAM	PROPN
NOM	NOUN
NUM	NUM
PRO	PRON
PRO :DEM	PRON
PRO :IND	PRON
PRO :PER	PRON
PRO :POS	PRON
PRO :REL	PRON
PRP	ADP
PRP :det	ADP
PUN	PUNCT
PUN :cit	PUNCT
SENT	PUNCT
SYM	SYM
VER :cond	VERB
VER :futu	VERB
VER :impe	VERB
VER :impf	VERB
VER :infi	VERB
VER :pper	VERB
VER :ppre	VERB
VER :pres	VERB
VER :simp	VERB
VER :subi	VERB
VER :subp	VERB

TABLE 1 – Conversion de jeu d'étiquettes de Treetagger vers UD

Etiquette Talismane	Etiquette UD
ADJ	ADJ
ADV	ADV
ADVWH	ADV
CC	CCONJ
CLO	PRON
CLR	PRON
CLS	PRON
CS	SCONJ
DET	DET
DETWH	DET
ET	X
I	INTJ
NC	NOUN
NPP	PROPN
P	ADP
P+D	ADP
P+PRO	ADP
PONCT	PUNCT
PREF	X
PRO	PRON
PRORE	PRON
PROWH	PRON
V	VERB
VIMP	VERB
VINF	VERB
VPP	VERB
VPR	VERB
VS	VERB

TABLE 2 – Conversion de jeu d'étiquettes de Talismane vers UD

Comme ces tableaux le montrent, les jeux d'étiquettes de Talismane et de TreeTagger étaient tous deux plus précis que le jeu d'étiquette UD, donc la conversion a pu se faire sans trop de problèmes. Néanmoins, les deux étiqueteurs étudiés possédaient tous les deux des étiquettes qui n'étaient pas

présentes dans le jeu d'étiquette UD, et qui ont donc du être converties en utilisant l'étiquette "autre" ou X de UD, ce qui pourrait affecter quelque peu leur performances lors des comparaisons finales.

Enfin, j'ai comparé cette version étiquetée manuellement avec les résultats obtenus en analysant le texte avec chacun des cinq étiqueteurs morphosyntaxiques et en les convertissant au jeu d'étiquettes UD si nécessaire.

Les résultats obtenus détaillés sont reproduits dans l'annexe C de ce document. De plus, un court extrait de ces résultats ne contenant que les performances globales et non les performances par tâche se trouve dans le tableau ci-dessous.

	SpaCy	TreeTagger	CoreNLP	Talismane	UDPipe
Performance globale	76,68 %	80,67 %	75,25 %	81,58 %	80,21 %

TABLE 3 – Extrait des résultats de l'analyse des outils morphosyntaxiques

Les deux logiciels les plus performants sur ce test étaient TreeTagger et Talismane, mais j'ai fait le choix de tous les conserver pour la phase de développement, car leurs performances n'étaient pas trop basses comparées aux deux meilleurs, et certains offraient des fonctions qui n'étaient pas disponibles avec TreeTagger ou Talismane.

3.2 Analyse des outils textométriques

Après avoir décidé quels étiqueteurs morphosyntaxiques devaient être intégrés à l'outil, il était aussi nécessaire de déterminer quel logiciel de textométrie serait le plus adapté. Les conditions étaient ici plus difficiles à remplir : en effet, il était nécessaire de trouver un logiciel qui puisse accepter les fichiers déjà traités par un des étiqueteurs morphosyntaxiques présentés ci-dessus, et qui puisse fonctionner dans une chaîne de traitement et non seulement de façon indépendante. De plus, il était nécessaire qu'il puisse fonctionner selon les choix techniques qui ont été précédemment indiqués. Pour ce faire, il était aussi nécessaire de constituer une liste, présentée ci-dessous.

3.2.1 Point de départ

Pour établir cette liste, j'ai reçu l'aide de l'équipe d'ELAN une fois de plus, mais je me suis aussi basé sur un article qui m'avait été recommandé. Cette article, intitulé *Sept logiciels de textométrie*[Pincemin, 2018] m'a servi de point de départ pour la constitution de cette liste en indiquant un premier échantillon de logiciels qui pourraient être utiles.

3.2.2 DTMVIC

DTMVIC (Data and Text Mining : Visualization, Inference, Classification) est un logiciel de textométrie développé par Ludovic Lebart. Il permet d'effectuer un grand nombre de mesures statistiques sur un texte ou un corpus de textes. Il est caractérisé par sa spécialisation dans le traitement de données textuelles lorsqu'elles sont associées à des données numériques, comme dans le cas d'un sondage par exemple. Néanmoins, cet outil reste capable de traiter des données purement textuelles aussi. Cependant, il se présente sous la forme d'un programme indépendant, ce qui rend son intégration à une chaîne de traitement difficile. De plus, il ne fonctionne que sous le système d'exploitation Windows, ce qui le rend inadapté aux besoins du projet.

3.2.3 HyperBase

HyperBase est un logiciel de traitement textométrique de corpus textuels, développé par Étienne Brunet. Il est capable d'effectuer un certain nombre de traitements statistiques sur les corpus de textes qui lui sont fournis comme entrées. C'est un outil très utilisé dans le domaine des sciences sociales, mais sa structure en tant que programme indépendant le rend lui aussi difficile à intégrer à une chaîne de traitement. De plus, la version téléchargeable est conçue pour Windows, ce qui le rend lui aussi inadapté aux besoins du projet.

3.2.4 IRaMuTeQ

IRaMuTeQ (Interface de R pour les Analyses Multidimensionnelles de Textes et de Questionnaires) est un logiciel de traitement statistique de

données textuelles, développé par le LERASS (Laboratoire d'Études et de Recherches Appliquées en Sciences Sociales) de l'Université de Toulouse 3. Contrairement aux deux outils présentés précédemment, il fonctionne grâce aux langages de programmation R et Python, ce qui le rend utilisable sur la majorité des systèmes d'exploitation existants, et particulièrement adapté au projet.

3.2.5 Lexico5

Lexico5 est un outil de traitement textométrique de corpus de textes, développé par l'équipe du SYLED-CLA2T (Système Linguistiques Énonciation Discursivité - Centre d'Analyse Automatique des Textes) de l'Université Paris 3. Il fonctionne sur la base de corpus de textes prétraités, ce qui peut rendre son adaptation difficile aux besoins de l'outil. De plus, il ne fonctionne que sous Windows et MacOS, ce qui veut dire qu'il ne remplit pas les conditions techniques indiquées auparavant.

3.2.6 Le Trameur

Le Trameur est un logicel de traitement textométrique développé aussi par le SYLED-CLA2T de l'Université Paris 3. Comme la plupart des logiciels présentés ci-dessus, il est indépendant, mais il a l'avantage de pouvoir aussi fonctionner depuis la ligne de commande, ce qui faciliterait son intégration à l'outil. De plus, bien que cela ne soit pas nécessaire dans le cas du développement du projet, Le Trameur intègre aussi l'analyseur morphosyntaxique TreeTagger. Cependant, il n'est disponible que sous Windows, ce qui le rend inadapté au projet.

3.2.7 TXM

TXM est un logiciel de textométrie crée dans le cadre du projet Textométrie, et décrit dans un article intitulé *TXM : Une plateforme logicielle open-source pour la textométrie - conception et développement.*[Heiden et al., 2010] Ses capacités sont similaires à celles des autres logiciels présentés ci-dessus,

mais contrairement à la plupart des logiciels présentés, il est compatible avec un plus grand nombre de systèmes d'exploitation, ce qui le rend plus adapté au contexte de ce projet. Cependant, son fonctionnement en tant que programme indépendant devant être lancé depuis sa propre interface fait que son intégration à une chaîne de traitement pourrait poser quelques difficultés. Néanmoins, il reste une option intéressante pour la réalisation du projet.

3.2.8 AnaText

AnaText[Kraif, 2014] est un programme d'analyse textométrique en ligne, développé par Olivier Kraif. Comme les autres outils, présentés ci-dessus, il est capable d'effectuer un grand nombre de traitements statistiques sur des données textuelles. Bien que son fonctionnement en ligne le rende peu adapté au projet, l'équipe d'ELAN a pu me fournir une version partielle de l'outil qui fonctionne indépendamment de toute intégration en ligne, évitant donc ce problème. Il fonctionne sur la majorité des systèmes d'exploitation, remplissant donc les conditions techniques requises.

3.2.9 Résultats

La sélection du programme à intégrer dans ce cas a été plus difficile car aucun d'entre eux ne remplissait les conditions techniques requises, soit car ils ne fonctionnaient pas sous les systèmes d'exploitation demandés, soit car ils ne pouvaient être facilement intégrés à une chaîne de traitement. Néanmoins, j'ai ici aussi rassemblé les spécificités de ces programme dans un tableau, reproduit dans l'annexe D. Après avoir obtenu une version hors ligne du toolkit d'AnaText grâce à l'équipe d'ELAN, j'ai décidé de l'utiliser car AnaText était le programme qui remplissait les conditions nécessaires le mieux maintenant qu'il était utilisable localement.

3.3 Les plate-formes de TAL

Une autre solution qui aurait pu être explorée lors de cette partie de conception du projet et de choix des programmes à intégrer était les plate-

formes de TAL. Elles remplissent une fonction très similaire au but du projet d'intégrer une multitude de programmes de traitement textuel dans une seule interface dans le but de faciliter leur utilisation. Malheureusement, j'ai découvert cette solution trop tard pour pouvoir l'intégrer à temps dans la structure du projet. Néanmoins, j'ai quand même choisi d'établir une liste de ces plate-formes qui aurait pu être très utiles dans la réalisation du projet.

3.3.1 GATE

La première de ces plate-formes est la plate-forme GATE (General Architecture for Text Engineering). Développée par une équipe de projet de l'Université de Sheffield, elle intègre un grand nombre de programmes ayant pour objectif de traiter des données textuelles. De plus, elle intègre aussi des outils de développement conçus pour faciliter tout travail sur de la matière linguistique, ainsi qu'un outil Web crée pour faciliter l'annotation et l'analyse de corpus textuels. Enfin, GATE est un logiciel libre, ce qui le rend très attractif pour des projets de recherche.

3.3.2 LinguaStream

La seconde plate-forme est la plate-forme LinguaStream. Elle a été développée par le GREYC de l'université de Caen. Tout comme GATE, elle intègre un grand nombre de programmes dont le but est de faciliter le traitement de données textuelles, notamment celles au format XML. Elle fournit aussi une interface graphique adaptée à la création de chaînes de traitement en TAL, et aurait donc pu être un outil très utile dans le cadre du projet. Néanmoins, elle semble ne plus être maintenue, et son site Internet n'est accessible qu'au travers d'archives, donc il semble qu'il n'aurait pas été possible d'utiliser cette plate-forme dans le cadre du projet.

3.3.3 BabelScape NLP

La troisème plate-forme de TAL est la plateforme de TAL crée par BabelScape, intitulée BabelScape NLP. Contrairement aux deux plate-formes préce-

dentes, celle-ci prend la forme d'une chaîne de traitement déjà construite. Elle est capable de traiter des textes dans treize langues différentes. Néanmoins, elle n'est pas capable de traiter une certaine partie des langues demandées dans le cadre du projet, donc elle n'aurait pas été très adaptée au projet. De plus, contrairement aux programmes utilisés dans le cadre du projet, cette plate-forme n'est pas un logiciel libre, ce qui aurait rendu son utilisation d'autant plus difficile.

Chapitre 4

Développement

Après avoir effectué les analyses préliminaires présentées ci-dessus, la phase suivante était le développement de l'outil lui-même. Ce chapitre détaille tout ce processus, les changements appliqués au projet, ainsi que les difficultés auxquelles j'ai fait face durant ce développement.

4.1 Le début

La première étape du développement à été la création d'un script Python simple permettant d'utiliser TreeTagger par le biais de la ligne de commande, et ce, sur un ou plusieurs fichiers fournis comme entrées au script. Ce script avait pour vocation de servir de base sur laquelle les autres outils allait pouvoir être ajoutés. A ce point, l'outil était seulement capable d'effectuer un traitement morphosyntaxique grâce à TreeTagger. Après cela, j'ai ajouté les quatre autres étiqueteurs morphosyntaxiques choisis à l'outil un par un jusqu'à ce que l'outil soit capable d'utiliser l'étiqueteur demandé par l'utilisateur parmi les cinq disponibles. A cette étape, j'ai aussi ajouté des options supplémentaires comme le choix de la langue à traiter. Au début, je comptais intégrer toutes les langues traitées par au moins un des étiqueteurs à l'outil. Néanmoins, je me suis aperçu que cela risquait de rendre l'outil trop imposant pour un gain minime. J'ai donc fait le choix de me limiter à 10 langues seulement, en incluant en priorité les langues qui seraient utiles dans le cadre

des projets de l'équipe ELAN. Les langues choisies sont divisées en deux groupes : Les langues vivantes (français, anglais, espagnol, italien, allemand, portugais, grec moderne), et les langues mortes (latin, grec antique, français médiéval, allemand médiéval).

Au terme de cette étape, l'outil était capable de recevoir des fichiers de texte bruts, ainsi que des fichiers XML/TEI comme entrées. Néanmoins, il ne pouvait fournir comme sortie que les formats supportés par les étiqueteurs, à savoir TXT dans la majorité des cas, ainsi que CSV pour ceux capables de produire ce type de sortie.

En tout, cette étape du développement a duré environ un mois. Cependant, au terme de cette étape, l'outil possèdait les fonctions de base demandées, mais certaines améliorations était nécessaires. Premièrement, la procédure d'installation était devenue complexe à cause du grand nombre de prérequis, dont il fallait trouver un meilleur moyen d'installer l'outil. Deuxièmement, le traitement du XML, pourtant but central du projet n'était pas encore au point, et nécessitait donc plus de travail. Enfin, l'interface était encore peu intuitive car basée entièrement sur la ligne de commande. La résolution des ces trois faiblesses de l'outil a formé une grande partie de mon stage, qui est décrite ci-dessous.

4.2 Migration vers Docker

Le grand nombre de prérequis de l'outil rendait son installation difficile. Par conséquent, il était nécessaire de trouver une solution qui permettait d'éviter de demander à l'utilisateur de suivre une procédure d'installation devenue très complexe à cause du grand nombre de programmes à installer. Afin de résoudre ce problème, l'équipe ELAN m'a recommandé d'utiliser l'outil Docker afin de créer un environnement isolé pour l'outil, et de pouvoir construire cet environnement automatiquement en incluant tous ce qui était nécessaire. La partie suivante de mon stage a donc été consacrée à la création d'un environnement Docker pour l'outil, et à lui appliquer les modifications nécessaires afin qu'il puisse fonctionner dans un environnement isolé.

Pour ce faire, la première étape a été la création d'un environnement isolé

qui pourrait recevoir l'outil. Il était donc nécessaire de trouver un moyen de télécharger et de paramétrer les cinq étiqueteurs morphosyntaxiques sans aucune action de la part de l'utilisateur après le premier lancement. Heureusement, Docker permet de créer un fichier contenant une liste d'instructions afin de construite l'environnement isolé, et cette liste d'instructions accepte les instructions de ligne de commande. Vu que l'outil était en train d'être développé sous Linux, j'avais accès à un grand nombre de commandes utiles, dont la commande "wget", qui permet de télécharger un fichier directement depuis son URL. Grâce à cette commande et quelques commandes de manipulation de fichiers, j'ai pu créer une liste d'instructions qui télécharge les cinq analyseurs morphosyntaxiques et les place dans les dossiers spécifiés, ce qui créeait donc l'environnement isolé requis.

La second étape a été l'adaptation de l'outil lui-même à cette nouvelle structure. Jusqu'à maintenant, il devait être configuré par chaque utilisateur afin de connaître la position de chacun des outils, mais maintenant qu'il était déployé dans un environnement isolé qui aurait toujours la même structure de fichier, les positions de chacun des outils m'était connues et pouvaient donc être directement intégrées au code au lieu de nécessiter une configuration manuelle par chaque utilisateur. De même, il avait maintenant des fichiers de sortie et d'entrée dédiés et stables, donc j'ai pu lui intégrer cette structure aussi.

Cette solution n'est néanmoins pas parfaite : En effet, elle rend le premier lancement de l'outil très lent, parce qu'il doit d'abord télécharger les données de tous les prérequis afin de tous les installer dans l'environnement isolé. Cependant, après cette première installation, l'outil fonctionne correctement comme auparavant. Cette solution a donc permis de simplifier grandement le processus d'installation du projet, ce qui le rend moins complexe à utiliser pour un utilisateur lambda. De plus, cette migration avait aussi pour but de faciliter l'intégration prévue de l'outil à une page Internet. Cette migration de la version indépendante vers la version Docker a pris environ un mois aussi.

4.3 Traitement du XML

L'étape suivante a été la création d'une solution pour le traitement des fichiers XML. Avant ce point, l'outil était fonctionnel, mais il n'était capable que de traiter des fichiers de texte brut, ce qui ne convenait pas encore aux besoins de l'équipe d'ELAN. Il m'a donc été nécessaire de l'adapter au traitement de fichiers XML. J'ai donc mis au point trois solutions pour traiter les fichiers XML.

La première solution que j'ai implémentée était de retirer tout élément XML afin d'obtenir seulement le texte brut contenu dans le fichier XML. Cette solution, bien que fonctionnelle, ne préservait pas le XML original dans le résultat, ce qui limitait son utilité. De plus, elle extrayait tout texte présent dans le fichier XML, peut importe sa position, ce qui faisait qu'elle traitait les métadonnées comme une partie du texte lui-même, et que certaines balises utilisées dans le formalisme XML-TEI posait problème, comme les balises '<choice>' par exemple.

La deuxième solution que j'ai mise au point utilisait le même principe d'extraction du texte brut depuis le XML, mais elle inscrivait les résultats directement dans le fichier XML lui même, sous la forme d'un nouvel élément. Cette solution permettait donc de conserver l'intégrité du XML tout en y ajoutant les résultats de l'analyse. Cette version produisait des résultats satisfaisants, mais le fait que les résultats soient inscrits dans un élément séparé dans le fichier XML faisait qu'il était difficile de les exploiter, et certains problèmes présents dans la première solution était toujours présents dans la seconde solution, car les deux solutions étaient construites autour du même principe.

Avec l'aide de l'équipe d'ELAN, je suis donc arrivé à une troisième solution plus satisfaisante. Cette solution utilisait aussi le principe d'extraction de texte brut indiqué ci-dessus, mais elle procédait élément par élément, remplaçant le contenu de l'élément par les résultats de l'analyse. Cela permettait donc à l'outil de conserver la structure du fichier XML tout en inscrivant les résultats dans cette même structure. Afin de fonctionner, cette solution utilisait un système d'élément minimal : Un élément du XML, par exemple

le paragraphe sert d'unité minimale : tout élément qui lui est subordonné est éliminé, tandis que tout élément qui lui est supérieur est conservé. Un exemple des résultats produits par cette approche se trouve dans l'annexe E. Cette approche par élément minimal permet à l'outil de gérer les problèmes qui pourraient être causés par les balises XML utilisées dans les projets de l'équipe ELAN, qui ne contiennent parfois qu'un seul mot, voire une partie d'un mot en les ignorant. Elle permet aussi de n'étiqueter que certaines parties du document, permettant ainsi à l'utilisateur d'éviter d'étiqueter des choses présentes dans le document XML, mais qui ne font pas partie du texte comme les métadonnées. Cette étape a demandé environ un mois et demi de travail.

4.4 Ajout d'une interface Web

Après avoir résolu les problèmes d'installation et de traitement du XML, le dernier problème à régler était celui de l'interface. Pour résoudre ce problème ainsi que pour faciliter l'utilisation de l'outil encore plus, l'équipe ELAN m'a recommandé de le déposer sur un serveur Apache afin de le rendre accessible depuis un navigateur Internet classique, évitant ainsi tout téléchargement de la part de l'utilisateur. Sa structure sous forme d'un conteneur Docker a rendu ce processus en particulier très facile, car l'outil Compose intégré à Docker a pour but d'agréger différents services tels que Apache et Python. Néanmoins, il a été nécessaire d'adapter l'outil lui même : En effet, depuis sa création, il était lancé depuis la ligne de commande mais ce type d'interface ne convenait plus, maintenant que les utilisateurs allaient y accéder par le biais d'un site Internet. Il m'a donc fallu lui créer une interface Web, interface que j'ai réalisé en utilisant les langages HTML et PHP. Cette interface n'a pas changé la structure de l'outil Python en lui-même, mais elle m'a permis de créer un moyen plus simple de l'utiliser, ainsi que de permettre à l'utilisateur d'accéder à des commandes qui étaient auparavant difficiles d'accès. Le développement de cette interface a occupé le reste de mon stage jusque aux tests finaux. Des exemples de captures d'écran de cette interface se trouvent ci-dessous, et des exemples supplémentaires, ainsi qu'une version agrandie

des deux captures d'écran ci-dessous se trouvent dans l'annexe F.

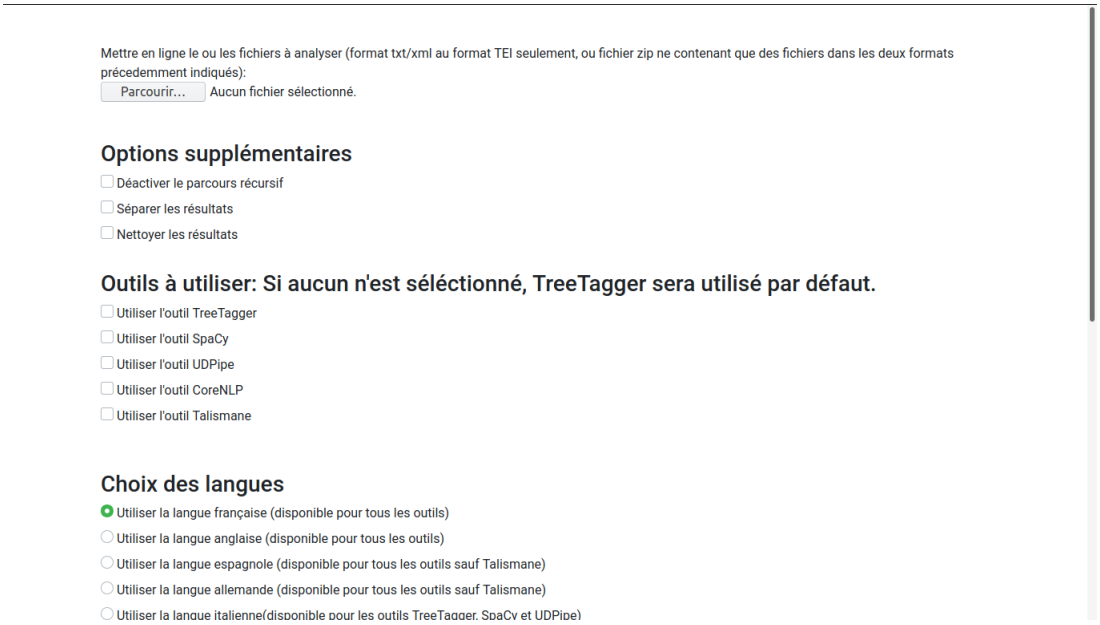


FIGURE 1 – Capture d'écran de la page d'accueil de l'outil version Web (1/2)

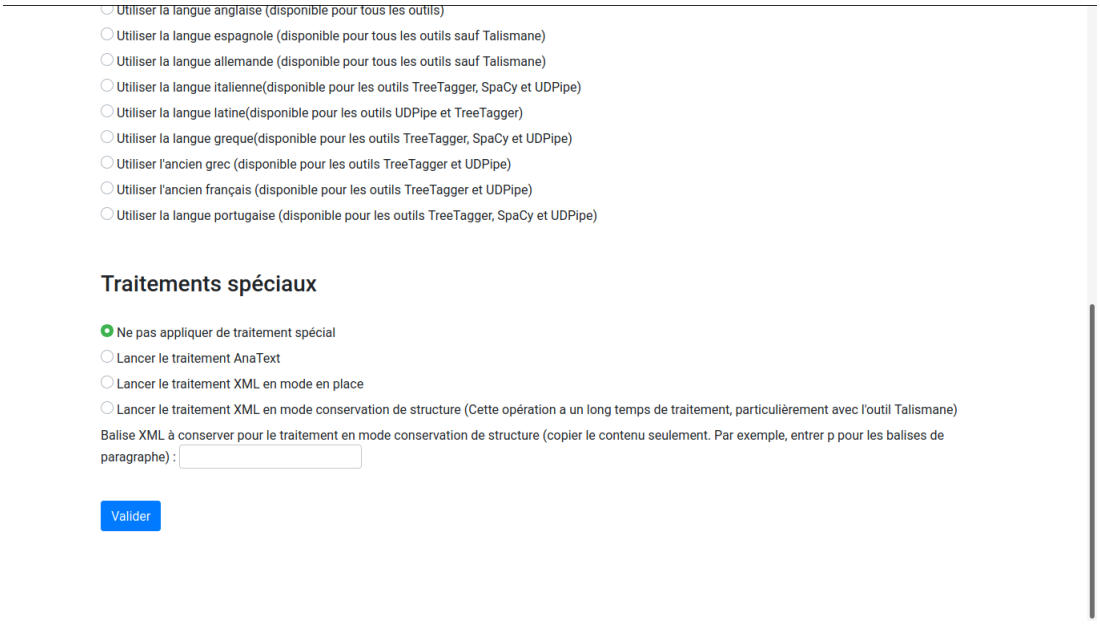


FIGURE 2 – Capture d'écran de la page d'accueil de l'outil version Web (2/2)

Cependant, l'ancienne interface en ligne de commande existe encore dans le programme bien qu'elle soit indisponible sur la version Web. Cela veut dire que l'outil peut toujours être utilisé par le biais de la ligne de commande, ce qui pourrait être utile pour l'intégrer à une grande variété de chaînes de traitement si nécessaire.

4.5 Difficultés rencontrées

Le développement de l'outil n'a pas été sans son lot de problèmes. Le premier que j'ai rencontré est devenu apparent dès le début de mon stage : puisque l'équipe d'ELAN utilisait le système d'exploitation Linux, il m'a fallu m'y adapter, ce qui m'a causé quelques difficultés pendant une période au début de mon stage. L'équipe d'ELAN m'a grandement aidé à accomplir ce but, mais un certain temps d'adaptation m'a quand même été nécessaire.

Le deuxième problème est survenu peu de temps après, durant la sélection d'un logiciel de textométrie adapté à l'outil. En effet, un grand nombre de ces logiciels ne convenaient pas à la structure du projet, et certains d'entre eux possédait des problèmes de compatibilité avec Linux. Le troisième problème auquel j'ai du faire face était le problème de traitement du XML. Comme décrit ci-dessus, il m'a fallu créer trois systèmes différents avant d'en obtenir un qui remplissait les conditions attendues. Néanmoins, malgré ces problèmes, j'ai réussi à accomplir ce qui m'était demandé en concevant l'outil qui correspondait aux demandes de l'équipe d'ELAN.

4.6 Bilan personnel du développement

Malgré les difficultés que j'ai rencontrées, le processus de développement entrepris durant mon stage m'a permis de me former à l'utilisation de nombreux outils très utiles dans le domaine du TAL, ainsi que dans le domaine de l'informatique en général. Par exemple, l'utilisation de l'outil Git au sein de l'équipe ELAN m'a aidé à maîtriser l'utilisation des outils de gestion de version, en particulier dans un contexte de travail collaboratif. De même, les

recommandations techniques qui m'ont été faites par l'équipe d'ELAN m'ont permis de découvrir plusieurs autres logiciels utiles pour certaines tâches spécifiques, tels que Docker. Ce stage a aussi été pour moi une opportunité d'utiliser plusieurs langages de programmation tels que Python ou PHP, ainsi que le langage de balisage HTML dans une situation de développement réelle et donc d'améliorer ma maîtrise de chacun d'entre eux. De plus, la rédaction de documents durant le stage et l'écriture de ce rapport m'ont donné l'opportunité d'apprendre à utiliser le langage LaTeX pour formater plus facilement mes productions écrites. Enfin, la situation quelque peu inhabituelle causée par la crise sanitaire m'a donné une chance de m'adapter au travail à distance et m'a permis de m'habituer à travailler de façon plus indépendante. Un lien vers le répertoire GitLab où le projet est stocké se trouve dans l'annexe G ainsi que dans la sitographie.

Conclusion

Ce qui a été accompli

Le résultat final de mon travail a été la création d'un outil complet, utilisé par le biais d'une interface Web capable d'effectuer l'étiquetage morphosyntaxique d'un ou plusieurs textes ou fichiers XML avec un ou plusieurs des cinq étiqueteurs morphosyntaxiques intégrés, puis d'afficher ces résultats sous trois formes différentes : sous la forme d'un texte brut produit par les étiqueteurs morphosyntaxiques, sous la forme d'un fichier XML modifié, ou sous la forme d'un page Web créée par AnaText. Cet outil constitue donc une chaîne de traitement comme demandé par l'équipe ELAN.

Ce qui devrait suivre

Les objectifs fixés par l'equipe ELAN ont été accomplis, néanmoins, certaines améliorations pourraient être apportées au résultat final afin de l'améliorer. Par exemple, l'interface Web actuelle n'a pas de feuille de style, ce qui lui donne une apparence encore très rudimentaire et peu ergonomique. De plus, bien que l'interface soit terminée, elle n'a pas été encore mise en ligne, ce qui la rend pour le moment inaccessible en dehors de l'équipe ELAN. De plus, l'outil n'intègre pas d'étape de normalisation, ce qui peut causer des problèmes si les textes fournis comme entrées ne suivent pas un format standardisé. Enfin, l'adaptation d'AnaText à l'outil n'est pas complète et possède encore quelques erreurs qui diminuent quelque peu son utilité. Le plus grand problème à résoudre reste l'harmonisation des étiquettes : en effet, les

étiqueteurs utilisent presque tous des étiquettes différentes qui doivent être converties vers un seul jeu d'étiquettes commun, ce qui pose de nombreux problèmes de perte de précision, surtout lorsque certains jeux d'étiquettes sont plus détaillés que d'autres. L'outil intègre un système de conversion, mais il est encore très peu développé et cause donc toujours un grand nombre d'erreurs de conversion, ce qui diminue la précision des résultats.

Bibliographie

[Bird and Klein, 2009] Bird, Steven, E. L. and Klein, E. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

[Heiden et al., 2010] Heiden, S., Magué, J.-P., and Pincemin, B. (2010). TXM : Une plateforme logicielle open-source pour la textométrie - conception et développement. In Bolasco, S., Chiari, I., and Giuliano, L., editors, *10th International Conference on the Statistical Analysis of Textual Data - JADT 2010*, volume 2, pages 1021–1032, Rome, Italy. Edizioni Universitarie di Lettere Economia Diritto.

[Kraif, 2014] Kraif, O. (2014). *Corpus parallèles, corpus comparables : quels contrastes ?* Habilitation à diriger des recherches, Université de Poitiers.

[Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[Pincemin, 2018] Pincemin, B. (2018). Sept logiciels de textométrie. https ://halshs.archives-ouvertes.fr/halshs-01843695.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing.*

[Schmid, 1995] Schmid, H. (1995). Improvements in part-of-speech tagging with an application to german. In *ACL SIGDAT-Workshop.*

[Straka and Straková, 2017] Straka, M. and Straková, J. (2017). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task : Multilingual Parsing from Raw Text to*

Universal Dependencies, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

[Urieli, 2013] Urieli, A. (2013). *Robust French syntax analysis : reconciling statistical methods and linguistic knowledge in the Talismane toolkit*. PhD thesis, Université de Toulouse II le Mirail.

[Urieli and Tanguy, 2013] Urieli, A. and Tanguy, L. (2013). L'apport du faisceau dans l'analyse syntaxique en dépendances par transitions : études de cas avec l'analyseur Talismane. In *Actes de la 20e conférence sur le Traitement Automatique des Langues Naturelles (TALN'2013)*, pages 188–201, Les Sables d'Olonne, France.

Sitographie

[1] Le trameur. http ://www.tal.univ-paris3.fr/trameur/, dernier accès 24 Mars 2020.

[2] Txm. http ://textometrie.ens-lyon.fr/, dernier accès 23 Mars 2020.

[3] BabelScape. Nlp pipeline. https ://babelscape.com/pipeline, dernier accès 7 Septembre 2020.

[4] Etienne Brunet. Hyperbase. http ://ancilla.unice.fr/, dernier accès 23 Mars 2020.

[5] Equipe d'ELAN. Outils-tal. https ://gitlab.com/litt-arts-num/outils-tal-docker, dernier accès 20 Août 2020.

[6] Equipe du GREYC. Linguastream. https ://web.archive.org/web/20130122052259/http ://www.linguastream.org/, (site archivé) dernier accès 7 Septembre 2020.

[7] Equipe du LERASS. Iramuteq. http ://www.iramuteq.org/, dernier accès 26 Mars 2020.

[8] Equipe du SYLED-CLA2T. Lexico5. http ://www.lexi-co.com/, dernier accès 25 Mars 2020.

[9] Explosion. Spacy : Industrial-strength natural language processing. spacy.io, dernier accès 16 Mars 2020.

[10] Stanford NLP Group. Stanford corenlp. stanfordnlp.github.io/CoreNLP/, dernier accès 13 Mars 2020.

[11] Olivier Kraif. Anatext. http ://phraseotext.univ-grenoble-alpes.fr/anaText/, dernier accès 27 Mars 2020.

[12] Ludovic Lebart. Dtmvic. http ://www.dtmvic.com/, dernier accès 23 Mars 2020.

[13] NLTK Project. Nltk. https ://www.nltk.org/, dernier accès 12 Mars 2020.

[14] Helmut Schmid. Treetagger - a part-of-speech tagger for many languages. www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/, dernier accès 17 Mars 2020.

[15] Milan Straka and Jana Straková. Udpipe. ufal.mff.cuni.cz/udpipe, dernier accès 16 Mars 2020.

[16] GATE Projet Team. Gate : General architecture for text engineering. https ://gate.ac.uk/, dernier accès 7 Septembre 2020.

[17] Assaf Urieli. Talismane traitement automatique des langues par inférence statistique moyennant l'annotation de nombreux exemples. http ://redac.univ-tlse2.fr/applications/talismane.html, dernier accès 18 Mars 2020.

Liste des tableaux

Table des figures

Annexe A

Documentation de l'outil : README

Cette annexe reproduit la documentation utilisateur de l'outil sous sa forme finale après la fin de la réalisation de l'outil.

Documentation du script tag.py - version Docker "lite"

En construction, certains éléments pourraient changer

Note: Pour le moment, cette documentation part du principe que vous utilisez un système d'exploitation basé sur Linux. Les commandes et prérequis peuvent être différents sur d'autres systèmes d'exploitation.

Prérequis

Docker installé, avec la commande:

```
sudo apt install docker
```

Cet outil fait appel aux outils suivants, mais il n'est pas nécessaire de les installer sur votre machine :

-TreeTagger

-UDPipe

-Talismane

-SpaCy

-CoreNLP

Installation et configuration:

Cloner le dépôt Github dans un dossier séparé.

Lors de la première utilisation: ouvrir une ligne de commande dans la copie du dépôt Github et construire l'image Docker avec la commande suivante:

```
docker-compose build
```

Attention

Cette opération peut prendre un certain temps en fonction de la vitesse de votre connexion Internet, car le script doit télécharger les cinq outils d'étiquetage pour les installer dans l'image. Cependant, cette opération n'est nécessaire qu'une seule fois, tout lancement de l'outil suivant réutilisera les fichiers téléchargés.

But:

Créer un script utilisable en ligne de commande pouvant lancer un analyseur morphosyntaxique choisi sur un ou plusieurs fichiers, et peut-être éventuellement intégrer un traitement textométrique des données.

Cet outil n'a pas vocation à implémenter toutes les options de chaque programme. Il est donc possible que certaines fonctionnalités de chacun des analyseurs morphosyntaxiques ne soient pas disponibles.

Le but de cette version allégée de l'installation est de diminuer le temps nécessaire pour l'installation de l'outil en n'installant qu'une partie des langues que peuvent traiter ces outils. Une liste complète des langues installées est disponible dans la section **Langues Installées**, avec leurs codes ISO-639-3.

Etat actuel:

Étiquetage avec TreeTagger, UDPipe, Talismane, SpaCy et CoreNLP disponible.

Traitement rudimentaire de fichiers XML-TEI avec TreeTagger, UDPipe, SpaCy, CoreNLP ou Talismane possible

Traitement en place rudimentaire de fichiers XML-TEI avec TreeTagger, CoreNLP, Talismane, UDPipe et SpaCy possible.

Des exemples de chaque type de sortie effectuées par chaque outil sont disponible dans le dossier *exemples*

Usage:

Déposer les dossier à traiter dans le dossier *input*, qui se trouve dans le dossier *data* du dépôt.

Ensuite, depuis la ligne de commande, dans le dossier contenant le dépôt, démarrer le conteneur Docker avec la commande suivante :

```
docker-compose up -d
```

Une fois le conteneur lancé, entrer la commande suivante pour se mettre à travailler dans le conteneur:

```
docker-compose exec app bash
```

Si la ligne de commande remplace son préfixe par le suivant, vous êtes bien en train de travailler dans le conteneur:

```
root@[12 caractères aléatoires]:
```

Toute commande entrée sera ensuite prise en charge par le conteneur et non par votre machine locale jusqu'à ce que vous quittiez le conteneur.

Pour quitter le conteneur, entrer cette commande:

```
exit
```

Enfin, pour arrêter le conteneur une fois que vous l'avez quitté:

```
docker-compose down
```

Tant que le conteneur n'est pas arrêté, il est toujours possible de le relancer avec la commande

```
docker-compose exec app bash
```

Une fois le conteneur arrêté, il faut le relancer avec la commande suivante pour travailler de nouveau dedans.

```
docker-compose exec app bash
```

Une fois dans le conteneur, l'outil peut être lancé à l'aide de cette commande:

```
python3 tag_d.py entree sortie (--sep) (--norec) (--lang xxx) (--exts .xxx (.xxx,.xxx,...)) (--prog xxxxxx (xxxxxx xxxxxx)) (--inpla
```

entree doit être un fichier ou dossier qui se trouve dans le dossier data/input, et *sortie* est un fichier ou dossier qui sera crée dans le dossier data/output. Dans le deux cas, le chemin d'accès doit être relatif à ces deux dossiers.

Toute commande entre parenthèses est optionnelle. Si une de ces options n'est pas renseignée, le script utilisera les valeurs par défaut indiquées ci-dessous.

Les commandes *entree* et *sortie* doivent être remplacés par leur valeurs, en respectant l'ordre indiqué ci-dessus.

Les *xxx* qui suivent certaines options doivent être remplacés par les choix que vous souhaitez faire, sous le format indiqué ci-dessous.

À l'exception de la commande *--exts*, toute commande qui nécessite un argument ne peut en accepter plus d'un seul.

Si une option n'est pas suivie par des xxx, cela veut dire qu'il n'est pas nécessaire de préciser un argument pour ces options.

Exemple de commande simple:

```
python3 tag_d.py Test Test_sortie.csv
```

Cette commande va lancer le script sur le dossier Test dans le dossier data/input avec les option par défaut précisées ci-dessous et va enregistrer les résultats dans le fichier Test_sortie.csv dans le dossier data/output.

Exemple de commande:

```
python3 tag_d.py Test Test_sortie.csv --norec --lang eng --exts .txt .xml --prog Treetagger SpaCy --inplace
```

Cette commande va lancer le script sur le dossier Test dans le dossier data/input, va enregistrer les résultats dans le fichier Test_sortie.csv dans le dossier data/output, elle va demander au script de ne pas effectuer un parcours récursif, elle précise que la langue à traiter est l'anglais, elle précise que le script ne doit traiter que les fichiers dont l'extension est .txt ou .xml, elle demande au script de lancer les analyseurs TreeTagger et SpaCy, et elle lui indique d'inscrire les résultats dans le fichier si le fichier d'entrée est du XML.

Il est possible d'obtenir un message d'aide résumant toutes ces informations en entrant l'une des deux commandes suivantes:

```
python3 tag_d.py -h
```

ou bien

```
python3 tag_d.py --help
```

Arguments:

entree (argument obligatoire)

Chemin d'accès (relatif au dossier data/input) au fichier ou dossier dont les textes doivent être analysés.

sortie (argument obligatoire)

Chemin d'accès relatif au dossier data/output où les données doivent être stockées. (Toute éventuelle extension de fichier sera retirée si l'option -sep est ajoutée, et le chemin d'accès sera utilisé comme chemin d'accès du dossier à créer). Les formats de sortie recommandés sont .txt et .csv

--sep (argument optionnel)

Change le format de sortie: par défaut, touts les résultats seront concaténés en un fichier avec un séparateur entre chaque fichier. (voir **Exemple de Séparateur**) Si cette option est ajoutée, un nouveau dossier sera créé qui contiendra une copie de l'arborescence du dossier fourni en entrée qui contiendra les fichiers traités, nommés selon le format suivant: "original.xxx_sortie_outil.txt" (par exemple: fichier1.txt_sortie_TreeTagger.txt) .

--norec (argument optionnel)

Indique au script de ne pas effectuer un parcours récursif. Par défaut, le script parcourt récursivement tout dossier trouvé dans le dossier d'entrée. Avec cette option, ce parcours récursif n'aura pas lieu. (aucun effet si l'entrée est un fichier isolé ou un dossier sans sous-dossiers)

--lang (argument optionnel)

Indique quelle langue doit être utilisée pour l'étiquetage. La langue doit être indiquée sous la forme de son code ISO-639-3. (Un outil permettant de rechercher ce code se trouve à l'adresse suivante : Ici. Un tableau présentant les codes ISO-639-3 des langues installées par l'outil se trouve aussi ci-dessous dans la section **Informations supplémentaires**. *(Valeur par défaut:fra)*

--exts (argument optionnel)

Indique la ou les extensions des fichiers qui doivent être traités. Chaque extension doit être séparée par un espace. Il est nécessaire de préciser le point devant le code de chaque extension (écrire .csv ou .txt par exemple). *(Valeur par défaut:.txt)*

--prog (argument optionnel)

Indique quel étiqueteur morphosyntaxique le script doit utiliser. Les commandes pour appeler chacun des étiqueteurs sont disponibles dans leurs fichiers d'information dans le dossier *Informations sur chaque outil*. Un tableau contenant les commandes de tous les étiqueteurs est aussi disponible ci-dessous dans la section **Informations supplémentaires**. Il est possible d'entrer plusieurs programmes. Dans ce cas, le script va les lancer les uns après les autres et inscrire les résultats des différents outils dans le même fichier. *(Valeur par défaut:Treetagger)*

--inplace (argument optionnel)

Indique au script de représenter les résultats de l'analyse de fichiers XML directement dans une copie du fichier XML lui-même sous forme d'un nouvel élément `<text>`. Cette copie se trouvera dans le dossier *output*, et son nom sera *XXX_tagged_YYY.xml*, XXX étant l'outil utilisé et YYY étant le nom original du fichier. Par défaut, le script écrit les résultats dans un nouveau fichier sans aucun formatage XML. Cette option n'a d'effet que sur les fichiers XML, tout autre type de fichier sera toujours traité normalement. Le script reste bien sûr capable de traiter plusieurs fichiers de type identique ou différents, même si cette option est activée. Renseigner cette option indiquera aussi au script d'ajouter *.xml* à la liste des extensions qui doivent être analysées, il n'est donc pas nécessaire de préciser l'option `--exts .xml` si cette option est activée.

Avertissement: Cette analyse ne fonctionne que sur des fichiers XML-TEI.

--keepstruct (arguement optionnel)

Indique au script d'inscrire les résultats de l'analyse de fichiers XML directement dans une copie du fichier XML, en essayant de conserver la structure originale autant que possible. Cette copie se trouvera dans le dossier *output*, et son nom sera *XXX_XML_tagged_YYY.xml*, XXX étant l'outil utilisé et YYY étant le nom original du fichier. Il est nécessaire de préciser une option après cet argument: ce doit être une balise XML présente dans le texte. Cette balise sera utilisée comme balise minimum: toute balise incluse dans la balise indiquée sera ignorée lors de l'étiquetage, seul le texte brut sera récupéré et traité. Par défaut, le script écrit les résultats dans un nouveau fichier sans aucun formatage XML. Cette option n'a d'effet que sur les fichiers XML, tout autre type de fichier sera toujours traité normalement. Le script reste bien sûr capable de traiter plusieurs fichiers de type identique ou différents, même si cette option est activée. Si cette option ainsi que l'option "inplace" sont activées, celle-ci sera la seule lancée. Le script tente aussi de séparer le texte analysé en phrase si possible, phrases qui sont représentées sous la forme d'éléments XML `<s></s>`. Cependant, tous les outils ne permettent pas ce traitement: les options de traitement des phrases intégrées à SpaCy et TreeTagger ne convenaient pas aux demandes du script, donc la version implémentée dans le script n'est pas capable de séparer les phrases, et inscrit tout le contenu de chaque balise dans une seule balise `<s>`. Cette information est présentée ci-dessous dans le tableau **Tableau récapitulatif des langues traitées par chaque étiqueteur**.

Avertissement: Cette analyse ne fonctionne que sur des fichiers XML-TEI.

Avertissement: Ajouter cette option allonge grandement le temps de traitement car chaque balise est traitée individuellement au lieu de traiter tout le texte en une fois, et le temps de chargement de chacun des outils peuvent devenir très long dans ce type de cas.

--clean (argument optionnel)

Indique au script de ne pas ajouter de séparateurs entre les résultats de l'analyse de chaque fichier. Normalement, le script ajoute le séparateur indiqué ci-dessous dans la partie **Exemple de Séparateur** entre la sortie de chaque fichier afin d'améliorer la lisibilité du résultat final. Cette option permet de désactiver cet ajout afin d'avoir des données plus "propres", qui peuvent être ensuite réutilisées plus facilement.

--ana (argument optionnel)

Indique au script de lancer la partie analyse de AnaText sur les fichiers donnés en entrée, et compile les résultats dans un fichier compressé qui apparaît dans le dossier de sortie habituel. Ne fonctionne pas si l'option --keepstruct ou l'option --inplace est activée. Si un programme de traitement autre que TreeTagger est choisi, les résultats seront convertis en résultats TreeTagger afin d'être traités par AnaText

Informations supplémentaires

Mini-serveur Apache

Cet outil inclut maintenant un serveur Apache local afin d'afficher les fichiers qui résultent du lancement d'AnaText sur les fichiers fournis en entrée. Pour accèder à ce serveur, il est d'abord nécessaire de renseigner le port que doit utiliser ce serveur dans le fichier *.env.dist* en complètant la ligne PORTAPACHE par le port auquel le serveur doit se relier, puis de renommer ce fichier en *.env*. Le serveur est en suite accessible tant que le conteneur Docker est lancé en tapant ceci dans la barre d'adresse de tout navigateur Internet: *http://localhost:XXX/*, en remplaçant les XXX par l'identifiant du port renseigné selon les instructions ci-dessus. Ce mini serveur permet aussi de lancer l'outil. Pour ce faire, il faut mettre en ligne le ou les fichiers à traiter. Ces fichiers doivent consister soit de fichiers aux formats XML (selon le formalisme TEI) ou TXT, soit de dossiers au format ZIP contenant des fichiers aux formats XML ou TXT. Ensuite, il est possible de choisir quelles options appliquer au traitement à l'aide des boutons présents sur la page du site. Les résultats apparaîtront dans le dossier de sortie comme normalement, mais ils seront aussi téléchargeables depuis le mini-site. Les résultats seront nommés automatiquement en fonction du nom du fichier fourni comme entrée. Selon les parties de l'outil que vous souhaitez utiliser, le temps de traitement peut être assez long.

Exemple de Séparateur:

```
 |\================================================================================|
Sortie yyy du fichier "x.xxx"
|\================================================================================|
```

yyy indique l'outil utilisé pour l'étiquetage du texte.

Ce séparateur n'est pas inscrit si l'option 'clean' est activée.

Langues installées:

Langue	Code ISO-639-3
Français	fra
Anglais	eng
Allemand	deu
Espagnol	spa
Italien	ita
Latin	lat
Grec moderne	ell
Grec ancien	grc
Vieux Français	fro
Portugais	por

Tableau des commandes des étiqueteurs

Etiqueteurs	Commandes
TreeTagger	TreeTagger, Treetagger, TTG, treetagger
SpaCy	SpaCy, Spacy, SPC, spacy
CoreNLP	CoreNLP, corenlp, NLP
UDPipe	UDPipe, UDpipe, UDP, udpipe
Talismane	Talismane, talismane, TAL

Tableau récapitulatif des langues traitées par chaque étiqueteur

Note : Certains de ces étiqueteurs sont capables de traiter d'autres langages si installés de façon indépendante, mais le choix de limiter le nombre de langues traitées à été fait afin de réduire la taille de l'installation de l'application.

Etiqueteurs	Langues	Capable de séparer les phrases

Etiqueteurs	Langues	Capable de séparer les phrases
TreeTagger	fra, eng, deu, spa, ita, lat, ell, grc, fro, por	Non
SpaCy	fra, eng, deu, spa, ita, ell, por	Non
CoreNLP	fra, eng, deu, spa	Oui
UDPipe	fra, eng, deu, spa, ita, lat, ell, grc, fro, por	Oui
Talismane	fra, eng	Oui

Liste des erreurs et de leur signification (liste non exhaustive)

Langage incorrect ou non supporté par (programme):

Le code ISO-639-3 entré ne correspond pas à un langage que le programme choisi est capable de traiter. Une référence des codes ISO-639-3 est disponible à cette adresse.

Le tableau **Tableau récapitulatif des langues traitées par chaque étiqueteur** ci-dessus liste les langues traitées par chaque outil.

Programme non supporté ou commande de programme incorrect:

La commande de programme entrée ne correspond pas à un programme implémenté dans l'outil. Une liste des commandes valides se trouve plus haut dans ce document.

Dossier déjà existant

Cette erreur indique qu'il existe déjà un dossier dans le dossier de sortie qui porte le nom du dossier que vous souhaitiez créer avec l'option *--sep*. Afin d'éviter tout écrasement de données accidentel, le script fait le choix de s'interrompre sans écrire quoi que ce soit dans ce cas. Il suffit de choisir un autre nom pour le dossier de sortie pour corriger cette erreur.

Etiquette 'text' non trouvée

Cette erreur indique que le fichier XML donné comme entrée ne possède pas de balise *text* qui délimite ce qui doit être analysé. Pour le moment, ce script n'est capable d'analyser un fichier XML que si il possède un élément *text* contenant ce qui doit être analysé. Cette erreur peut aussi indiquer que le fichier fourni en entrée n'est pas un fichier XML-TEI, auquel cas il ne peut être traité par l'outil.

Annexe B

Tableau résumé des étiqueteurs morphosyntaxiques

Cette annexe reproduit le tableau dans lequel j'ai inscrit les fonctionnalités de chacun des étiqueteurs morphosyntaxiques ; tableau que j'ai utilisé afin de sélectionner quels analyseurs devaient être intégrés à l'outil.

	Etiqueteurs	SpaCy	NLTK	TreeTagger	CoreNLP	Talismane	UDPipe
Type		Module Python	Module Python	Outil indépendant	Outil indépendant+Outil Java	Outil indépendant+ Outil Java	Outil indépendant
Fonctionnalités (Oui/Non/Partiel)							
Tokénisation		Oui	Oui	Oui	Oui	Oui	Oui
Lemmatisation		Oui	Oui	Oui	Partielle (Nombreux langages n	Oui	Oui
Etiquetage morphosyntaxique		Oui	Oui	Oui	Oui	Oui	Oui
Segmentation en phrases		Oui	Oui	Non	Oui	Oui	Oui
Analyse grammaticale par dépendances		Oui	Oui	Non	Oui	Oui	Oui
Chunking		Non	Oui	Partiel (Prise en c	Non	Non	Non
Reconnaissance d'entités nommées		Oui	Non	Non	Partielle (Nombreux langages n	Non	Non
Nettoyage XML		Non	Non	Non	Oui	Non	Non
Analyse du genre et du nombre		Non	Non	Non	Non	Oui	Non
Résolution des coréférences		Non	Non	Non	Partielle (Nombreux langages n	Non	Non
Récupération de la casse		Non	Non	Non	Oui	Non	Non
Extraction des relations entre entités		Non	Non	Non	Partielle (Nombreux langages n	Non	Non
Opérations de classification de textes		Non	Oui	Non	Non	Non	Non
Extraction des connaissances		Non	Non	Non	Oui	Non	Non
Classification par *Clustering*		Non	Oui	Non	Non	Non	Non
Analyse des sentiments		Non	Oui	Non	Partielle (Nombreux langages n	Non	Non
Lecture de formats spéciaux (XML,ConLL,etc...)		Non	Oui	Non	Non	Non	Non
Extraction de radicaux		Non	Oui	Non	Non	Non	Non
Calcul de métriques d'évaluation		Non	Oui	Non	Non	Non	Non
Analyse sémantique		Non	Oui	Non	Non	Non	Non
Analyse morphologique		Oui	Non	Non	Non	Non	Non
Identification de colocations		Non	Oui	Non	Non	Non	Non
Langues traitées par défaut (Oui/Non)							
Français		Oui	Non	Oui	Oui	Oui	Oui
Anglais		Oui	Oui	Oui	Oui	Oui	Oui
Espagnol		Oui	Non	Oui	Oui	Non	Oui
Italien		Oui	Non	Oui	Non	Non	Oui
Allemand		Oui	Non	Oui	Oui	Non	Oui
Chinois		Non	Non	Oui	Oui	Non	Oui
Arabe		Non	Non	Non	Oui	Non	Oui
Russe		Non	Non	Oui	Non	Non	Oui
Grec		Oui	Non	Oui	Non	Non	Oui (Moderne et Ancien)
Néerlandais		Oui	Non	Oui	Non	Non	Oui
Latin		Non	Non	Oui	Non	Non	Oui
Ancien Français		Non	Non	Oui	Non	Non	Oui
Autres		Portugais,Norvé	Aucune	Oui, voir TreeTagg	Aucune	Occitan	Oui
Ajout de nouvelles langues par l'utilisateur possible		Oui	Oui	Oui	Oui	Oui	Oui
License		MIT License	Apache License	License non comm	GNU General Public License v	Affero GPL v3	Mozilla Public License 2.0

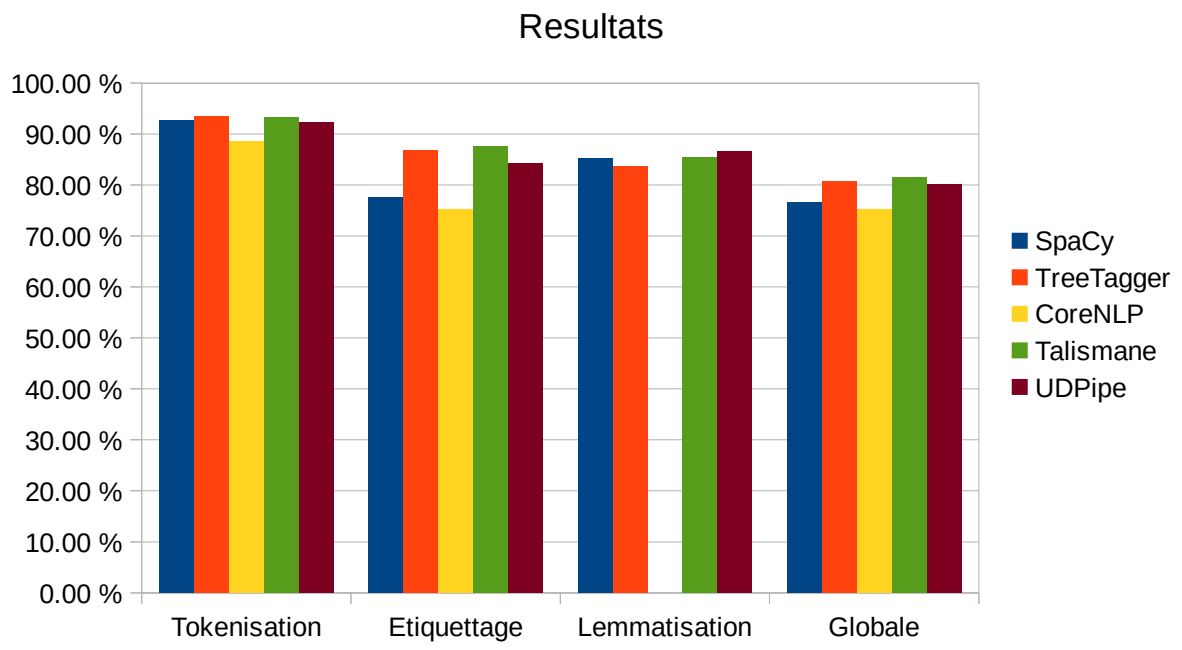
Annexe C

Court rapport sur les outils d'analyse morphosyntaxique

Cette annexe reproduit le court rapport que j'ai écrit après avoir manuellement évalué les étiqueteurs morphosyntaxiques.

Résultats de l'évaluation des étiqueteurs morphosyntaxiques :

Cinq étiqueteurs (TreeTagger[1,2,3],SpaCy[4],CoreNLP[5,6],Talismane[7,8,9] et UDPipe[10,11,12] ont été lancés sur un texte de ~500 mots tiré du corpus ENCHRE (Edition Numérique des Cahiers de Henri de Régnier), qui avait été tokenisé, lemmatisé et étiquetté manuellement auparavant. Les résultats produits par les étiqueteurs ont ensuite été comparés a la version annotée manuellement, et la précision de chaque tâche à été évaluée par un comptage manuel des erreurs. Les résultats de cette évaluation sont reportés dans le graphique ci-dessous :



La fonction de lemmatisation de l'étiqueteur CoreNLP n'est pas représentée dans ce graphique, car elle n'a pas pu être évaluée, puisqu'elle ne fonctionne que sur des textes en anglais. Comme ce diagramme le montre, les performances de CoreNLP dans le cas étudié sont inférieures aux quatre autres étiqueteurs testés dans tous les domaines. Ce diagramme montre aussi que les performances de SpaCy sont relativement basses pour des tâches d'étiquetage dans le cas étudié par rapport à celles de TreeTagger et de Talismane et que les performances d'UDPipe sont les meilleures des cinq étiqueteurs pour la lemmatisation. Malgré ces quelques problèmes, les performances de SpaCy pour les autres tâches sont presque équivalentes à celles des trois autres étiqueteurs. Quant aux trois autres étiqueteurs, la différence de performance entre eux est faible, avec moins d'un point de différence pour la plupart des tâches comme le montre le tableau ci-dessous.

	SpaCy	TreeTagger	CoreNLP	Talismane	UDPipe
Tokenisation	92,71 %	93,41 %	88,63 %	93,27 %	92,23 %
Etiquetage	77,55 %	86,82 %	75,25 %	87,57 %	84,16 %
Lemmatisation	85,28 %	83,75 %	N/A	85,53 %	86,66 %
Globale	76,68 %	80,67 %	75,25 %	81,58 %	80,21 %

La différence de performance globale entre TreeTagger et Talismane est elle aussi très faible, mais Talismane a été plus performant d'environ un point dans ce test.

Une observation en profondeur des erreurs commise par les outils d’étiquetage morphosyntaxique montre que les erreurs les plus fréquentes sont causées par des mot composés, des noms propres et des constructions poétiques propres à cet auteur.

Les résultats de ce test semblent indiquer que, dans ce cas particulier, Talismane est l’étiqueteur le plus performant. Néanmoins, il n’est pas capable de traiter le nombre de langues que peuvent tous deux traiter TreeTagger et UDPipe, donc il n’est peut être pas le plus approprié à la tâche.

Bibliographie :

[1] :Helmut Schmid (1995): Improvements in Part-of-Speech Tagging with an Application to German. *Proceedings of the ACL SIGDAT-Workshop*. Dublin, Ireland.

[2] :Helmut Schmid (1994): Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

[3]:*The Treetagger*, Accessed 12 March 2020 https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[4] :*SpaCy*, Accessed 12 March 2020
https://spacy.io/

[5] :Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky (2014). The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60

[6:]Stanford CoreNLP, Accessed 12 March 2020
https://stanfordnlp.github.io/CoreNLP/

[7] : Urieli, Assaf (2013). Robust French syntax analysis: reconciling statistical methods and linguistic knowledge in the Talismane toolkit. *PhD thesis*. Université de Toulouse II-Le Mirail.

[8] : Urieli, Assaf et Tanguy, Ludovic (2013)**.** L'apport du faisceau dans l'analyse syntaxique en dépendances par transitions : études de cas avec l'analyseur Talismane. *Actes de la conférence Traitement Automatique des Langues Naturelles (TALN 2013)*. Les Sables d'Olonne, France.

[9] : Talismane, Accessed 12 March 2020
http://redac.univ-tlse2.fr/applications/talismane.html

[10] :(Straka et al. 2017) Milan Straka and Jana Straková. *Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe*. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 2017.

[11] :(Straka et al. 2016) Straka Milan, Hajič Jan, Straková Jana. *UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing*. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia, May 2016.

[12] : UDPipe, Accessed 12 March 2020
http://ufal.mff.cuni.cz/udpipe

Annexe D

Tableau résumé des outils de textométrie

Cette annexe reproduit le tableau que j'ai constitué à partir des fonctionnalités de chacun des outils de textométrie que j'ai étudié.

Outils	AnaText	Lexico5	DTMVIC	HyperBase	Iramuteq	LeTrameur	TXM
Fonctionnalités							
Comptage des tokens, des types, etc.	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Analyse multi-texte	Non	Oui	Oui	Oui	Oui	Oui	Oui
Analyse de données numériques	Non	Non	Oui	Non	Non	Non	Non
Analyse du vocabulaire	Oui	Oui	Oui	Oui	Non	Oui	Oui
Analyse de réponses ouvertes	Non	Non	Oui	Non	Non	Non	Non
Opérations de classification automatique	Non	En développement	Oui	Non	Oui	Non	Oui
Calculs de spécificité	Oui	Oui	Non	Oui	Oui	Non	Oui
Extraction de concordances	Oui	Oui	Non	Oui	Non	Oui	Oui
Extraction de cooccurrences	Oui	Oui	Non	Oui	Oui	Oui	Oui
Extraction de segments répétés	Oui	Oui	Non	Oui	Non	Oui	Oui
Visualisation graphique des résultats	Non	Oui	Oui	Oui	Oui	Oui	Oui
Importation de corpus dans leur format natif	Non	Non	Non	Non	Non	Oui	Oui
Suppression de chevrons	Non	Non	Oui	Non	Oui	Oui	Oui
Analyse de similarité	Non	Non	Non	Non	Oui	Non	Oui
Utilisation depuis la ligne de commande	Peut-être	Non	Non	Non	Non	Oui	Non
Plateformes							
Windows	Oui	Oui	Oui	Oui	Oui	Oui	Oui
MAC OS	Oui	Non	Non	Non	Oui	Oui	Oui
Linux	Oui	Non	Non	Non	Oui	Non	Oui (Ubuntu seulement)

Annexe E

Extrait de résultat de la version finale de l'analyse XML

Cette annexe reproduit un court extrait des résultats produits lors du traitement de fichiers XML avec la version finale du traitement intégrée à l'outil.

```
<div type="titre">
  <head type="main">Les Chevaliers</head>
  <head type="sub">Comédie.</head>
</div>
<p>
  <s>
    <w lemma="&lt;unknown&gt;" pos="NAM">Aristophane</w>
    <w lemma="avoir" pos="VER:impf">avait</w>
    <w lemma="railler" pos="VER:pper">raillé</w>
    <w lemma="un" pos="DET:ART">un</w>
    <w lemma="peu" pos="ADV">peu</w>
    <w lemma="trop" pos="ADV">trop</w>
    <w lemma="librement" pos="ADV">librement</w>
    <w lemma="sur" pos="PRP">sur</w>
    <w lemma="le" pos="DET:ART">le</w>
    <w lemma="sujet" pos="NOM">sujet</w>
    <w lemma="du" pos="PRP:det">des</w>
    <w lemma="magistrat" pos="NOM">magistrats</w>
    <w lemma="," pos="PUN">,</w>
    <w lemma="dans" pos="PRP">dans</w>
    <w lemma="un" pos="DET:ART">une</w>
    <w lemma="pièce" pos="NOM">pièce</w>
    <w lemma="intituler" pos="VER:pper">intitulée</w>
    <w lemma=":" pos="PUN">:</w>
    <w lemma="le" pos="DET:ART">Les</w>
    <w lemma="babylonien" pos="ADJ">Babyloniens</w>
    <w lemma="," pos="PUN">,</w>
    <w lemma="qui" pos="PRO:REL">qui</w>
    <w lemma="avoir" pos="VER:impf">avait</w>
    <w lemma="être" pos="VER:pper">été</w>
    <w lemma="représenter" pos="VER:pper">représentée</w>
    <w lemma="dans" pos="PRP">dans</w>
    <w lemma="un" pos="DET:ART">une</w>
    <w lemma="saison" pos="NOM">saison</w>
    <w lemma="ou" pos="KON">ou</w>
    <w lemma="où" pos="PRO:REL">où</w>
    <w lemma="il" pos="PRO:PER">il</w>
    <w lemma="se" pos="PRO:PER">se</w>
    <w lemma="trouver" pos="VER:impf">trouvait</w>
    <w lemma="beaucoup" pos="ADV">beaucoup</w>
    <w lemma="de" pos="PRP">d'</w>
    <w lemma="étranger" pos="NOM">étrangers</w>
    <w lemma="à" pos="PRP">à</w>
    <w lemma="Athènes" pos="NAM">Athènes</w>
```

Annexe F

Captures d'écran

Cette annexe reproduit des captures d'écran de l'interface Web finale de l'outil.

Mettre en ligne le ou les fichiers à analyser (format txt/xml au format TEI seulement, ou fichier zip ne contenant que des fichiers dans les deux formats précedemment indiqués):
Parcourir... Aucun fichier sélectionné.

Options supplémentaires

- [ ] Déactiver le parcours récursif
- [ ] Séparer les résultats
- [ ] Nettoyer les résultats

Outils à utiliser: Si aucun n'est séléctionné, TreeTagger sera utilisé par défaut.

- [ ] Utiliser l'outil TreeTagger
- [ ] Utiliser l'outil SpaCy
- [ ] Utiliser l'outil UDPipe
- [ ] Utiliser l'outil CoreNLP
- [ ] Utiliser l'outil Talismane

Choix des langues

- (•) Utiliser la langue française (disponible pour tous les outils)
- ( ) Utiliser la langue anglaise (disponible pour tous les outils)
- ( ) Utiliser la langue espagnole (disponible pour tous les outils sauf Talismane)
- ( ) Utiliser la langue allemande (disponible pour tous les outils sauf Talismane)
- ( ) Utiliser la langue italienne(disponible pour les outils TreeTagger, SpaCy et UDPipe)

- [ ] Utiliser l'outil SpaCy
- [ ] Utiliser l'outil UDPipe
- [ ] Utiliser l'outil CoreNLP
- [ ] Utiliser l'outil Talismane

Choix des langues

- (•) Utiliser la langue française (disponible pour tous les outils)
- ( ) Utiliser la langue anglaise (disponible pour tous les outils)
- ( ) Utiliser la langue espagnole (disponible pour tous les outils sauf Talismane)
- ( ) Utiliser la langue allemande (disponible pour tous les outils sauf Talismane)
- ( ) Utiliser la langue italienne(disponible pour les outils TreeTagger, SpaCy et UDPipe)
- ( ) Utiliser la langue latine(disponible pour les outils UDPipe et TreeTagger)
- ( ) Utiliser la langue greque(disponible pour les outils TreeTagger, SpaCy et UDPipe)
- ( ) Utiliser l'ancien grec (disponible pour les outils TreeTagger et UDPipe)
- ( ) Utiliser l'ancien français (disponible pour les outils TreeTagger et UDPipe)
- ( ) Utiliser la langue portugaise (disponible pour les outils TreeTagger, SpaCy et UDPipe)

Traitements spéciaux

- (•) Ne pas appliquer de traitement spécial
- ( ) Lancer le traitement AnaText
- ( ) Lancer le traitement XML en mode en place
- ( ) Lancer le traitement XML en mode conservation de structure (Cette opération a un long temps de traitement, particulièrement avec l'outil Talismane)

Utiliser la langue anglaise (disponible pour tous les outils)

Utiliser la langue espagnole (disponible pour tous les outils sauf Talismane)

Utiliser la langue allemande (disponible pour tous les outils sauf Talismane)

Utiliser la langue italienne(disponible pour les outils TreeTagger, SpaCy et UDPipe)

Utiliser la langue latine(disponible pour les outils UDPipe et TreeTagger)

Utiliser la langue greque(disponible pour les outils TreeTagger, SpaCy et UDPipe)

Utiliser l'ancien grec (disponible pour les outils TreeTagger et UDPipe)

Utiliser l'ancien français (disponible pour les outils TreeTagger et UDPipe)

Utiliser la langue portugaise (disponible pour les outils TreeTagger, SpaCy et UDPipe)

Traitements spéciaux

Ne pas appliquer de traitement spécial

Lancer le traitement AnaText

Lancer le traitement XML en mode en place

Lancer le traitement XML en mode conservation de structure (Cette opération a un long temps de traitement, particulièrement avec l'outil Talismane)

Balise XML à conserver pour le traitement en mode conservation de structure (copier le contenu seulement. Par exemple, entrer p pour les balises de paragraphe) :

Valider

Opération terminée.

Téléchargement des fichiers de résultats:

Télécharger Test_20200724071450

retour

Annexe G

Lien vers le répertoire GitLab du projet

Cette annexe contient le lien vers le répertoire GitLab où le projet est stocké. Le projet se trouve à l'adresse suivante : *https ://gitlab.com/litt-arts-num/outils-tal-docker*.

Annexe H

Texte de référence de l'évaluation des analyseurs morphosyntaxiques

Cette annexe contient le court extrait de texte qui a été utilisé pour évaluer les cinq analyseurs morphosyntaxiques.

N° 4

Henri De Regnier
Journal
Cahier IV
De Février 1887 à Avril 1887

H. de Regnier
Journal Février 1887 –
Cahier IV 4
Du Lundi 7 Février au Vendredi 22 Avril 1887
H. de Regnier

Lundi 7 Février. Travaillé la journée, et vu Froc à 6 heures , place de la Concorde ; et sur la Seine, là-bas, par delà les tours du Trocadéro, un ciel d'Orient froid s'embrase d'un coucher de soleil. Les teintes violettes, tassées à l'horizon, s'éclaircissent, et passent au rouge qui se fond dans le bleu supérieur ; un couchant de cendres et d'or, sur lequel se détachent en jaune clair la flamme des réverbères du pont, et tout le long du fleuve, rouge et rose, et noir, les arbres des quais filent en violâtres perspectives. Plus tard, en revenant, la lune était au ciel, et je ne me suis aperçu de sa présence que par son clair reflet aux vitres d'une maison.

L'autre jour, je me suis promené au commencement de cette longue avenue du Bois de Boulogne, sous un doux soleil prématuré, qui, sur les bancs, dorlotait des lassitudes ; des bébés faisaient des pâtés dans le sable, des hommes sans paletot erraient, et je n'ai remarqué dans la blonde et jeune lumière que la couleur et l'élégance de quelques nuques de femmes, aux cheveux retroussés et frisottants. Au ciel, au dessus de l'Arc de Triomphe, dans l'azur pâle, une lune diurne émiettait son disque de pain à cacheter dissous de salive.

Entre les deux lumières d'une table de jeu, m'apparaît une tête de jeune fille, dans la clarté qui baigne ses traits. C'est ce qu'on appelle une belle fille, aux traits durs, hommasses, mais d'un aspect de visage irrémédiablement sale. Le duvet qui s'accentue sous le nez a des noirceurs et des piqûres de barbe, mal rasée, et dans la peau des joues, les lentilles brunes de têtes de boutons arrachés, et de la coiffure sur le front dégringole une mèche de cheveux durs et noirs.

Mardi 8 Février. Mariage de Dorchain à Ste Clothilde. La mariée : une brune figure allongée aux lèvres entrouvertes irrémédiablement. Un cortège restreint. Sully Prudhomme, un peu voûté, l'air las, Coppée, avec un gros paletot, et sa régulière et imberbe figure teintée de pain d'épice. Dans l'église, parmi des Dames élégantes, des littérateurs : Banville, un paletot jusqu'aux talons, le cou engoncé d'un foulard blanc, une tête chauve, et sa face rase de vieux Pierrot ; Lemerre lippu et massif ; Bourget, très bien mis, monocle à l'œil, un peu poseur, des sourcils froncés, l'air dur et faux ; un sentimental égoïste, bien d' aplomb, épaules larges. Le soir, travail mauvais.

Mardi 15 Février. Chez Mallarmé avec Vielé : il y a Ghil, Beaumanoir, Dujardin, puis Duret – cinquante ans environ – un peu grisonnant, nez trop court, trop distant d'une bouche à lèvre un peu pendante – on parle de Manet. Je soupçonne Manet d'avoir été homme si charmant qu'il a fasciné ses amis jusqu'à leur inspirer de l'admiration pour sa peinture – et maintenant grâce aux nombreuses conversations où il fut question de lui ici, je vois assez bien ce qu'il était : peintre indécis, doutant, préoccupé de succès et guettant la venue des Rothschild.

D'amusants détails sur Burty – doué d'un flair puissant, collectionneur enragé jusqu'à se priver pour les objets d'art des femmes qu'il aimait.

Annexe I

Annotation manuelle de réference

Cette annexe contient l'annotation manuelle de référence que j'ai utilisé pour l'évaluation des analyseurs morphosyntaxiques.

Tokens	Token	Cat (UD)	Lemme
1	N°	NOUN	numéro
2	4	NUM	4
3	Henri De Regnier	PROPN	Henri De Regnier
4	Journal	NOUN	journal
5	Cahier	NOUN	cahier
6	IV	NUM	IV
7	De	ADP	de
8	Février	NOUN	février
9	1887	NUM	1887
10	à	ADP	à
11	Avril	NOUN	avril
12	1887	NUM	1887
13	H. de Regnier	PROPN	H. de Regnier
14	Journal	NOUN	journal
15	Février	NOUN	février
16	1887	NUM	1887
17	–	PUNCT	–
18	Cahier	NOUN	cahier
19	IV	NUM	IV
20	4	NUM	4
21	Du	ADP	de le
22	Lundi	NOUN	lundi
23	7	NUM	7
24	Février	NOUN	février
25	au	ADP	à le
26	Vendredi	NOUN	vendredi
27	22	NUM	22
28	Avril	NOUN	avril
29	1887	NUM	1887
30	H. de Regnier	PROPN	H. de Regnier
31	Lundi	NOUN	lundi
32	7	NUM	7
33	Février	NOUN	février
34	.	PUNCT	.
35	Travaillé	VERB	travailler
36	la	DET	le
37	journée	NOUN	journée
38	,	PUNCT	,
39	et	CCONJ	et
40	vu	VERB	voir
41	Froc	PROPN	Froc
42	à	ADP	à
43	6	NUM	6
44	heures	NOUN	heure
45	,	PUNCT	,
46	place de la Concorde	PROPN	place de la Concorde
47	;	PUNCT	;
48	et	CCONJ	et
49	sur	ADP	sur
50	la	DET	le
51	Seine	PROPN	Seine
52	,	PUNCT	,

53	là-bas	ADV	Là-bas
54	,	,	,
55	par	ADP	par
56	delà	ADV	delà
57	les	DET	le
58	tours	NOUN	tour
59	du	ADP	de le
60	Trocadéro	PROPN	Trocadéro
61	,	PUNCT	,
62	un	DET	un
63	ciel	NOUN	ciel
64	d’	ADP	de
65	Orient	PROPN	Orient
66	froid	ADJ	froid
67	s’	PRON	se
68	embrase	VERB	embraser
69	d’	ADP	de
70	un	DET	un
71	coucher	NOUN	coucher
72	de	ADP	de
73	soleil	NOUN	soleil
74	.	PUNCT	.
75	Les	DET	Le
76	teintes	NOUN	teinte
77	violettes	ADJ	violet
78	,	PUNCT	,
79	tassées	VERB	tasser
80	à	ADP	à
81	l’	DET	Le
82	horizon	NOUN	horizon
83	,	PUNCT	,
84	s’	PRON	se
85	éclaircissent	VERB	éclaircir
86	,	PUCT	,
87	et	CCONJ	et
88	passent	VERB	passer
89	au	ADP	à le
90	rouge	NOUN	rouge
91	qui	SCONJ	qui
92	se	PRON	se
93	fond	VERB	fondre
94	dans	ADP	dans
95	le	DET	Le
96	bleu	NOUN	bleu
97	supérieur	ADJ	supérieur
98	;	PUNCT	;
99	un	DET	un
100	couchant	NOUN	couchant
101	de	ADP	de
102	cendres	NOUN	cendre
103	et	CCONJ	et
104	d’	ADP	de
105	or	NOUN	or

106	,	PUNCT	,
107	sur	ADP	sur
108	lequel	PRON	lequel
109	se	PRON	se
110	détachent	VERB	détacher
111	en	ADP	en
112	jaune	NOUN	jaune
113	clair	ADJ	clair
114	la	DET	le
115	flamme	NOUN	flamme
116	des	DET	un
117	réverbères	NOUN	réverbère
118	du	ADP	de le
119	pont	NOUN	pont
120	,	PUNCT	,
121	et	CCONJ	et
122	tout	ADV	tout
123	le	DET	le
124	long	NOUN	long
125	du	ADP	de le
126	fleuve	NOUN	fleuve
127	,	PUNCT	,
128	rouge	ADJ	rouge
129	et	CCONJ	et
130	rose	ADJ	rose
131	,	PUNCT	,
132	et	CCONJ	et
133	noir	ADJ	noir
134	,	PUNCT	,
135	les	DET	le
136	arbres	NOUN	arbre
137	des	ADP	de le
138	quais	NOUN	quai
139	filent	VERB	filer
140	en	ADP	en
141	violâtres	ADJ	violâtre
142	perspectives	NOUN	perspective
143	.	PUNCT	.
144	Plus	ADV	plus
145	tard	ADV	tard
146	,	PUNCT	,
147	en	ADP	en
148	revenant	VERB	revenir
149	,	PUNCT	,
150	la	DET	le
151	lune	NOUN	lune
152	était	AUX	être
153	au	ADP	à le
154	ciel	NOUN	ciel
155	,	PUNCT	,
156	et	CCONJ	et
157	je	PRON	je
158	ne	ADV	ne

159	me	PRON	me
160	suis	AUX	être
161	aperçu	VERB	apercevoir
162	de	ADP	de
163	sa	DET	son
164	présence	NOUN	présence
165	que	ADV	que
166	par	ADP	par
167	son	DET	son
168	clair	ADJ	clair
169	reflet	NOUN	reflet
170	aux	ADP	à le
171	vitres	NOUN	vitre
172	d’	ADP	de
173	une	DET	un
174	maison	NOUN	maison
175	.	PUNCT	.
176	L’	DET	le
177	autre	ADJ	autre
178	jour	NOUN	jour
179	,	PUNCT	,
180	je	PRON	je
181	me	PRON	me
182	suis	AUX	être
183	promené	VERB	promener
184	au	ADP	à le
185	commencement	NOUN	commencement
186	de	ADP	de
187	cette	PRON	ce
188	longue	ADJ	long
189	avenue	NOUN	avenue
190	du	ADP	de le
191	Bois de Boulogne	PROPN	Bois de Boulogne
192	,	PUNCT	,
193	sous	ADP	sous
194	un	DET	un
195	doux	ADJ	doux
196	soleil	NOUN	soleil
197	prématuré	ADJ	prématuré
198	,	PUNCT	,
199	qui	SCONJ	qui
200	,	PUNCT	,
201	sur	ADP	sur
202	les	DET	le
203	bancs	NOUN	banc
204	,	PUNCT	,
205	dorlotait	VERB	dorloter
206	des	DET	un
207	lassitudes	NOUN	lassitude
208	;	PUNCT	;
209	des	DET	un
210	bébés	NOUN	bébé
211	faisaient	VERB	faire

212	des	DET	un
213	pâtés	NOUN	pâté
214	dans	ADP	dans
215	le	DET	le
216	sable	NOUN	sable
217	,	PUNCT	,
218	des	DET	un
219	hommes	NOUN	homme
220	sans	ADP	sans
221	paletot	NOUN	paletot
222	erraient	VERB	errer
223	,	PUNCT	,
224	et	CCONJ	et
225	je	PRO	je
226	n’	ADV	ne
227	ai	AUX	avoir
228	remarqué	VERB	remarquer
229	dans	ADP	dans
230	la	DET	le
231	blonde	ADJ	blond
232	et	CCONJ	et
233	jeune	ADJ	jeune
234	lumière	NOUN	lumière
235	que	CCONJ	que
236	la	DET	le
237	couleur	NOUN	couleur
238	et	CCONJ	et
239	l’	DET	le
240	élégance	NOUN	élégance
241	de	ADP	de
242	quelques	DET	quelque
243	nuques	NOUN	nuque
244	de	ADP	de
245	femmes	NOUN	femme
246	,	PUNCT	,
247	aux	ADP	à le
248	cheveux	NOUN	cheveu
249	retroussés	VERB	retrousser
250	et	CCONJ	et
251	frisottants	ADJ	frisottant
252	.	PUNCT	.
253	Au	ADP	à le
254	ciel	NOUN	ciel
255	,	PUNCT	,
256	au	ADP	à le
257	dessus	ADP	dessus
258	de	ADP	de
259	l’	DET	le
260	Arc de Triomphe	PROPN	Arc de Triomphe
261	,	PUNCT	,
262	dans	DANS	dans
263	l’	DET	le
264	azur	NOUN	azur

265	pâle	ADJ	pâle
266	,	PUNCT	,
267	une	DET	un
268	lune	NOUN	lune
269	diurne	ADJ	diurne
270	émiettait	VERB	émietter
271	son	DET	son
272	disque	NOUN	disque
273	de	ADP	de
274	pain à cacheter	NOUN	pain à cacheter
275	dissous	VERB	dissoudre
276	de	ADP	de
277	salive	NOUN	salive
278	.	PUNCT	.
279	Entre	ADP	Entre
280	les	DET	le
281	deux	NUM	deux
282	lumières	NOUN	lumière
283	d’	ADP	de
284	une	DET	un
285	table	NOUN	table
286	de	ADP	de
287	jeu	NOUN	jeu
288	,	PUNCT	,
289	m’	PRON	me
290	apparaît	VERB	apparaître
291	une	DET	un
292	tête	NOUN	tête
293	de	ADP	de
294	jeune	ADJ	jeune
295	fille	NOUN	fille
296	,	PUNCT	,
297	dans	ADP	dans
298	la	DET	le
299	clarté	NOUN	clarté
300	qui	SCONJ	qui
301	baigne	VERB	baigner
302	ses	DET	son
303	traits	NOUN	trait
304	.	PUNCT	.
305	C’	PRON	Ce
306	est	AUX	être
307	ce	PRON	ce
308	qu’	CCONJ	que
309	on	PRON	on
310	appelle	VERB	appeler
311	une	DET	un
312	belle	ADJ	beau
313	fille	NOUN	fille
314	,	PUNCT	,
315	aux	ADP	à le
316	traits	NOUN	trait
317	durs	ADJ	dur

318	,	PUNCT	,
319	hommasses	ADJ	hommasse
320	,	PUNCT	,
321	mais	CCONJ	mais
322	d’	ADP	de
323	un	DET	un
324	aspect	NOUN	aspect
325	de	ADP	de
326	visage	NOUN	visage
327	irrémédiablement	ADV	irrémédiablement
328	sale	ADJ	sale
329	.	PUNCT	.
330	Le	DET	le
331	duvet	NOUN	duvet
332	qui	SCONJ	qui
333	s’	PRON	se
334	accentue	VERB	accentuer
335	sous	ADP	sous
336	le	DET	le
337	nez	NOUN	nez
338	a	AUX	avoir
339	des	DET	un
340	noirceurs	NOUN	noirceur
341	et	CCONJ	et
342	des	DET	un
343	piqûres	NOUN	piqûre
344	de	ADP	de
345	barbe	NOUN	barbe
346	,	PUNCT	,
347	mal	ADV	mal
348	rasée	VERB	raser
349	,	PUNCT	,
350	et	CCONJ	et
351	dans	ADP	dans
352	la	DET	le
353	peau	NOUN	peau
354	des	ADP	de le
355	joues	NOUN	joue
356	,	PUNCT	,
357	les	DET	le
358	lentilles	NOUN	lentille
359	brunes	ADJ	brun
360	de	ADP	de
361	têtes	NOUN	tête
362	de	ADP	de
363	boutons	NOUN	bouton
364	arrachés	VERB	arracher
365	,	PUNCT	,
366	et	CCONJ	et
367	de	ADP	de
368	la	DET	le
369	coiffure	NOUN	coiffure
370	sur	ADP	sur

371	le	DET	le
372	front	NOUN	front
373	dégringole	VERB	dégringoler
374	une	DET	un
375	mèche	NOUN	mèche
376	de	ADP	de
377	cheveux	NOUN	cheveu
378	durs	ADJ	dur
379	et	CCONJ	et
380	noirs	ADJ	noir
381	.	PUNCT	.
382	Mardi	NOUN	mardi
383	8	NUM	8
384	Février	NOUN	février
385	.	PUNCT	.
386	Mariage	NOUN	Mariage
387	de	ADP	de
388	Dorchain	PROPN	Dorchain
389	à	ADP	à
390	Ste Clothilde	PROPN	Ste Clothilde
391	.	PUNCT	.
392	La	DET	Le
393	mariée	NOUN	marié
394	:	PUNCT	:
395	une	DET	un
396	brune	ADJ	brun
397	figure	NOUN	figure
398	allongée	VERB	allonger
399	aux	ADP	à le
400	lèvres	NOUN	lèvre
401	entrouvertes	VERB	entrouvrir
402	irrémédiablement	ADV	irrémédiablement
403	.	PUNCT	.
404	Un	DET	Un
405	cortège	NOUN	cortège
406	restreint	ADJ	restreint
407	.	PUNCT	.
408	Sully Prudhomme	PROPN	Sully Prudhomme
409	,	PUNCT	,
410	un peu	ADV	un peu
411	voûté	VERB	voûter
412	,	PUNCT	,
413	l’	DET	le
414	air	NOUN	air
415	las	ADJ	las
416	,	PUNCT	,
417	Coppée	PROPN	Coppée
418	,	PUNCT	,
419	avec	ADP	avec
420	un	DET	un
421	gros	ADJ	gros
422	paletot	NOUN	paletot
423	,	PUNCT	,

424	et	CCONJ	et
425	sa	PRON	son
426	régulière	ADJ	régulier
427	et	CCONJ	et
428	imberbe	ADJ	imberbe
429	figure	NOUN	figure
430	teintée	VERB	teinter
431	de	ADP	de
432	pain d’épice	NOUN	pain d’épice
433	.	PUNCT	.
434	Dans	ADP	Dans
435	l’	DET	le
436	église	NOUN	église
437	,	PUNCT	,
438	parmi	ADP	parmi
439	des	DET	un
440	Dames	NOUN	dame
441	élégantes	ADJ	élégant
442	,	PUNCT	,
443	des	DET	un
444	littérateurs	NOUN	littérateur
445	:	PUNCT	:
446	Banville	PROPN	Banville
447	,	PUNCT	,
448	un	DET	un
449	paletot	NOUN	paletot
450	jusqu’	ADV	jusqu’
451	aux	ADP	à le
452	talons	NOUN	talon
453	,	PUNCT	,
454	le	DET	le
455	cou	NOUN	cou
456	engoncé	VERB	engoncer
457	d’	ADP	de
458	un	DET	un
459	foulard	NOUN	foulard
460	blanc	ADJ	blanc
461	,	PUNCT	,
462	une	DET	un
463	tête	NOUN	tête
464	chauve	ADJ	chauve
465	,	PUNCT	,
466	et	CCONJ	et
467	sa	DET	son
468	face	NOUN	face
469	rase	ADJ	ras
470	de	ADP	de
471	vieux	ADJ	vieux
472	Pierrot	PROPN	Pierrot
473	;	PUNCT	;
474	Lemerre	PROPN	Lemerre
475	lippu	ADJ	lippu
476	et	CCONJ	et

477	massif	ADJ	massif
478	;	PUNCT	;
479	Bourget	PROPN	Bourget
480	,	PUNCT	,
481	très	ADV	très
482	bien	ADJ	bien
483	mis	VERB	mettre
484	,	PUNCT	,
485	monocle	NOUN	monocle
486	à	ADP	à
487	l’	DET	le
488	œil	NOUN	œil
489	,	PUNCT	,
490	un peu	ADV	un peu
491	poseur	NOUN	poseur
492	,	PUNCT	,
493	des	DET	un
494	sourcils	NOUN	sourcil
495	froncés	VERB	froncer
496	,	PUNCT	,
497	l’	DET	le
498	air	NOUN	air
499	dur	ADJ	dur
500	et	CCONJ	et
501	faux	ADJ	faux
502	;	PUNCT	;
503	un	DET	un
504	sentimental	NOUN	sentimental
505	égoïste	ADJ	égoïste
506	,	PUNCT	,
507	bien	ADV	bien
508	d’aplomb	ADJ	d’aplomb
509	,	PUNCT	,
510	épaules	NOUN	épaule
511	larges	ADJ	large
512	.	PUNCT	.
513	Le	DET	Le
514	soir	NOUN	soir
515	,	PUNCT	,
516	travail	NOUN	travail
517	mauvais	ADJ	mauvais
518	.	PUNCT	.
519	Mardi	NOUN	Mardi
520	15	NUM	15
521	Février	NOUN	février
522	.	PUNCT	.
523	Chez	ADP	chez
524	Mallarmé	PROPN	Mallarmé
525	avec	ADP	avec
526	Vielé	PROPN	Vielé
527	:	PUNCT	:
528	il y a	VERB	il y a
529	Ghil	PROPN	Ghil

530	,	PUNCT	,
531	Beaumanoir	PROPN	Beaumanoir
532	,	PUNCT	,
533	Dujardin	PROPN	Dujardin
534	,	PUNCT	,
535	puis	ADV	puis
536	Duret	PROPN	Duret
537	–	PUNCT	–
538	cinquante	NUM	cinquante
539	ans	NOUN	ans
540	environ	ADV	environ
541	–	PUNCT	–
542	un peu	ADV	un peu
543	grisonnant	ADJ	grisonnant
544	,	PUNCT	,
545	nez	NOUN	nez
546	trop	ADV	trop
547	court	ADJ	court
548	,	PUNCT	,
549	trop	ADV	trop
550	distant	ADJ	distant
551	d’	ADP	de
552	une	DET	un
553	bouche	NOUN	bouche
554	à	ADP	à
555	lèvre	NOUN	lèvre
556	un peu	ADV	un peu
557	pendante	ADJ	pendant
558	–	PUNCT	–
559	on	PRON	on
560	parle	VERB	parler
561	de	ADP	de
562	Manet	PROPN	Manet
563	.	PUNCT	.
564	Je	PRON	Je
565	soupçonne	VERB	soupçonner
566	Manet	PROPN	Manet
567	d’	ADP	de
568	avoir	AUX	avoir
569	été	VERB	être
570	homme	NOUN	homme
571	si	ADV	si
572	charmant	ADJ	charmant
573	qu’	SCONJ	que
574	il	PRON	il
575	a	AUX	avoir
576	fasciné	VERB	fasciner
577	ses	DET	son
578	amis	NOUN	ami
579	jusqu’	ADV	jusqu’
580	à	ADP	à
581	leur	PRON	leur
582	inspirer	VERB	inspirer

583	de	ADP	de
584	l’	DET	le
585	admiration	NOUN	admiration
586	pour	ADP	pour
587	sa	DET	son
588	peinture	NOUN	peinture
589	–	PUNCT	–
590	et	CCONJ	et
591	maintenant	ADV	maintenant
592	grâce aux	ADP	grâce a
593	nombreuses	ADJ	nombreux
594	conversations	NOUN	conversation
595	où	SCONJ	où
596	il	PRON	il
597	fut	AUX	être
598	question	NOUN	question
599	de	ADP	de
600	lui	PRON	lui
601	ici	ADV	ici
602	,	PUNCT	,
603	je	PRON	je
604	vois	VERB	voir
605	assez	ADV	assez
606	bien	ADJ	bien
607	ce	PRON	ce
608	qu’	SCONJ	que
609	il	PRON	il
610	était	AUX	être
611	:	PUNCT	:
612	peintre	NOUN	peintre
613	indécis	ADJ	indécis
614	,	PUNCT	,
615	doutant	VERB	douter
616	,	PUNCT	,
617	préoccupé	VERB	préoccuper
618	de	ADP	de
619	succès	NOUN	succès
620	et	CCONJ	et
621	guettant	VERB	guetter
622	la	DET	la
623	venue	NOUN	venue
624	des	ADP	de le
625	Rothschild	PROPN	Rothschild
626	.	PUNCT	.
627	D’	ADP	De
628	amusants	ADJ	amusant
629	détails	NOUN	détail
630	sur	ADP	sur
631	Burty	PROPN	Burty
632	–	PUNCT	–
633	doué	VERB	douer
634	d’	ADP	de
635	un	DET	un

636	flair	NOUN	flair
637	puissant	ADJ	puissant
638	,	PUNCT	,
639	collectionneur	NOUN	collectionneur
640	enragé	VERB	enrager
641	jusqu’	ADV	jusqu’
642	à	ADP	à
643	se	PRON	se
644	priver	VERB	priver
645	pour	ADP	pour
646	les	DET	le
647	objets	NOUN	objet
648	d’	ADP	de
649	art	NOUN	art
650	des	ADP	de le
651	femmes	NOUN	femme
652	qu’	SCONJ	que
653	il	PRON	il
654	aimait	VERB	aimer
655	.	PUNCT	.

Résumé

J'ai développé un outil en Python afin de faciliter l'utilisation de plusieurs outils de traitement morphosyntaxique et textométrique sur plusieurs types de fichiers, y compris les fichiers XML. Cet outil fonctionne à l'aide d'une interface Web développée en PHP dont le but est de simplifier autant que possible l'utilisation de l'outil par un utilisateur non formé à l'utilisation d'outils informatiques complexes.

Mots clefs

TAL, étiquetage morphosyntaxique, textométrie, XML

Summary

Using Python, I developped a tool whose goal was to simplify the use of part-of-speech tagging and textometrical tools on multiple types of files, including XML files. This tool uses a Web interface written in PHP in order to ensure that even an user who is untrained in the use of complex technological tools can make use of it.

Keywords

NLP, Part-of-speech tagging, textometrics, XML