UNIVERSITÉ DE PARIS NANTERRE
Master 1 Traitement Automatique des Langues

Enrichissement de corpus
M1 TAL 2022-2023

Évaluation d'annotateurs automatiques

Xinhao Zhang, Kexin Gui, Sandra Jagodzinska, Léna Gaubert

Évaluation d'annotateurs automatiques

Xinhao Zhang, Kexin Gui, Sandra Jagodzinska, Léna Gaubert

16 mars 2023

Table des matières

1 Introduction

L'annotation morphosyntaxique est une étape importante dans le traitement automatique du langage naturel, qui consiste à étiqueter chaque mot d'un texte avec des informations avec par exemple la partie du discours correspondante ou bien encore le lemme. Dans le cadre de ce projet, nous avons exploré les fonctions de trois annotateurs automatiques : Spacy, SEM et Treetagger, en nous concentrant sur leur capacité à annoter les parties du discours et les lemmes de manière précise et efficace.

Spacy est un outil d'analyse de texte open source qui utilise des techniques d'apprentissage automatique pour annoter les textes avec des informations telles que les parties du discours, les entités nommées et les relations syntaxiques. SEM, quant à lui, est une bibliothèque open source développée par Yoan Dupont pour l'analyse sémantique des textes, qui utilise des techniques de deep-learning pour extraire des informations sémantiques et contextuelles à partir de textes. Enfin, Treetagger

est un logiciel d'étiquetage morphosyntaxique, utilisé pour l'analyse morphologique, le marquage morphosyntaxique et la lemmatisation des textes.

Dans ce rapport, nous présentons les résultats de notre analyse comparative des performances de ces trois annotateurs dans l'annotation morphosyntaxique. Nous avons examiné leur exactitude, et leur facilité d'utilisation, en nous concentrant sur leur capacité à annoter les parties du discours et les lemmes avec précision.

Nous espérons que ce rapport aidera les utilisateurs à choisir l'annotateur automatique le plus adapté à leurs besoins en matière d'annotation morphosyntaxique. En présentant les avantages et les inconvénients de chaque annotateur, nous souhaitons fournir aux utilisateurs des informations précieuses pour prendre des décisions éclairées sur l'utilisation de ces outils dans leurs projets de traitement automatique du langage naturel.

2 Choix et constitution du corpus

Le corpus traité est extrait du roman "Le Comte de Monte-Cristo" d'Alexandre Dumas. Il s'agit d'un dialogue entre le personnage principal Edmond Dantès et son compagnon de cellule, l'abbé Faria, qui lui explique comment il a passé ses années en prison. Le roman présente un large éventail de personnages, d'intrigues et de thèmes qui en font un excellent choix pour l'annotation. Les personnages sont complexes et bien développés, offrant de nombreuses occasions de marquer les relations, les émotions et les traits de personnalité. Les intrigues sont complexes et pleines de rebondissements, permettant aux annotateurs de marquer les événements et les retournements de situation importants. Les thèmes sont variés et incluent des sujets tels que la vengeance, la justice, la trahison, la loyauté, la famille et l'amitié.

Quant à notre travail d'annotation, nous pouvons facilement découvrir que le texte contient plusieurs caractéristiques qui nous semblent intéressantes :

- diversité des temps verbaux : passé simple, imparfait, présent, participes passé et aussi présent,
- globalité des entités nommées, telles que les titres des ouvrages cités : "Traité sur la possibilité d'une monarchie générale en Italie", les noms des personnages : Edmond Dantès, abbé Faria, et aussi les références à des personnages historiques ou des œuvres littéraires : duc de Beaufort, abbé Dubuquoi, Latude, Lavoisier, Cabanis, Thucydide, Xénophon, Plutarque, Tite-Live, Tacite, Strada, Jornandès, Dante, Montaigne, Shakespeare, Spinosa, Machiavel et Bossuet.

3 Méthodologie

Nous avons exclusivement travaillé sur notebooks : notre notebook principal, "annotation monte cristo" comprend l'annotation effectuée avec Spacy, ainsi que les résultats issus de la manipulation des données résultantes des annotations avec Treetagger et SEM.

Suite aux annotations effectuées avec les trois annotateurs automatiques mentionnés précédemment, nous avons réalisé une révision manuelle, de façon à finalement obtenir le *gold* (annotation parfaite) qui nous servira de standard (pour la comparaison des données) par la suite.

Afin de s'assurer de l'homogénéité de nos fichiers, toutes les sorties des annotateurs et les golds sont au format .csv. Cette homogénéité nous a notamment permis de pouvoir utiliser le même script pour l'ensemble de nos données, ce qui constitue un gain de temps et de ressources. Suite à la manipulation des données .csv avec pandas (module de Python), nous sommes parvenus à calculer les matrices de confusions, ainsi que la précision, le rappel, et la f-mesure.

Afin d'aller plus loin dans nos calculs et visualisation de données, nous avons également implémenté un second script (dans le notebook nommé "confusion matrix") qui réalise une grande matrice de confusion sur les résultats de l'annotation des parties du discours (*POS-tagging*). Ce notebook supplémentaire vient donc compléter le reste de notre travail.

4 Annotations

4.1 Annotation avec Spacy

L'avantage principal de Spacy est son utilisation, qui s'avère particulièrement accessible, même pour un utilisateur qui ne serait pas nécessairement familier avec la programmation. L'annotation se fait grâce au modèle de langue pré-entraîné en français, que l'on charge dans une variable, avant de l'appliquer à notre texte (le modèle s'applique au texte tel une fonction : on avait ainsi nlp(text)). Ce dernier a été préalablement pré-traité, afin de retirer les sauts de ligne marqués dans le texte.

Notre notebook permet par la suite l'affichage des résultats de l'annotation de Spacy : pour chaque token, on affiche l'étiquette POS assignée, le lemme correspondant, le tag associé, la dépendance, ainsi qu'une courte explication de l'étiquette POS donnée par Spacy. Cet affichage permet ainsi une première visualisation des résultats. Dans le cas où l'utilisateur ne serait pas familier avec Spacy, celui-ci lui permettra de se familiariser aux étiquettes propres au module.

Enfin, nous écrivons les résultats de l'annotation dans un fichier CSV, à l'aide de la bibliothèque csv de Python, et des fonctions propres à la création et l'écriture de fichiers.

4.2 Annotation avec Treetagger

Treetagger, c'est un outil disponible en ligne qui sert à l'annotation automatique d'un fichier textuel. Celui que nous avons utilisé est disponible sous le lien suivant : https ://cental.uclouvain.be/treetagger/ et permet d'annoter les fichiers de point de vue morpho-syntaxique (POS tagging). L'output de ce logiciel est un fichier d'extension .tsv dans lequel nous trouvons trois colonnes : la première avec un token, la deuxième avec une étiquette POS attribuée par l'outil à un token, et la troisième avec un lemme qui est une forme le plus basique, qui nous appelons aussi une forme de dictionnaire, d'une unité considère comme un token.

Avant de commencer le processus d'évaluation, il a fallu préparer le fichier que nous considérons comme parfaitement annoté et auquel nous allons référer les résultats de treetagger. Ce fichier s'appelle Gold. Pour le préparer, nous avons pris le résultat du treetagger et nous avons ensuite corrigé toutes les erreurs trouvées. Nous avons parcouru le résultat en corrigeant des fautes, en utilisant les étiquettes propres à ce logiciel, (voir, étiquettes de Treetagger).

Comme mentionné précédemment, il est essentiel pour se travail de s'assurer de l'homogénéité des formats des différents fichiers que nous allons manipuler. Ainsi, nous avons opté pour les traitements effectués manuellement suivants :

1. Changement de l'extension .tsv en .csv en utilisant l'option d'exportation vers .csv disponible dans tous les éditeurs de fichiers tabulaires possibles.

2. Dans les lignes où le treetagger a mal séparé les mots-token l'ajout d'une ligne vide à l'un des fichiers pour que le nombre final de lignes soit le même que dans le fichier Gold et la conséquence des token reste le même.

    - Si treetagger a séparé j'étais comme un token au lieu de deux tokens, nous avons ajouté un rang vide juste au dessus.
    - Si treetagger a séparé étais comme deux tokens au lieu d'un token, nous avons ajouté un rang vide dans le Gold.

4.3 Annotation avec SEM

SEM est également un étiqueteur disponible en ligne permettant de réaliser une annotation morpho-syntaxique et sémantique d'un texte brut. En tant que fonctionnalités supplémentaires, les parenthésages en chunks et l'annotation des entités nommées sont parallèlement susceptibles d'être effectués sur ce logiciel assez puissant.

Capable de produire des fichiers de sortie sous différents formats, tels que html, conll, json, SEM nous a mis à disposition un fichier texte suffisamment structuré (ce dernier dispose de deux colonnes "token", et "pos") comme fichier de sortie. En observant minutieusement la mise en page du texte obtenu, nous avons rapidement trouvé la piste afin de le convertir en fichier csv. Cette conversion fut possible à l'aide du script suivant. La conversion en fichier .csv est essentielle, car elle nous permet d'avoir un fichier manipulable avec le module pandas.

```
import csv

with open('/content/sem_lzFne2.text','r',encoding = 'utf-8') as infile:
  txt = infile.read()

header = ["TOKEN", "POS"]

with open('sem_output.csv', 'w', newline='') as outfile:
    writer = csv.writer(outfile)
    writer.writerow(header)
    for line in txt.split(' '):
        # on divise ici les deux attributs par l'espace
        fields = line.strip().split('/')
        writer.writerow(fields)
```

Fig. 1 : Script permettant la conversion au format .csv

En effet, la préparation d'un fichier GOLD (ou "gold standard") est une étape indispensable lors de l'évaluation d'un système d'étiquetage morpho-syntaxique comme SEM. Tout en absorbant des champs d'étiquettes prédéfinis[1] nous avons ensuite pu corriger toutes les erreurs potentielles commises par SEM, soit celle de tokenisation, soit celle de catégorisation (pos-tagging). Ayant découvert que des tokenizations différentes dans les deux fichiers ont entraîné des ennuis pendant la comparaison, il faudrait alors aligner les tokenisations pour les comparer rangée par rangée. À l'instar de notre travail avec Treetagger, nous avons utiliser la méthode d'ajout de rangs vides.

[1] Étiquettes SEM

5 Résultats

Chaque annotateur a finalement été évalué sous trois points de vue : la précision de l'extraction des tokens (*tokenisation*), l'attribution des étiquettes morpho-syntaxiques appropriées, ainsi que la reconnaissance correcte des lemmes (pour chaque token extrait).

Rappel des formules

$$RappelR = \frac{VP}{VP + FN}$$

$$PrcisionP = \frac{VP}{VP + FP}$$

$$F - mesure = 2 \times \frac{R \times P}{R + P}$$

5.1 Matrice de confusion : script

Afin de calculer le rappel, la précision, et la f-mesure, nous avons donc besoin de calculer la matrice de confusion correspondante, en particulier de trois mesures : VP, FN, et FP.

- unités **vrais positifs** qui sont toutes les unités pertinentes bien retournées par l'outil. Dans le cas de la tokenisation, il s'agirait d'un mot bien tokenisé. Pour l'évaluation du *POS-tagging* et de la lemmatisation, il s'agit d'une bonne étiquette morpho-syntaxique attribuée ou d'un bon lemme associé au token donné.

- unités **faux positifs** qui sont toutes les unités non-pertinentes retournées par l'outil. Ainsi, si l'annotateur considère « j'étais » comme un seul et unique token, alors que le Gold reconnaît deux tokens, « je » et « étais », alors le token « j'étais » de l'annotateur est un faux positif..

- unités **faux négatifs** qui sont toutes les unités pertinentes non-retournées par l'outil. Si on reprend l'exemple mentionné précédemment, « j'étais » sera comparé à « je », tandis que « étais » n'aura aucun token auquel il pourra être comparé. Ce token vide est donc un faux négatif.

Afin de collecter les valeurs de ces mesures automatiquement, nous avons créé un script qui parcourt les index des rangs, et les rangs des colonnes données. Si l'on souhaite évaluer l'étiquettage morpho-syntaxique par exemple, nous allons parcourir les colonnes pos et pos_gold, et $token$ et $token_gold$. On compare les éléments correspondants un à un, à chaque index. Afin d'attribuer à chaque résultat la catégorie correspondante, nous avons besoin des conditions suivantes :

- si à l'index i, le $token_i$ est égal au $token_i^{gold}$, et que le pos_i est également égal au pos_i^{gold}, alors nous avons trouvé un ***vrai positif*** .

- si, à l'index i, le $token_i$ ou le $token_i^{gold}$ est vide, ou alors si pos_i est différent de pos_i^{gold}, nous constatons que nous avons trouvé un ***faux positif*** .

- enfin, si à l'index i, $token_i$ est vide, alors que $token_i^{gold}$ ne l'est pas, et que pos_i est différent de pos_i^{gold}, alors nous avons trouvé un ***faux négatif*** .

L'algorithme présenté a finalement été implémenté sous forme de fonction : au total, trois fonctions nous permettent d'obtenir les résultats désirés.

- $confmatrix$: renvoie la matrice de confusion évaluant la lemmatisation ou le *POS-tagging* (au choix)
- $confmatrixtoken$: renvoie la matrice de confusion évaluant la tokenisation de l'annotateur évalué.
- $prfscore$: prend les vrais positifs VP, les faux négatifs FN, et les faux positifs FP en argument, renvoie la précision P, le rappel R, et la f-mesure.

5.2 Matrice de confusion : POS-tagging

Comme mentionné dans notre introduction, nous avons également implémenté un script réalisant des matrices de confusions évaluant explicitement l'étiquettage morpho-syntaxique de chacun des annotateurs. Ce script, plus complexe, utilise pandas, numpy, ainsi que les modules graphiques seaborn et matplotlib.pyplot de Python. Les résultats obtenus nous ont servis dans notre analyse des résultats obtenus par l'annotation morpho-syntaxique pour chaque annotateur.

6 Analyse des résultats

6.1 Spacy

Résultats obtenus :

- Pour l'annotation morpho-syntaxique :
  - $P = 0.909$
  - $R = 1.0$
  - $F - score = 0.952$
- Pour la lemmatisation :
  - $P = 0.909$
  - $R = 1.0$
  - $F - score = 0.952$
- Pour la tokenisation :
  - $P = 0.999$
  - $R = 1.0$
  - $F - score = 1.0$

Selon les résultats obtenus, Spacy semble être un annotateur performant pour la tokenisation, l'annotation morpho-syntaxique et la lemmatisation. En effet, pour la tokenisation, Spacy a obtenu un score parfait de 1 pour la précision et le rappel. Pour l'annotation morpho-syntaxique et la lemmatisation, Spacy a obtenu un score de précision de 0.909 et de rappel de 1.0, ainsi qu'une F-mesure de 0.952 pour les deux tâches. En regardant la fig.2, nous pouvons voir qu'une douzaine de verbes ont été étiquetés comme des noms, tandis que neuf pronoms sont étiquetés comme des déterminants.

Ces résultats suggèrent que Spacy est un choix solide pour les tâches de traitement du langage

naturel nécessitant une tokenisation précise, une annotation morpho-syntaxique fiable et une lemmatisation précise. Cependant, il est important de noter que les performances de Spacy peuvent varier en fonction des types de texte et des langues. En effet, pour chaque langue, le modèle donné par Spacy est pré-entraîné sur des données différentes, pas nécessairement comparables. Il est ainsi possible d'obtenir des résultats différents pour une autre langue. Nous considérons donc qu'il est essentiel de tester Spacy pour chaque utilisation spécifique.

En comparaison avec les autres annotateurs, Spacy semble avoir des performances similaires à Treetagger et SEM pour la lemmatisation et l'annotation morpho-syntaxique. Cependant, Treetagger a obtenu un score légèrement supérieur pour la tokenisation avec une précision de 0.995, un rappel de 0.996 et une F-mesure de 0.996. SEM a obtenu des scores légèrement inférieurs pour la tokenisation, avec une précision de 0.968, un rappel de 0.951 et une F-mesure de 0.959.

En conclusion, les résultats obtenus montrent que Spacy est un annotateur performant pour la tokenisation, l'annotation morpho-syntaxique et la lemmatisation. Il est important de considérer les performances de Spacy en fonction des types de texte et des langues, mais dans l'ensemble, Spacy semble être un choix solide pour de nombreuses tâches de traitement du langage naturel.

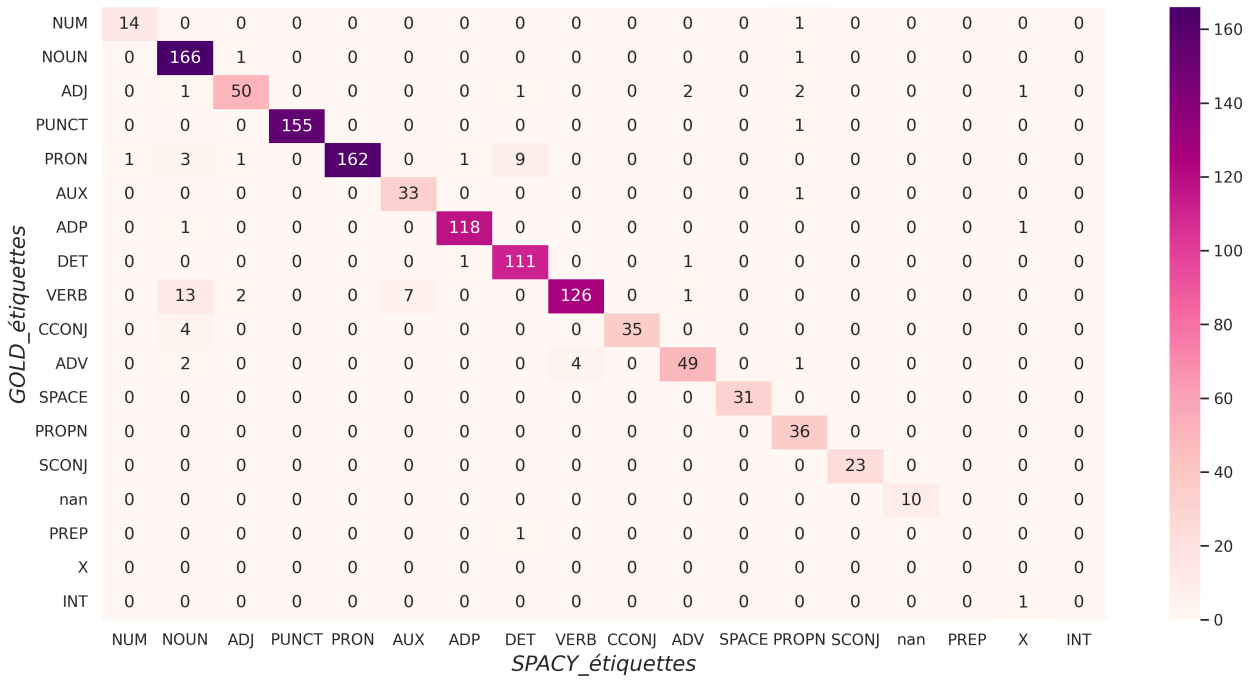

Fig. 2 : Matrice de confusion pour l'étiquettage morpho-syntaxique de Spacy

6.2 Treetagger

Résultats obtenus :

- Pour l'annotation morpho-syntaxique :
  - $P = 0.936$
  - $R = 0.996$

- $F - score = 0.965$

- Pour la lemmatisation :
  - $P = 0.936$
  - $R = 0.996$
  - $F - score = 0.965$

- Pour la tokenisation :
  - $P = 0.995$
  - $R = 0.996$
  - $F - score = 0.996$

Malgré une tokenisation qui a pu nous poser problème lors de la manipulation des données, Treetagger a l'avantage de présenter à son utilisateur un vaste éventail d'étiquettes morpho-syntaxique, ce qui rend l'annotation plus exhaustive, et plus précise comparativement à Spacy.

Si Treetagger nous donne la moins bonne tokenisation, c'est très certainement l'annotateur qui nous offre la meilleure annotation morpho-syntaxique. Dans le cas du POS tagging, il parvient à très bien maximiser le nombre de vrais positifs (d'où une valeur de R aussi proche de 1). Il minimise un peu moins bien le nombre de faux positifs (on a une précision de 0.936 pour un rappel de 0.996), mais le résultat finalement obtenu est très satisfaisant. Si on se penche à présent sur la Fig 3, la donnée qui frappe le plus est celle-ci : parmis les noms propres (annoté NAM dans Treetagger) figurants dans l'extrait choisit, 12 sont annotés comme noms communs (NOM), et 28 sont bien annotés comme noms propres.

De façon assez surprenante, les résultats obtenus pour la lemmatisation sont identiques. Les parties du discours dont la lemmatisation posait vraisemblablement le plus de problème à Treetagger étaient les verbes conjugués. (Par exemple, le token "répondit" avait pour lemme "répondit" et non "répondre".)

6.3 SEM

Résultats obtenus :

- Pour l'annotation morpho-syntaxique :
  - $P = 0.943$
  - $R = 0.95$
  - $F - score = 0.946$

- Pour la tokenisation :
  - $P = 0.968$
  - $R = 0.951$
  - $F - score = 0.959$

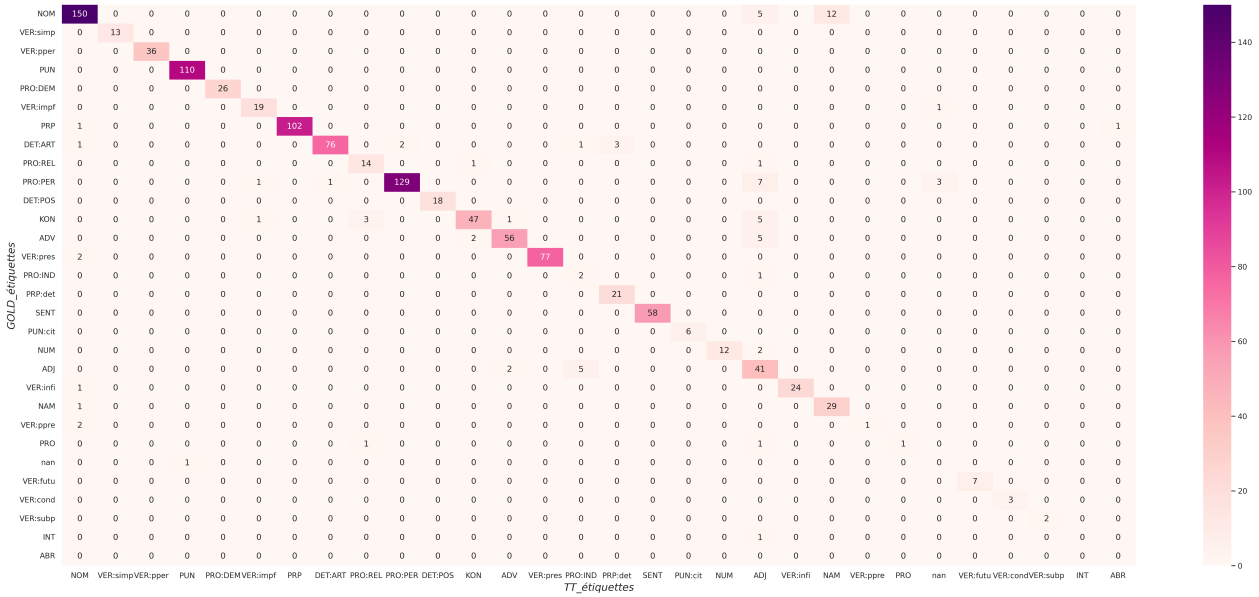

Fig. 3 : Matrice de confusion pour l'étiquettage morpho-syntaxique de Treetagger

En comparaison avec Treetagger et SPACY, SEM s'avère moins performant pour la segmentation de textes, comme en témoigne la valeur de Rappel relativement basse de 0.968. Nous supposons que cela est dû à la spécialisation de SEM dans d'autres fonctionnalités, telles que les chunks. En effet, le résultat renvoie des fois des syntagmes reconnus comme un seul token, tels que "en effet", "à peu près", etc. Par conséquent, conformément à notre algorithme défini, ce désavantage risque de dégrader considérablement la valeur de Rappel.

À notre grande surprise, la valeur de précision n'est pas pour autant forcément affecté, tandis qu'celle-ci de 0.943 est voire un peu plus élevée que celle de Treetagger. Il est donc démontré que SEM est capable de nous apporter des résultats relativement convaincants et satisaisants en termes d'annotation de POS tagging.

En ce qui concerne la matrice de confusion ci-dessous, la remarque évidente réside tout de même dans la colonne "nan", c'est-à-dire qu'il existe à peu près 40 valeurs non retournées dans ce cas. De plus, en nous focalisant sur la matrice liée à l'étiquette "NC", il semble que SEM rencontrent plus de difficultés pour l'annotation des noms communs : presque dix tokens sont mal interprétés. Enfin, il est intéressant de noter dans cette figure que l'annotateur a tendance à confondre adverbe avec préposition.

7 Comparaison et conclusion

Ce projet visait à évaluer les annotateurs automatiques disponibles en ligne. Nous étions curieux de voir à quel point les résultats obtenus seront adéquats, et dans quelle mesure la machine est capable d'obtenir des résultats similaires à l'annotation par un humain. Alors, nous avons testé trois annotateurs automatiques différentes, et nous avons tiré les conclusions suivantes :

- les trois annotateurs utilisent des étiquettes morpho-syntaxique différentes, et proposent une gamme et un nombre d'étiquettes différents, qui rend la comparaison des résultats entre les annotateur difficile.

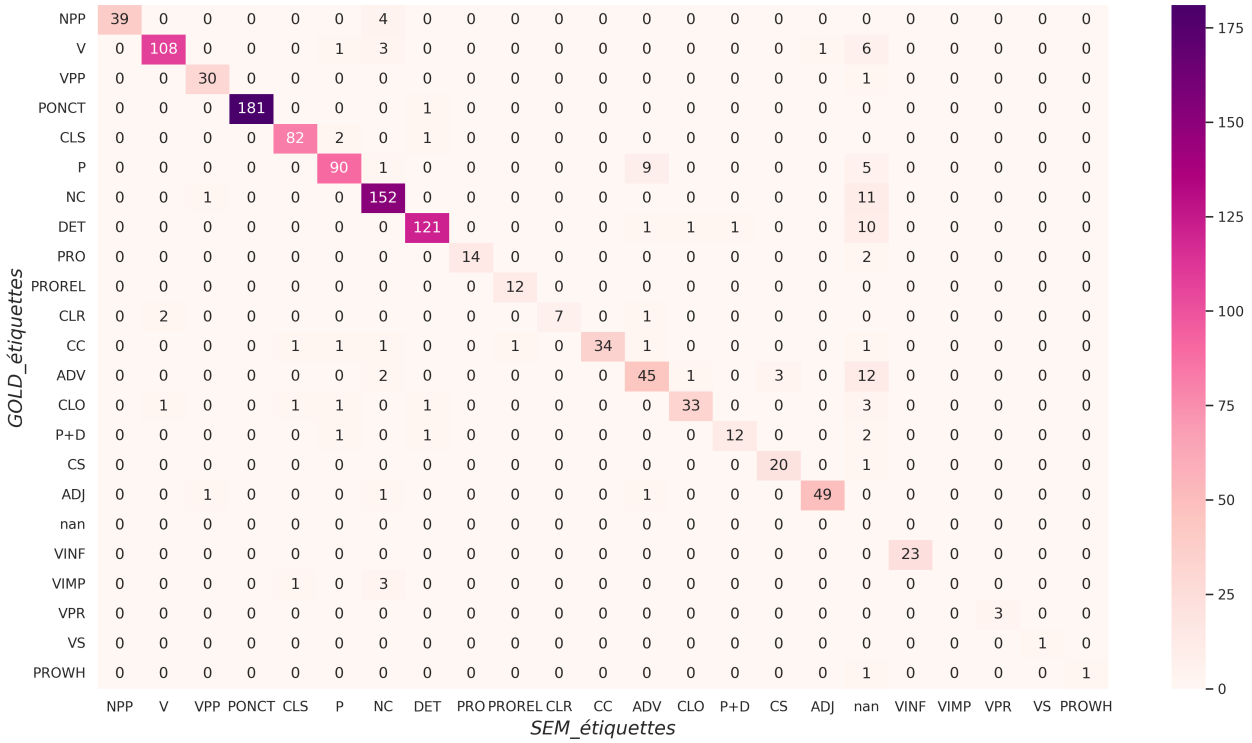

Fig. 4 : Matrice de confusion de l'étiquettage morpho-syntaxique de SEM

- L'annotateur le plus précis et le plus adéquat s'est avéré être le module de python Spacy. Cependant, la gamme d'étiquettes morpho-syntaxiques utilisées dans ce module est plus petite que celles de Treetagger et SEM, ce qui réduit considérablement le risque de confusion, en augmentant ainsi sa précision. De plus, ses étiquettes ne donnent que des informations de base sur les parties du discourse, et ne contiennent pas d'informations plus précise. Ex. : le mode ou le temps du verbe.

- SEM s'est également avéré être un outil très précis qui contient des informations plus précises dans ses étiquettes, quoique moins que Treetagger. Ses résultats étaient proches de ceux de Treetagger. Néanmoins il était bien meilleur pour annoter les noms propres.

- Contrairement aux autres outils, SEM ne proposent pas de lemmatisation. (Cela est valable uniquement pour la version en ligne. Après vérification, nous avons pu voir que la version source proposait une lemmatisation, mais nous n'avons pas pu tester cette fonction.)

- Treetagger s'est avéré être un outil très précis en termes d'attribution d'étiquettes de lemmatisation et de tokenisation. Son système d'étiquetage est le plus complet, ce qui souligne sa fiabilité.

- Dans les trois cas, l'obtention des résultats a été rapide et les trois outils se sont avérés efficaces. Cependant, il faut choisir un outil en fonction des besoins de l'utilisateur, SEM et TreeTagger fournissent des informations plus précises sur les unités que Spacy. Cependent, les multiples étiquettes dee TreeTagger peuvent être moins attrayantes pour un utilisateur qui n'a besoin que d'informations de base, semblable à ce que Spacy fournit de manière fiable à presque 100

- Étonnamment, il s'avère que ces outils ne diffèrent pas en précision, rappel et F-mesure les uns des autres dans une très large mesure. Tous les trois ont fourni des résultats satisfaisants. Il

convient de souligner cependant qu'ils sont différents et peuvent être utilisés pour des besoins et des tâches plus ou moins complexes.

Pour conclure, ce projet nous a permis d'explorer de façon exhaustive trois annotateurs automatiques bien différents, et d'en comparer les résultats. Il est évident que l'évaluation aurait pu être plus complète (bien plus d'outils statistiques sont à notre disposition, ici, nous avons préféré nous concentré sur la précision, le rappel et la f-mesure, car ceux-ci nous ont semblé suffisant). Par ailleurs, il serait également intéressant pour nous de comparer Spacy à NLTK, autre module de Python développé pour le TAL. Enfin, ce projet fut l'occasion pour nous de réaliser un premier vrai projet de TAL axé sur l'annotation automatique. Il nous a permis d'approfondir nombreuses de nos connaissances, et d'apprendre à maîtriser les outils que nous venons de vous présenter.