Sommaire

Vous retrouverez tout mes fichiers sur mon GitHub :
github.com/deboraptor/annotation_automatique

Note : je risque de faire des modifications fréquentes, les captures d'écran ne seront pas forcément à jour par rapport à mon code.

Introduction

Je vous conseille de lire tout d'abord le README.md pour comprendre comment utiliser et exécuter les programmes pythons. J'ai créé un environnement virtuel parce que j'avais un conflit entre la dernière version de NumPy (1.26) et sklearn.metrics, toutes les étapes sont disponibles sur le README.md.

On va chercher à faire une annotation automatique sur deux corpus différents. Le premier aura été extrait de la page wikipédia de l'acteur britannique Henry Cavill : https://fr.wikipedia.org/wiki/Henry_Cavill

Et le deuxième est une sélection des derniers tweets trouvé sur la page https://twitter.com avec le filtre de recherche "langage : français" et les mots clés "Henry Cavill".

Dans un deuxième temps, nous ferons l'évaluation de ces modèles avec les mesure de précision, rappel et f-mesure. Pour ce faire, il a fallu que je fasse une annotation manuel des résultats. Nous discuterons de ces résultats plus en détails à la fin.

Notez que j'ai essayé plusieurs modules python et que ça s'est terminé en échec. Nous parlerons aussi de tout cela dans ce document.

Premiers pas

Au premier regard, quand l'annotation est faite automatiquement on remarque qu'il y a des ponctuations et des espaces qui correspondent aux retours chariot. Je me suis posée la question de savoir s'il fallait faire un pré-traitement et j'ai essayé de le faire avec des regex pour enlever la ponctuation mais ce n'était pas concluant.

J'ai voulu faire un affichage graphique plus intéressant qu'un simple retour dans un terminal. J'ai donc pris le parti de faire une page html avec BeautifulSoup pour le gérer. J'ai eu beaucoup d'erreurs, notamment des doublons de tableaux et la mise en page de ces derniers qui m'ont pris pas mal de temps à gérer.

J'ai choisi de faire le test pour deux corpus différents : un normé et un non-normé. Pour chaque corpus, je me suis contraintes à plus de 500 mots.

Le corpus normé est extrait de la page Wikipédia sur Henry Cavill. Pour le corpus non-normé, j'ai choisi de prendre des tweets sur le site twitter. Je les ai extrait à la main puis mis bout à bout dans un fichier texte exploitable. J'ai trié les tweets de manière à enlever ceux qui avaient du contenu explicite (évidemment, le premier tweet que j'ai eu en contenait..) et j'ai retiré les émojis encodé en unicode qui posaient un problème d'encodage.

SpaCy

Observation des résultats de SpaCy

Texte normé, wikipédia

Le texte ici est normé, c'est à dire qu'il répond aux exigences de la grammaire et de la conjugaison. Ce sont donc des mots qui apparaissent dans le dictionnaire et qui sont déjà connu de la machine. On peut donc imaginer que les mots qui pourront poser problème sont donc les mots qui n'apparaissent pas dans celui-ci comme les noms propres. On peut aussi se dire que les références, qui sont collées aux mots (romaine2, Cavill"3), vont avoir du mal à être annoté correctement aussi.

Texte non normé, tweets

Comme on pouvait s'y attendre, les résultats avec SpaCy sont moins concluants avec un texte non normé. En effet, nous pouvons observer :

- des fautes d'orthographe : geniale, synchroniser
- des hashtags : #TheMinistryofUngentlemanlyWarfare, #GuyRitchie
- des abréviations type SMS : pcq, tjs
- des onomatopées : krr, bah
- des expressions populaires chez les jeunes : fav, claqué, j'suis, crush
- des anglicismes : reboot, hyper, acting
- des émojis en ASCII : o_o

SpaCy a bien reconnu certains mots grâce à leur position dans la phrase. Par exemple, dans "c'est vraiment mon crush" le mot crush a bien été catégorisé par un nom car il est précédé par un déterminant. C'est aussi le cas pour "l'ont fancast" qui a bien été catégorisé en tant que verbe. Cependant, l'émoji "o_o" a été catégorisé en verbe, ce qui n'est pas du tout le cas.

Nous reviendrons plus en détail sur comment j'ai décidé de les annoter lors de mon annotation manuelle.

Annotation manuelle

Texte normé, wikipédia

J'ai choisi d'annoter les phrases en anglais avec leur vrai catégorie grammaticale et pas X parce que j'ai observé que SpaCy le faisait de lui-même, comme dans le syntagme "man of steel" annoté comme ça par la machine :
('Man', 'man', 'NOUN'), ('of', 'of', 'X'), ('Steel', 'Steel', 'PROPN')

Texte non normé, tweets

J'ai dû faire un choix pour les résultats et mon choix s'est scindé en deux catégorie que je vais donc expliciter ci-dessous.

- pour les hashtags, émojis et les onomatopées, j'ai décidé de les annoter en tant que X.
- les abréviations, les fautes d'orthographes, les anglicisme et les expressions populaires chez les jeunes seront annotées par la catégorie du mot qu'elles représentent.
    + mots spécifiques ambigu :
        - "bah" annoté en tant qu'interjection (INTJ).

Différences : résultats observés

J'ai utilisé le module difflib qui est très simple d'utilisation : il faut indiquer deux chaînes de caractères ou deux listes à comparer. C'est la méthode ndiff() qui s'en occupe et après j'ai mis les résultats dans des listes pour pouvoir les afficher dans un tableau.

```
diff = difflib.ndiff(resultats_norme_machine, resultats_norme_moi)

      mon_changement = []
```

```
      mal_annote_machine = []

      for ligne in diff:
      if ligne[0] == " ":
            continue
      if ligne[0] == "+":
            mon_changement.append(ligne[2:])
      elif ligne[0] == "-":
            mal_annote_machine.append(ligne[2:])
```

Pour cette partie, j'ai voulu tout mettre dans des tableaux html et afficher en rouge les différences et en vert les similitudes cependant je n'ai pas réussi et j'ai perdu énormément de temps à essayer de trouver une solution, en vain. J'ai donc décidé de seulement afficher dans le terminal les résultats, tout de même sous forme de tableau.

J'ai donc pour cette partie utilisé le module Pretty Table qui permet d'afficher sous forme de tableau ASCII des données. Voici les résultats.

Tableau texte normé

```
+-----------------------------------------+-----------------------------------------+
|          Mon changement manuel          |        Mal annoté par la machine        |
+-----------------------------------------+-----------------------------------------+
|       Nombre de lignes différentes      |                   26                    |
|                                         |                                         |
|        ('Tudors', 'tudor', 'PROPN')     |       ('Tudors', 'tudor', 'NOUN')       |
|         ('puis', 'puis', 'CCONJ')       |       ('puis', 'pouvoir', 'CCONJ')      |
|            ('of', 'of', 'ADP')          |            ('of', 'of', 'X')            |
|              ('v', 'v', 'X')            |            ('v', 'v', 'AUX')            |
|          ('Aube', 'aube', 'NOUN')       |         ('Aube', 'Aube', 'PROPN')       |
|       ('Justice', 'justice', 'NOUN')    |      ('Justice', 'Justice', 'PROPN')    |
|        ('League', 'league', 'NOUN')     |       ('League', 'League', 'PROPN')     |
|      ("Snyder'", "Snyder'", 'PROPN')    |       ("Snyder'", "snyder'", 'DET')     |
|       ('Justice', 'justice', 'NOUN')    |      ('Justice', 'Justice', 'PROPN')    |
|       ('League', 'league', 'PROPN')     |       ('League', 'League', 'PROPN')     |
|        ('Enola', 'Enola', 'PROPN')      |         ('Enola', 'enola', 'NOUN')      |
|        ('Enola', 'Enola', 'PROPN')      |         ('Enola', 'enola', 'NOUN')      |
|           ('de', 'de', 'PROPN')         |            ('de', 'de', 'ADP')          |
|           ('The', 'The', 'DET')         |            ('The', 'The', 'X')          |
|       ('Witcher', 'Witcher', 'NOUN')    |      ('Witcher', 'Witcher', 'PROPN')    |
|      ('Jeunesse', 'Jeunesse', 'NOUN')   |     ('Jeunesse', 'Jeunesse', 'PROPN')   |
|           ('îles', 'île', 'NOUN')       |           ('îles', 'île', 'ADJ')        |
| ('anglo-normandes', 'anglo-normand', 'ADJ') | ('anglo-normandes', 'anglo-normand', 'NOUN') |
|     ('Marianne', 'marianne', 'PROPN')   |      ('Marianne', 'marianne', 'NOUN')   |
|      ('anglaise', 'anglaise', 'ADJ')    |      ('anglaise', 'anglaise', 'NOUN')   |
|           ('issu', 'issu', 'ADJ')       |           ('issu', 'issu', 'VERB')      |
|           ('St.', 'st.', 'PROPN')       |           ('St.', 'st.', 'NOUN')        |
|     ('Michael'', 'Michael'', 'PROPN')   |      ('Michael'', 'michael'', 'ADJ')    |
|  ('Preparatory', 'preparatory', 'PROPN') |   ('Preparatory', 'preparatory', 'VERB') |
|      ('souffre', 'souffrir', 'VERB')    |       ('souffre', 'souffre', 'NOUN')    |
|           ('Fat', 'fat', 'PROPN')       |           ('Fat', 'fat', 'NOUN')        |
+-----------------------------------------+-----------------------------------------+
```

J'ai décidé de rajouter de la couleur pour que ce soit moins austère avec colortable. Notez que les résultats ont peut-être évolué depuis le temps donc je vous conseille d'aller tester

le script par vous-même ! Nous pouvons très clairement voir toutes les différences que la machine à trouvé. Je n'ai pas eu de problème pour l'affichage de ce tableau.

La grande majorité des différences vient donc bel et bien des noms propres mal annotés, le modèle considère souvent qu'une majuscule en début de mot est un nom propre. Cas isolé pour Tudors, Marianne, Michael et Enola qu'il a annoté en nom, ce qui est assez étrange car Marianne et Michael sont des prénoms communs.

Les références qui apparaissent dans le texte Wikipédia (Cavill3) n'ont finalement posé aucun problème, la machine les a totalement ignoré.

Tableau texte non normé

Mon changement manuel	Mal annoté par la machine
Nombre de lignes différentes	48
décris décrire VERB	décris décri ADJ
pcq parce que SCONJ	pcq pcq ADV
ai avoir AUX	ai avoir VERB
coupe couper VERB	coupe coupe VERB
Queen queen PROPN	Queen queen NOUN
genre genre VERB	genre genr ADJ
hype hyper VERB	hype hype ADP
Subjectivement Subjectivement ADV	Subjectivement Subjectivement PROPN
fav favori ADJ	fav fav NOUN
Kent Kent PROPN	Kent kent ADV
# # X	# # PROPN
TheMinistryofUngentlemanlyWarfare theministryofungentlemanlywarfare X	TheMinistryofUngentlemanlyWarfare theministryofungentlemanlywarfare ADP
# # X	# # PROPN
GuyRitchie guyritchie PROPN	GuyRitchie guyritchie NOUN
Santa santa PROPN	Santa santa NOUN
tjs toujours ADV	tjs tjs DET
torse torse NOUN	torse tors ADJ
nu nu VERB	nu nu CCONJ
avoir avoir AUX	avoir avoir VERB
krr krr X	krr krr ADJ
krr krr X	krr krr VERB
Tudors tudor PROPN	Tudors tudor NOUN
Epoque Epoque NOUN	Epoque Epoque PROPN
o_o o_o X	o_o o_o VERB
2009 2009 NUM	2009 2009 NOUN
- - PUNCT	- - NOUN
2010 2010 NUM	2010 2010 NOUN
poils poil NOUN	poils poil ADJ
coquille coquille NOUN	coquille coquill ADJ
palmarès palmarès NOUN	palmarès palmarès ADV
visuellement visuellement ADV	visuellement visuellement NOUN
perso personnage NOUN	perso perso PRON
pue puer VERB	pue pu ADJ
HENRY Henry PROPN	HENRY henry VERB
CAVILL Cavill PROPN	CAVILL cavill PRON
j' je PRON	j' j' PROPN
billy billy PROPN	billy billy NOUN
henry henry PROPN	henry henry PRON
cavill cavill PROPN	cavill cavill VERB
rigolaient rigoler VERB	rigolaient rigoler ADV
Dis dire VERB	Dis Dis PROPN
- - PUNCT	- - PROPN
toi toi PRON	toi toi NOUN
fancast fancaster VERB	fancast fancast VERB
Ritchie ritchie PROPN	Ritchie ritchie NOUN
queen queen PROPN	queen queen ADJ
stylé stylé ADJ	stylé stylé NOUN
Bah Bah INTJ	Bah Bah PROPN

Pour le deuxième tableau, j'ai eu un problème avec le type de données dans la liste. En effet, l'erreur m'indiquait que c'était des tuples et non pas des strings. J'ai donc simplement fait la conversion, mais je ne comprend pas pourquoi c'est différent alors que mon code pour le texte non normé est strictement identique au texte normé. C'est pour cette raison que le tableau s'affiche légèrement différemment, mais le principe est le même.

On a donc bel et bien plus d'erreurs de la part de la machine pour le corpus non normé (48 contre 26). Comme on pouvait se l'attendre, les abréviations type "SMS" n'ont pas été reconnue par la machine. On peut voir qu'elle tente des choses car la machine annote comme adjectif le premier "krr krr" et met en verbe le second, ce qui est un schéma typique des langues SVO comme le français.

Pour les hashtags, la machine a réagit étrangement et a annoté le # en nom propre puis son contenu tantôt en adposition, tantôt en nom. J'imagine qu'elle a essayé de respecter un ordre SVO mais qu'elle a été larguée et qu'elle a mis plutôt au hasard.

Stanza

J'ai voulu installer le module lorsque….

ERROR: Could not install packages due to an OSError: [Errno 28] No space left on device

Je n'ai plus de place sur ma partition, et comme c'est sur Linux je n'avais jamais géré ça. J'ai donc dû me renseigner pour ajouter de l'espace à ma partition.

Pour Stanza, il faut télécharger le corpus en français via python. Il faut donc écrire la commande suivante : `stanza.download('fr')`. Un seul téléchargement suffit, on peut donc effacer cette ligne de commande après le téléchargement.

J'ai copié collé le code de SpaCy pour travailler parce dessus parce que je veux que les tableaux html soient similaires, j'ai entendu dire que Stanza et spaCy fonctionnaient exactement de la même manière donc j'ai fait de cette manière, mais j'ai eu des erreurs comme :

```
AttributeError: 'Token' object has no attribute 'pos'
```

Et ce… pour pos mais aussi pour lemma, pour deprel et pour text. J'ai contré cette erreur avec :

```
for sentence in doc.sentences:
      for token in sentence.tokens:
            ligne_token = getattr(token, 'text', '')
            ligne_lemme = getattr(token, 'lemma', '')
            ligne_pos = getattr(token, 'pos', '')
            ligne_dep = getattr(token, 'deprel', '')
```

Mes listes de résultats n'avaient pas la même longueur, j'ai donc dû faire des adaptations pour que ça fonctionne, mais mon premier essai n'affichait que le tokens.

Après un moment à essayer de débugger le code, j'ai réussi à avoir quelque chose de satisfaisant.

NLTK

J'ai commencé par regarder le module NLTK. J'avais immédiatement remarqué qu'il fallait importer énormément de modules différents, ce qui peut porter à confusion.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.parse.corenlp import CoreNLPServer
from nltk.parse.corenlp import CoreNLPParser
```

Ensuite, il faut télécharger des ressources en mettant la commande sur Python. C'est utile de ne le faire qu'une seule fois, mais si je dois partager mon code il vaut mieux le laisser. Or, si je le laisse, il télécharge les ressources à chaque exécution du script…

Globalement, tout se passait bien jusqu'à que je décide d'afficher aussi les DEP. Pour obtenir les dépendances avec NLTK, il est nécessaire d'ouvrir un serveur, ce qui s'avère assez compliqué si je voulais partager mon code pour qu'il fonctionne également sur votre machine.

En bref, NLTK a l'air très complet mais assez complexe et j'ai décidé de le laisser de côté pour ce devoir, mais je me suis promise de retravailler dessus une prochaine fois.

Tree Tagger

J'ai eu énormément de problèmes avec le module TreeTaggerWrapper. Tout d'abord, c'est un module à installer à la main, ce qui n'est pas si grave. Sauf que dans le cas de Tree Tagger, il y a plusieurs fichiers à installer, et il faut les piocher sur un site qui a passé l'âge. Cela aurait été plus pratique d'avoir un fichier zip avec tout le nécessaire plutôt que d'avoir à aller chercher.

Mon module n'a tout simplement jamais fonctionné, j'ai passé beaucoup de temps à essayer de régler le problème. Voici le message d'erreur qui ne cessait de s'afficher peut importe la solution que j'essayais d'apporter :

```
treetaggerwrapper.py:739: FutureWarning: Possible nested set at position 8
punct2find_re = re.compile("([^ ])([[" + ALONEMARKS + "])",
/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py:
2043: FutureWarning: Possible nested set at position 152 DnsHostMatch_re =
re.compile("(" + DnsHost_expression + ")",
/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py:
2067: FutureWarning: Possible nested set at position 409 UrlMatch_re =
re.compile(UrlMatch_expression, re.VERBOSE | re.IGNORECASE)
/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py:
2079: FutureWarning: Possible nested set at position 192 EmailMatch_re =
re.compile(EmailMatch_expression, re.VERBOSE | re.IGNORECASE) Traceback
(most recent call last): File
"/home/debora/Desktop/Cours/S2/enrichissement_de_corpus/dm_annotation_autom
atique/annotation_automatique/projet/dm_corpus_spacy.py", line 147, in
<module> tagger = treetaggerwrapper.TreeTagger(TAGLANG='fr',
TAGDIR="../modele_treetagger") File
"/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py
", line 1006, in __init__ self._set_tagger(kargs) File
"/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py
", line 1106, in _set_tagger raise TreeTaggerError("TreeTagger parameter
file invalid: " + \ treetaggerwrapper.TreeTaggerError: TreeTagger parameter
file invalid: french.par
```

J'ai tenté plusieurs solutions et en voici un aperçu :

- Donner le chemin du dossier du module

```
tagger = treetaggerwrapper.TreeTagger(TAGLANG='fr',
TAGDIR="/home/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper-2.
3.egg-info)
```

- Renommer les fichiers french.par et french-chunker.par par french-utf8.par selon ce que j'avais pu trouver dans des forums
- Installer des versions ultérieures pour vérifier la compatibilité

Comme j'utilise un environnement virtuel, je me suis dis que c'était possiblement la cause, et étant donné que je vous conseille également de faire ça, j'ai pensé qu'il vallait mieux laisser tomber TreeTagger.

Note à moi-même…

- ☐ pourquoi cette erreur ? peut-être que le module bug avec git ou un environnement virtuel
- ☐ essayer de copier coller tout les fichiers du module dans mon git
- ☐ regarder si une extension existe pour pour trouver tout les modules dans mon arborescence

Trankit

Ma tentative

Au début, j'ai eu un problème pour afficher les pos, et après je n'arrivais pas à afficher les lemmes. Si j'avais les lemmes, je n'avais pas les pos, et vice-versa. Pour comprendre mes dictionnaires, j'ai donc fait des prints pour vérifier où ils étaient.

```
print(phrase)
'id': 1, 'text': 'Henry William Dalgliesh Cavill, né le 5 mai 1983 à
Saint-Hélier, est un acteur britannique.', 'dspan': (0, 91), 'tokens':
[{'id': 1, 'text': 'Henry', 'upos': 'PROPN', 'head': 16, 'deprel': 'nsubj',
'dspan': (0, 5), 'span': (0, 5)}, {'id': 2, 'text': 'William', 'upos':
'PROPN', 'head': 1, 'deprel': 'flat:name', 'dspan': (6, 13), 'span': (6,
13)}, {'id': 3, 'text': 'Dalgliesh', 'upos': 'PROPN', 'head': 1, 'deprel':
'flat:name', 'dspan': (14, 23), 'span': (14, 23)}, {'id': 4, 'text':
'Cavill', 'upos': 'PROPN', 'head': 1, 'deprel': 'flat:name', 'dspan': (24,
30), 'span': (24, 30)}, {'id': 5, 'text': ',', 'upos': 'PUNCT', 'head': 6,
'deprel': 'punct', 'dspan': (30, 31), 'span': (30, 31)}, {'id': 6, 'text':
'né', 'upos': 'VERB', 'feats':
'Gender=Masc|Number=Sing|Tense=Past|VerbForm=Part', 'head': 1, 'deprel':
'acl', 'dspan': (32, 34), 'span': (32, 34)}, {'id': 7, 'text': 'le',
'upos': 'DET', 'feats':
'Definite=Def|Gender=Masc|Number=Sing|PronType=Art', 'head': 8, 'deprel':
'det', 'dspan': (35, 37), 'span': (35, 37)}, {'id': 8, 'text': '5', 'upos':
```

```
'NUM', 'head': 6, 'deprel': 'obl:arg', 'dspan': (38, 39), 'span': (38,
39)}, {'id': 9, 'text': 'mai', 'upos': 'NOUN', 'feats':
'Gender=Masc|Number=Sing', 'head': 8, 'deprel': 'nmod', 'dspan': (40, 43),
'span': (40, 43)}, {'id': 10, 'text': '1983', 'upos': 'NUM', 'head': 9,
'deprel': 'nmod', 'dspan': (44, 48), 'span': (44, 48)}, {'id': 11, 'text':
'à', 'upos': 'ADP', 'head': 12, 'deprel': 'case', 'dspan': (49, 50),
'span': (49, 50)}, {'id': 12, 'text': 'Saint-Hélier', 'upos': 'PROPN',
'head': 6, 'deprel': 'obl:arg', 'dspan': (51, 63), 'span': (51, 63)},
{'id': 13, 'text': ',', 'upos': 'PUNCT', 'head': 6, 'deprel': 'punct',
'dspan': (63, 64), 'span': (63, 64)}, {'id': 14, 'text': 'est', 'upos':
'AUX', 'feats': 'Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin',
'head': 16, 'deprel': 'cop', 'dspan': (65, 68), 'span': (65, 68)}, {'id':
15, 'text': 'un', 'upos': 'DET', 'feats':
'Definite=Ind|Gender=Masc|Number=Sing|PronType=Art', 'head': 16, 'deprel':
'det', 'dspan': (69, 71), 'span': (69, 71)}, {'id': 16, 'text': 'acteur',
'upos': 'NOUN', 'feats': 'Gender=Masc|Number=Sing', 'head': 0, 'deprel':
'root', 'dspan': (72, 78), 'span': (72, 78)}, {'id': 17, 'text':
'britannique', 'upos': 'ADJ', 'feats': 'Gender=Masc|Number=Sing', 'head':
16, 'deprel': 'amod', 'dspan': (79, 90), 'span': (79, 90)}, {'id': 18,
'text': '.', 'upos': 'PUNCT', 'head': 16, 'deprel': 'punct', 'dspan': (90,
91), 'span': (90, 91)}]}
```

On voit ici qu'il n'y a pas les lemmes, ci-dessous le print d'un token pour que ce soit plus lisible.

```
print(token)
{'id': 18, 'text': '.', 'upos': 'PUNCT', 'head': 16, 'deprel': 'punct',
'dspan': (90, 91), 'span': (90, 91)}
```

De toute évidence, ils ne sont pas dans token, j'ai onc dû louper quelque chose. Sur la documentation, ils disent de faire ça :

```
lem = pipe.lemmatize(texte)
```

Donc j'ai imprimé lem pour vérifier.

```
print(lem)
{'text': 'Henry William Dalgliesh Cavill, né le 5 mai 1983 à Saint-Hélier,
est un acteur britannique.\n\nIl est principalement connu pour son rôle de
Charles Brandon dans Les Tudors puis de Superman dans cinq films de
l\'univers cinématographique DC : Man of Steel (2013), Batman v Superman :
```

```
L\'Aube de la justice (2016), Justice League (2017), Zack Snyder\'s Justice League (2021) et Black Adam (2022), dont trois réalisés par Zack Snyder. Il incarne aussi Sherlock Holmes dans les films Netflix, Enola Holmes et Enola Holmes 2, respectivement sortis en 2020 et 2022. De 2019 à 2023, il est l\'interprète de Geralt de Riv, personnage principal de la série The Witcher, diffusée sur Netflix.\nBiographie\nJeunesse et formation\n\nOriginaire de Jersey, dans les îles anglo-normandes, Henry Cavill y a grandi avec ses deux parents ; Marianne Dalgliesh, secrétaire dans une banque, d\'origine irlandaise, écossaise et anglaise, et Colin Cavill, agent de change, originaire de Chester1, ainsi qu\'avec ses quatre frères ; Piers Cavill, Niki Richard Cavill (né en 1975), Simon Cavill (né en 1979) et Charlie Cavill (né en 1988). Il est issu d\'une famille catholique romaine2. Il étudie à la St. Michael's Preparatory School (en) de Saint-Sauveur avant d\'intégrer l\'internat de la Stowe School (en) de Buckingham, où il fait ses premières gammes en tant qu\'acteur et souffre des moqueries de ses camarades à cause de son embonpoint ; ces derniers le surnommant "Fat Cavill"3.\n\n', 'sentences': [{'id': 1, 'text': 'Henry William Dalgliesh Cavill, né le 5 mai 1983 à Saint-Hélier, est un acteur britannique.', 'dspan': (0, 91), 'tokens': [{'id': 1, 'text': 'Henry', 'dspan': (0, 5), 'span': (0, 5), 'lemma': 'Henry'}, {'id': 2, 'text': 'William', 'dspan': (6, 13), 'span': (6, 13), 'lemma': 'William'}, {'id': 3, 'text': 'Dalgliesh', 'dspan': (14, 23), 'span': (14, 23), 'lemma': 'Dalgliesh'}, {'id': 4, 'text': 'Cavill', 'dspan': (24, 30), 'span': (24, 30), 'lemma': 'Cavill'}, {'id': 5, 'text': ',', 'dspan': (30, 31), 'span': (30, 31), 'lemma': ','}, {'id': 6, 'text': 'né', 'dspan': (32, 34), 'span': (32, 34), 'lemma': 'naître'}, {'id': 7, 'text': 'le', 'dspan': (35, 37), 'span': (35, 37), 'lemma': 'le'}, {'id': 8, 'text': '5', 'dspan': (38, 39), 'span': (38, 39), 'lemma': '5'}, {'id': 9, 'text': 'mai', 'dspan': (40, 43), 'span': (40, 43), 'lemma': 'mai'}, {'id': 10, 'text': '1983', 'dspan': (44, 48), 'span': (44, 48), 'lemma': '1983'}, {'id': 11, 'text': 'à', 'dspan': (49, 50), 'span': (49, 50), 'lemma': 'à'}, {'id': 12, 'text': 'Saint-Hélier', 'dspan': (51, 63), 'span': (51, 63), 'lemma': 'Saint-Hélier'}, {'id': 13, 'text': ',', 'dspan': (63, 64), 'span': (63, 64), 'lemma': ','}, {'id': 14, 'text': 'est', 'dspan': (65, 68), 'span': (65, 68), 'lemma': 'être'}, {'id': 15, 'text': 'un', 'dspan': (69, 71), 'span': (69, 71), 'lemma': 'un'}, {'id': 16, 'text': 'acteur', 'dspan': (72, 78), 'span': (72, 78), 'lemma': 'acteur'}, {'id': 17, 'text': 'britannique', 'dspan': (79, 90), 'span': (79, 90), 'lemma': 'britannique'}, {'id': 18, 'text': '.', 'dspan': (90, 91), 'span': (90, 91), 'lemma': '.'}]
```


On voit bien qu'il y a les lemmes mais pas les pos ! Ici, pour récupérer le lemme dans ma variable lem, il faut que je parcours 'sentences' puis 'token' et 'lemma'.

En fait, si je fais ma lemmatization, il n'y a pas les pos et pour obtenir les pos il n'y a pas les lemmes, donc il faut que je fasse deux boucles différentes. J'ai décidé de créer 3 listes avec les pos, les lemmes et les tokens et de les fusionner après dans une autre boucle mais ça n'alignait pas les bons éléments…

```
sentence = lem['sentences'][0]

list_lemmas = []
list_upos = []
list_text = []

for phrase in pos['sentences']:
      for token in sentence['tokens']:
      lemma = token.get('lemma', " ")
      list_lemmas.append(lemma)
      for token in phrase['tokens']:
      # on utilise get pour que si une clé est vide ça ne pose pas de
problème
      upos = token.get('upos', " ")
      list_upos.append(upos)
      text = token.get('text', " ")
      list_text.append(text)
      for t, p, l in zip(list_text, list_upos, list_lemmas):
            print(f"{t}, {p}, {l}")
```

Conclusion

Trankit a l'air d'être très complet, et donc par extension, très compliqué. J'ai vu quelque part qu'il valait mieux utiliser spaCy pour faire une pré-tokenisation et en effet, sur la documentation, ils conseillent d'utiliser des données pré-tokenisées. Je pense cependant qu'il y a une solution plus simple et rapide que celle que je propose.

Les métriques

Comment on s'y prend ?

Je pensais au début qu'il vallait mieux donner le résultat en une liste de listes car ce format sera plus simple à traiter pour faire le rappel, la mesure de précision et la m-mesure avec le module sklearn. J'avais ensuite aplati les listes pour qu'elles deviennent une seule liste de valeurs unidimensionnelles avec la méthode chain() du module itertools. Seulement, en débuggant mon code j'ai trouvé une solution plus simple.

J'ai choisi de faire la f-mesure sur les données "Mot", "Lemme" et "POS" car regarder manuellement les DEP prendrait trop de temps et serait en plus assez difficile à faire. Pour afficher les métriques, il faut lancer le script sur le terminal.

Mon premier résultat

```
Précision : 0.5
Rappel : 0.5
F-mesure : 0.5
```

Je me suis rendue compte que si le premier élément de la liste était le même, évidemment que ça ne marcherait pas. Il faut alors que je compare uniquement les POS pour que ce soit plus simple. À la base j'avais 532 mots dans le corpus mais comme je devais annoter à la main le reste j'ai décidé de diviser par deux le corpus.

Deuxième tentative

```
Précision : 0.0
Rappel : 0.0
F-mesure : 0.0
```

J'ai réussi à avoir d'autres résultats en faisant simplement une liste de strings avec [ligne_token, ligne_lemme, ligne_pos, ligne_dep].

n tentative

En suite, je me suis rendue compte qu'il fallait que je mette average="binary" pour que ce soit True ou False, et là j'ai obtenu un tout autre résultat :

```
Précision : 1.0
Rappel : 1.0
F-mesure : 1.0
```

Finalement, une réussite

```
Texte non normé :
Précision : 0.9559748427672956
Rappel : 1.0
F-mesure : 0.977491961414791

Texte normé :
Précision : 0.9656699889258029
Rappel : 1.0
F-mesure : 0.9825352112676057
```

Comme j'ai réussi à avoir tout les types de résultats possibles et inimaginables, je me disais que j'allais bien y arriver un jour. J'ai alors compris que je n'avais pas une liste de tuples, mais bien une seule liste de tuples, répétée plusieurs fois, soit des centaines de listes uniques avec un seul tuples à cause de ça : [ligne_token, ligne_lemme, ligne_pos, ligne_dep]

J'ai donc ensuite créé une boucle pour analyser les résultats. J'ai considéré que mon annotation manuelle était 100% juste et j'ai comparé chaque éléments de chaque tuples : si l'élément n'est pas le même, on met un 0 parce qu'on considère que la machine s'est trompée et si c'est le même on met un 1.

Analyse des résultats

```
Texte non normé :
Précision : 0.9559748427672956
Rappel : 1.0
F-mesure : 0.977491961414791

Texte normé :
Précision : 0.9656699889258029
Rappel : 1.0
F-mesure : 0.9825352112676057
```

Comme on pouvait s'y attendre, la f-mesure pour le texte normé est plus haute que par rapport au texte non normé. Cependant, je m'attendais à ce que la différence soit plus flagrante. En effet, comme le rappel est toujour égal à zéro, elle sera toujours plus ou moins élevée.

À mon avis, la raison pour laquelle la précision est très élevée c'est parce qu'il y a beaucoup de ponctuation et qu'elle sera toujours à 100% bien annotée. Si on veut tester la précision pour les mots il faudrait enlever la ponctuation.

Conclusion

J'ai passé beaucoup de temps à régler des problèmes de module et l'affichage en html qui était automatique. Malgré tout cela, j'ai beaucoup apprécié faire ce devoir et j'ai appris beaucoup de choses en python et sur l'analyse automatique. Ce sont ces genres de projets qui poussent à aller chercher de plus en loin pour avoir un résultats optimal.

Malheureusement, j'ai été prise de court par le temps et il a pas mal de choses que j'aurais aimé approfondir et que j'ai laissé de côté pour primer la qualité à la quantité, notamment pour l'affichage en html et le nombre de modules python utilisé. Même si je n'ai pas utilisé les modules NLTK, Trankit et TreeTagger dans ce devoir, je me suis beaucoup renseigné sur eux et je compte les approfondir plus tard.

Je suis donc au final assez satisfaite de ce que j'ai pu faire car j'ai appris énormement de choses et parce que le projet était très intéressant et ça me motive pour en faire d'autres de mon côté dans le futur.