Traitement du Langage Naturel et Linguistique

LE BELLEGO Victor, MARTZLOFF Alice, MOUSSOX Vincent

M2 IAAA 2021/2022

1 Certaines langues sont elles plus difficiles à analyser que d'autres ?

L'objectif de ce projet est d'analyser et de comprendre les raisons pour lesquelles des analyseurs syntaxiques partageant la même architecture et entraînés sur la même quantité de données obtiennent des performances très différentes sur différentes langues. On peut observer ce phénomène dans la Table 2 qui présente les performances calculées à l'aide des mesures LAS (Labeled Accuracy Score) et UAS (Unlabeled Accuracy Score) obtenues par un analyseur sur 36 langues différentes.

lang	las	uas	lang	las	uas	lang	las	uas
da	66.18	73.06	zh	46.76	55.60	pl	68.76	80.69
hr	58.88	69.67	lv	51.46	59.32	sv	64.52	71.11
id	69.99	75.63	he	64.82	70.06	cs	69.77	77.40
ar	60.88	69.78	ko	46.06	55.00	nl	54.92	63.69
eu	47.94	58.17	ja	71.54	82.71	hu	57.21	69.24
it	74.82	81.01	ca	64.79	71.13	bg	68.08	78.14
fa	47.70	56.17	en	66.57	71.22	vi	48.40	49.62
es	66.09	71.01	pt	70.84	74.55	sl	49.46	61.51
ro	57.23	68.61	et	59.04	73.09	nno	73.58	78.30
de	58.12	64.19	fr	68.40	73.46	nob	66.22	73.93
hi	64.03	72.54	el	69.96	76.58			

Table 1 – LAS/UAS calculées pour les différentes langues (sans les *configurational features*)

1.1 Variables explicatives

Nous faisons l'hypothèse qu'il est possible de prédire les capacités d'un analyseur entraîné sur une langue à partir de variables directement observables. La pluspart sont observées sur le corpus d'apprentissage, mais nous utilisons aussi un score de complexité à partir du WALS.

1.1.1 Observations effectuées sur le corpus d'apprentissage

Ces variables observables sur le corpus sont les suivantes :

- Taille du vocabulaire
- Longueur moyenne d'une phrase
- Longueur moyenne d'un mot
- Taux de projectivité
- Longueur moyenne des dépendances
- Nombre moyen de dépendances
- Longueur moyenne de la plus longue dépendance

Taux de projectivité

La projectivité est une proprieté de certains arbres de dépendance. Elle se traduit par le fait que lorsqu'on trace les dependances au dessus des mots de la phrase, il n'y a jamais de croisement.

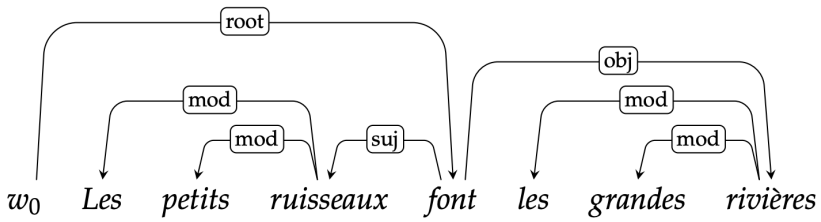

Figure 1 – Exemple d'une phrase française projective

La grande majorité des phrases françaises sont projectives, mais il existe certains cas de non projectivité.

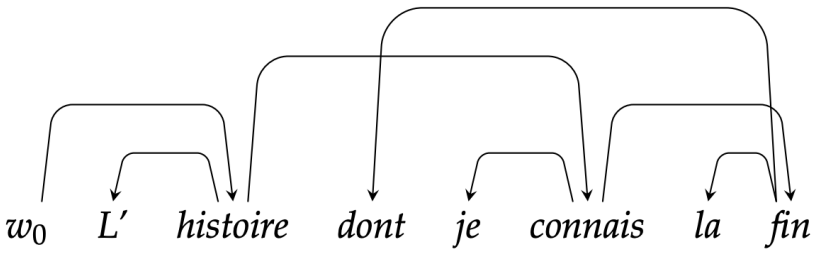

Figure 2 – Exemple d'une phrase française non projective

Le degré de non projectivité varie d'une langue à l'autre. Il est elevé pour le tchèque, le néerlandais ou le turc, par exemple.

Nous supposons que le taux de non projectivité peut expliquer les performances d'un analyseur.

Le calcul du taux de projectivité est très simple. Il suffit de compter, pour une langue, le nombre de phrases contenues dans le fichier `lang.conllu` et le nombre de phrases contenues dans le fichier `lang_proj.conllu` (qui ne contient que les phrases projectives) et d'en faire le quotient (voir la fonction `taux_projectvite()` du fichier `featuresAnalyser.py` en annexe)

Longueur des dépendances

Nous supposons que la longueur des dépendances des phrases projectives ou non projectives peut aussi expliquer les performances de l'analyseur.

Pour calculer la longueur des dépendances, nous utilions les fichiers `.mcf` et la fonction `long_dependances()` (voir `featuresAnalyser.py` en annexe). L'agorithme opère de la manière suivante :

1. découpage des phrases grâce à l'attribut EOS (*End Of Sentence*)
2. création liste de listes contenant l'attribut GOV (positions relatives des gouverneurs pour chaque mot)
3. conversion de ces postions relatives en positions absolues grâce à la fonction `rel2abs()`
4. création de l'arbre correspondant pour chaque liste grâce à la fonction `maketree()`
5. calcul de la profondeur de l'arbre

Nous avons extrait 3 variables explicatives pour chaque langue :

— Longueur moyenne des dépendances

— Nombre moyen de dépendances par phrase

— Longueur moyenne de la plus longue dépendance de chaque phrase

lang	Taille du vocabulaire	Taille moyenne des phrases	Longueur moyenne des mots	Taux de non projectivité	Nombre moyen de dépendances	Longueur moyenne des dépendances	Longueur de la plus longue dépendance
da	3545	15.71	7.62	0.2	10.77	3.15	4.56
hr	3974	19.2	7.65	0.08	13.06	3.64	5.59
id	3580	18.72	7.4	0.04	12.64	3.66	5.59
ar	5304	36.68	6.43	0.09	17.07	5.04	7.83
eu	2772	11.39	7.31	0.34	7.26	2.76	4.01
it	3354	20.23	8.06	0.05	13.34	3.49	5.11
nno	2161	14.9	7.42	0.08	10.87	3.11	4.59
fa	3904	24.02	5.42	0.07	14.64	3.78	6.11
es	3424	25.63	7.58	0.08	17.69	3.93	5.94
ro	3767	20.01	7.01	0.12	13.52	3.75	5.75
de	4121	16.38	9.1	0.11	11.5	3.18	4.52
hi	2453	19.43	5.33	0.14	12.14	3.44	5.0
zh	4289	21.62	1.99	0.0	15.34	3.45	5.25
lv	3359	11.68	7.34	0.08	8.39	2.98	4.15
he	6050	29.75	5.16	0.01	15.49	3.99	6.18
ko	10283	10.42	3.63	0.12	5.72	3.39	4.55
ja	4034	20.18	2.51	0.0	14.74	3.54	5.36
ca	2904	27.82	7.41	0.09	18.93	3.99	6.13
en	3375	14.61	6.57	0.08	10.32	3.07	4.4
pt	3980	23.93	7.5	0.13	15.18	3.75	5.55
et	3683	8.71	7.31	0.09	6.55	2.56	3.4
fr	3351	23.27	7.53	0.12	15.5	3.61	5.43
el	2734	21.28	7.86	0.11	13.76	3.82	5.63
pl	4762	8.51	7.79	0.01	6.09	2.85	3.9
sv	4451	14.83	7.66	0.11	9.79	3.11	4.51
cs	4875	15.52	7.35	0.13	10.22	3.45	5.02
nl	3552	12.52	7.62	0.27	7.73	2.82	3.75
hu	3677	19.69	8.05	0.23	12.5	3.37	5.1
nob	2155	13.1	7.41	0.08	9.42	3.05	4.42
bg	3736	11.68	7.8	0.03	8.4	3.19	4.42
vi	1798	12.36	3.42	0.02	8.81	3.1	4.68
sl	4219	13.38	7.3	0.14	9.34	2.97	4.2
moy	3863	18.04	6.73	0.10	11.77	3.41	5.02

Table 2 – Synthèse des variables explicatives pour les corpus *train* pour chaque langue

1.1.2 Complexité selon M. Parkvall

Dans son papier *The simplicity of creoles in a cross-linguistic perspective* (Parkvall 2008) sorti en 2008, Mikael Parkvall s'intéresse à quantifier la complexité des langues. Il part du postulat qu'une expression est d'autant plus complexe qu'elle implique de règles, c'est-à-dire qu'elle requiert une longue description. Ainsi, l'hypothèse de base de l'auteur est la suivante : une langue complexe est une langue avec des constructions plus complexes. Il

explore un aspect de complexité structurelle.

Prenons par exemple la voix passive. Lorsqu'elle existe dans une langue, il faut pouvoir définir comment passer de la voie active à la voix passive, ce qui exige une explication de règle supplémentaire. Une langue qui possède une voix passive est donc, en ce qui concerne cette construction spécifique, plus complexe qu'une autre n'en possédant pas. Si on énumère donc un grand nombre de « constructions complexes », la langue la plus complexe sera celle qui en compte le plus grand nombre.

Pour extraire les « constructions complexes » qu'on peut trouver dans une langue, Parkvall utilise le set de données *World Atlas of Linguistic Structures* (WALS) publié en 2005 par Haspelmath et al. Il choisit 155 langues parmi plus de 2 500, et 47 caractéristiques parmi plus de 140.

Choix des caractéristiques

Parkvall exclut des caractéristiques selon un raisonnement défendu dans son papier et qui s'efforce de mettre la majorité des linguistes d'accord sur le fait qu'une caractéristique apporte ou non de la complexité. Il retient les caractéristiques suivantes :

Caractéristiques du WALS		
Size of consonant inventories	Distance contrast in demonstratives	Morphological imperative
Size of vowel quality inventories	Gender in pronouns	Morphological optative
Phonemic vowel nasalization	Politeness in pronouns	Grammaticalized evidentiality distinctions
Complexity of syllable structure	Person marking on adpositions	Both indirect and direct evidentials
Tone	Comitative ≠ instrumental	Non-neutral marking of full NPs
Overt marking of direct object	Ordinals exist as a separate class beyond 'first'	Non-neutral marking of pronouns
Double marking of direct object	Suppletive ordinals beyond 'first'	Subject marking as both free word and agreement
Possession by double marking	Obligatory numeral classifiers	Passive
Overt possession marking	Possessive classification	Antipassive
Reduplication	Conjunction 'and' ≠ adposition 'with'	Applicative
Gender	Difference between nominal and verbal conjunction	Obligatorily double negation
Number of genders	Grammaticalized perfective/imperfective	Asymetric negation
Non-semantic gender assignment	Grammaticalized past/non-past	Equative copula ≠ Locative copula
Grammaticalized nominal plural	Remoteness distinctions of past	Obligatorily overt equative copula
Definite articles Indefinite articles	Morphological future	
Inclusivity (in either pronouns or verb morphology)	Grammaticalized perfect	

Table 3 – Liste des caractéristiques extraite directement du WALS

Il ajoute à ces caractéristiques, d'autres données « résiduelles » d'auteurs contributeurs au WALS. Ce sont les suivantes :

Caractéristiques d'auteurs du WALS		
Demonstratives marked for number	Demonstratives marked for gender	Demonstratives marked for case
Total amount of verbal suppletion	Alienability distinctions	

Table 4 – Liste de caractéristiques proposées par les auteurs du WALS

Enfin, il s'intéresse aussi à une donnée de Harley and Ritter (2002) à laquelle il a eu accès :

Caractéristique de Harley et Ritter
Number of pronominal numbers

Table 5 – Une caractéristique accessible, inspirée de Harley et al. (2002)

Les valeurs de ces caractéristiques ont toutes été traduite par l'auteur comme des valeurs comprises entre 0 et 1 :

— « Oui » ou « non » deviennent 0 ou 1 avec parfois l'introduction de valeurs intermédiaire 0,5 ;

— Des valeurs d'intensité comprises entre 1 et 4 sont compréssées en : 0 - 0,25 - 0,5 - 0,75 et 1 ;

— Des valeurs catégoriques comme la classification en « simple », « modérément complexe » et « complexe » sont traduites en 0 , 0,5 et 1.

Choix des langues

L'auteur s'attèle à choisir des langues dont les annotations sont le moins lacunaires possible pour les caractéristiques décrites ci-dessus. Pour une langue i donnée :

$$\text{Score}_{\,i} = \frac{\sum_{k=1}^{L} \text{contribution}_{\,k}}{L} \tag{1}$$

où k représente une caractéristique. Chaque langue comptant un nombre différent L de caractéristiques (parmi celles choisies par l'auteur) effectivement annontées pour cette langue.

N.B. : en effet, de même que pour les caractéristiques que nous extrayons directement de nos données, Parkvall note que le set de données WALS n'est pas identiquement distribué : les langues ne sont pas identiquement annotées pour les caractéristiques proposées...

1.2 Cohérence des annotations

Dans leur article *Divergences entre annotations dans le projet Universal Dependencies (*Nivre *et al. 2017) et leur impact sur l'évaluation de l'étiquetage morpho-syntaxique* (Wisniewski et Yvon 2018), Guillaume Wisniewski et François Yvon montrent que la dégradation des performances observée lors de l'application d'un analyseur morpho-syntaxique à des données hors domaine comme ici, d'une langue à l'autre, résulte d'incohérences entre les annotations des ensembles de test et d'apprentissage. Ils montrent qu'appliquer le principe de variation des annotations de Dickinson & Meurers (Meurers 2003) permet d'identifier les erreurs d'annotation et donc les incohérences et évaluer leur impact. Nous souhaitions nous inspirer de ces méthodes mais n'avons pu en raison notamment du temps qui nous manque proposer une telle amélioration...

1.3 Mise en oeuvre

Nous réalisons une régression multiple à l'aide du modèle `LinearRegression` de `scikit-learn` pour prédire LAS après **normalisation** des variables explicatives.

Il est d'abord intéressant de regarder les potentielles corrélations entre nos variables explicatives.

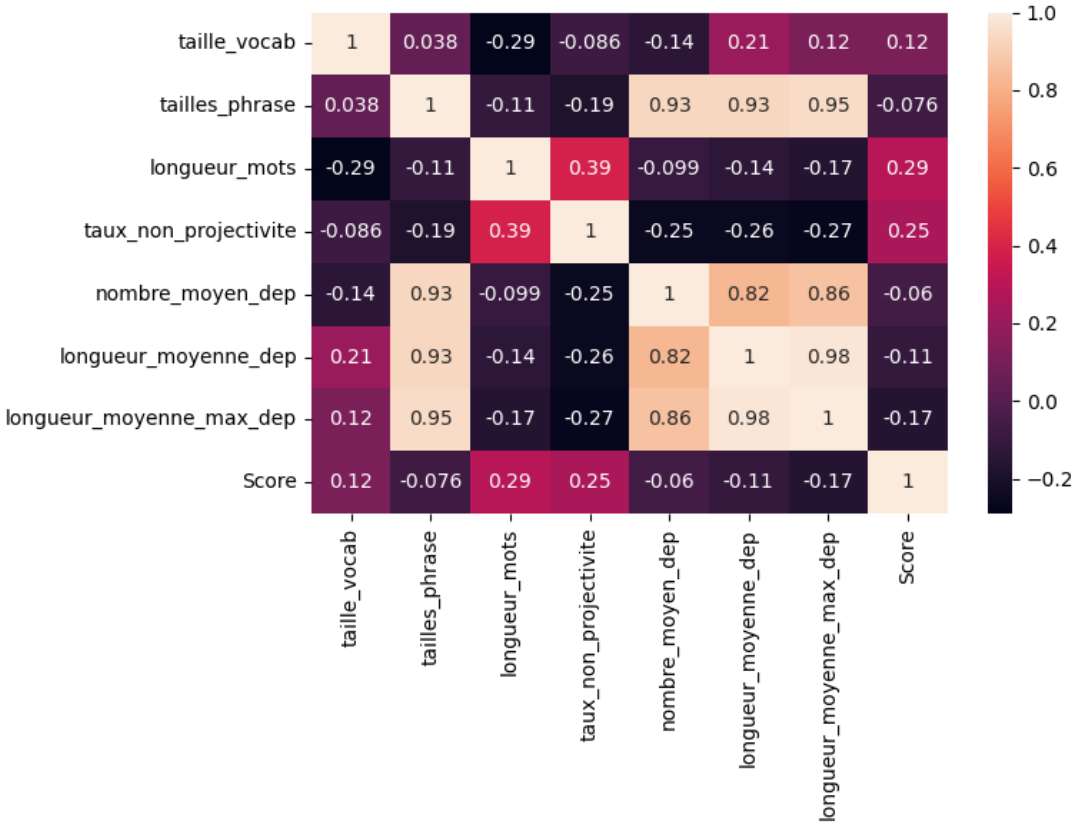

Figure 3 – Corrélations entre les variables explicatives

On remarque, en toute logique, que la taille des phrases est corélée au nombre et à la taille des dépendances. On constate aussi que la longueur moyenne des dépendances est corrélée au nombre moyen des dépendances : plus il y a de dépendances, plus elles sont longues.

Nos variables normalisées sont notées comme suit :

- x_1 : Taille du vocabulaire
- x_2 : Longueur moyenne d'une phrase
- x_3 : Longueur moyenne d'un mot
- x_4 : Taux de projectivité
- x_5 : Longueur moyenne des dépendances
- x_6 : Nombre moyen de dépendances
- x_7 : Longueur moyenne de la plus longue dépendance
- x_8 : Score de M. Parkvall (WALS)

Les LAS sont notés y. On souhaite estimer les y tels que :

$$\exists [\alpha_1, \ldots, \alpha_8, \beta] \in \mathbb{R}, \forall i \quad y_i = \beta + \sum_{j=1}^{8} \alpha_j \cdot x_{i,j}$$

On utilise l'estimateur des moindres carrés :

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} \left(y_i - (\alpha x_i + \beta)\right)^2$$

On obtient les résultats suivants :

$\beta = 61.66$, $\alpha_1 = -2.32$, $\alpha_2 = -0.75$, $\alpha_3 = 3.63$, $\alpha_4 = -3.56$, $\alpha_5 = 3.76$, $\alpha_6 = 17,40$, $\alpha_7 = -18.33$, $\alpha_8 = 0.02$

avec $R^2 = 0.476$, qui est moyen. L'erreur RMS s'élève à 6.198.

Du fait de la normalisation des variables d'entrée, il est directement possible de déduire l'impact de chaque variable. On remarque que le score de M. Parkvall (WALS) a un impact quasi nul sur la régression. En revanche, la longueur moyenne des dépendance et la longueur moyenne de la plus longue dépendance ont une importance majeur, bien qu'opposé.

1.4 Conclusion

1.4.1 Est-il possible de connaître a priori les performances de l'analyseur en fonction de certaines caractéristiques de la langue ?

Bien que les résultats obtenus ne soient pas à la hauteur des espérance, il semblerait certaines caractéristiques de la langue permettent de connaître a priori les performances de l'analyseur.

Notre étude porte sur un **corpus de faible taille qui comporte, par essence, de nombreux biais** ; il est donc difficile de généraliser. Une étude avec des corpus de plus grande taille serait nécessaire pour affiner la régression et ainsi conclure de manière plus certaine.

1.4.2 Est-il possible d'utiliser les conclusions de l'étude statistique pour améliorer les performances de l'analyseur ?

Il est a priori possible d'utiliser l'étude statistique pour améliorer les performances de l'analyseur. En effet, connaissant les caractéristiques de la langues, on peut choisir un analyseur moins sensible à la non projectivité ou aux longues dépendances, mais plus couteux seulement pour les langues qui le requierent. Les langues projectives (comme le japonais) peuvent être traités avec des analyseurs plus simples et moins coûteux.

Ainsi, l'étude statistique permet d'améliorer les performances de l'analyseur en orientant l'investissement en ressources pour les langues les plus « complexes »

Références

Meurers, Detmar (jan. 2003). “Detecting Errors in Part-of-Speech Annotation”. In : p. 107-114.

Nivre, Joakim et al. (2017). *Universal Dependencies 2.1*. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. url : http://hdl.handle.net/11234/1-2515.

Parkvall, Mikael (jan. 2008). “The simplicity of creoles in a cross-linguistic perspective”. In : p. 265-285. doi : 10.1075/slcs.94.17par.

Wisniewski, Guillaume et François Yvon (mai 2018). “Divergences entre annotations dans le projet Universal Dependencies et leur impact sur l’évaluation de l’étiquetage morpho-syntaxique”. In : *Conférence sur le Traitement Automatique des Langues Naturelles*. Rennes, France. url : https://hal.archives-ouvertes.fr/hal-01793784.

Annexes

../resultsTable.py

```python
import os
from os import listdir
from subprocess import run
import pandas as pd
from tabulate import tabulate

base_dir = os.getcwd()
os.chdir(base_dir)

list_lang = [k.replace('.conllu','').replace('train_','') for k in listdir('data') if '.conllu' in k and 'train' in k]
print('Training '+str(len(list_lang))+' languages')

os.chdir('expe/')
i=1
nbre = len(list_lang)
for lang in list_lang:
    try :
        print('Testing '+lang+' '+str(i)+'/'+str(nbre))
        command = 'make lang=' + lang
        run(command.split(), capture_output=True)
        print('Success !')
        i += 1
    except :
        print("Erreur " + lang)
        pass

print('Reading results...')
os.chdir('out/')
df = pd.DataFrame(columns=["lang", "las", "uas"])
for lang in list_lang:
    try:
        with open(lang+'.res','r') as f:
            lines = f.readlines()
            to_append = lines[-1].split()
            df_length = len(df)
            df.loc[df_length] = to_append
    except :
        print("Erreur " + lang)
        pass

df.set_index('lang', inplace=True)
df.to_csv('out.csv')
print(tabulate(df, headers='keys', tablefmt='psql'))
```

../featuresAnalyser.py

```python
import os
from os import listdir
import pandas as pd
import numpy as np


base_dir = os.getcwd()

def extract_features(lang, filetype, proj=False):
    if proj == True:
        lang += "_proj"
    os.chdir(base_dir + '/data/')
    taille_phrases = []
    liste_mots = []
    with open(filetype + "_" + lang + ".conllu", 'r', encoding='utf8') as f:
        lines = f.readlines()
        empty_lines = [-1]
        for i,line in enumerate(lines):
            if line == "\n":
                empty_lines.append(i)
        for indice in range(len(empty_lines)-1):
            phrase = lines[empty_lines[indice]+1:empty_lines[indice+1]-1]
            taille_phrase = 0
            for word in phrase:
                if "PUNCT" not in word: #On enlève la ponctuation pour compter
                    taille_phrase += 1
                    if "PROPN" not in word and "NUM" not in word: #On enlève
                    ↪ les nombres et les noms propres pour la diversité du
                    ↪ vocab
                        word = word.split()
                        if word[2] != "_":
                            liste_mots.append(word[2])
                        else :
                            liste_mots.append(word[1])
            taille_phrases.append(taille_phrase)
        liste_mots = list(set(liste_mots))
        return {"liste_mots" : liste_mots, "taille_phrases":taille_phrases}

def var_exp_df(filetype):
    os.chdir(base_dir)
    list_lang = [k.replace('.conllu', '').replace('train_', '') for k in
    ↪ listdir('data') if
                 '.conllu' in k and 'train' in k]
    tailles_vocab, tailles_phrases, longueur_mots, tx_non_projectivite,
    ↪ nombre_moyen_dep, longueur_moyenne_dep, longueur_moyenne_max_dep  =
    ↪ [],[],[],[],[],[],[]
    for lang in list_lang:
        print('Analysing lang : ',lang)
        my_dict = extract_features(lang, filetype)
        tailles_vocab.append(len(my_dict["liste_mots"]))
        tailles_phrases.append(np.mean(my_dict["taille_phrases"]))
```

```
47          longueur_mots.append(np.mean([len(x) for x in my_dict["liste_mots"]]))
48          tx_non_projectivite.append(1-taux_projectvite(filetype,lang))
49          my_dict_dep = long_dependances(filetype,lang)
50          nombre_moyen_dep.append(my_dict_dep['nombre_moyen_dep'])
51          longueur_moyenne_dep.append(my_dict_dep['longueur_moyenne_dep'])
52
            ↪  longueur_moyenne_max_dep.append(my_dict_dep['longueur_moyenne_max_dep'])
53      df = pd.DataFrame({'lang':list_lang,
54                         'taille_vocab' : tailles_vocab,
55                         'tailles_phrase' : np.round(tailles_phrases,2),
56                         'longueur_mots' : np.round(longueur_mots,2),
57                         'taux_non_projectivite':
                           ↪  np.round(tx_non_projectivite,2),
58                         'nombre_moyen_dep':  np.round(nombre_moyen_dep,2),
59                         'longueur_moyenne_dep':
                           ↪  np.round(longueur_moyenne_dep,2),
60                         'longueur_moyenne_max_dep':
                           ↪  np.round(longueur_moyenne_max_dep,2),
61                         })
62      return df
63
64  def make_df(filetype):
65      df = var_exp_df(filetype)
66      df.set_index('lang', inplace=True)
67      df2 = pd.read_csv(base_dir + '/out.csv', index_col='lang')
68      df = df.join(df2)
69      df2 = pd.read_csv(base_dir + '/complexityScore.csv', index_col='Code')
70      df2.drop(['Language', 'Extrapolation'], axis = 1, inplace=True)
71      df = df.join(df2)
72      os.chdir(base_dir)
73      df.to_csv('results_'+filetype+'.csv')
74      return df
75
76  def taux_projectvite(filetype,lang):
77      os.chdir(base_dir + '/data/')
78      with open(filetype + "_" + lang + ".conllu", 'r', encoding='utf8') as f:
79          lines = f.readlines()
80          empty_lines = [-1]
81          for i, line in enumerate(lines):
82              if line == "\n":
83                  empty_lines.append(i)
84          nbre_phrases = len(empty_lines)
85      os.chdir(base_dir + '/expe/out/')
86      with open(filetype + "_" + lang + "_proj.conllu", 'r', encoding='utf8') as
        ↪  f:
87          lines = f.readlines()
88          empty_lines = [-1]
89          for i, line in enumerate(lines):
90              if line == "\n":
91                  empty_lines.append(i)
92          nbre_phrases_proj = len(empty_lines)
93      return nbre_phrases_proj/nbre_phrases
```

```python

def long_dependances(filetype,lang):
    os.chdir(base_dir + '/expe/out/')
    long, nbre, longmax = [],[],[]
    with open(filetype + "_" + lang + "_pgle.mcf", 'r', encoding='utf8') as f:
        ## construction des indices de phrases ##
        lines = f.readlines()
        sentences = [0]
        for i,line in enumerate(lines):
            line = line.split("\t")
            if int(line[4]) == 1:
                sentences.append(i+1)
        liste = []
        for i in range(len(sentences) - 1):
            liste2 = []
            for line in lines[sentences[i]:sentences[i + 1]]:
                word = line.split("\t")
                if word[3] == 'root':
                    liste2.append(0)
                else :
                    liste2.append(int(word[2]))
            liste.append(liste2)
    liste_abs = rel2abs(liste)
    for liste3 in liste_abs:
        arbre = maketree(liste3)
        nbre.append(len(arbre))
        longueurs = [len(dep) for dep in arbre]
        long.append(np.mean(longueurs))
        longmax.append(np.max(longueurs))
    return {"nombre_moyen_dep" : np.mean(nbre),
    ↪  "longueur_moyenne_dep":np.mean(long), "longueur_moyenne_max_dep" :
    ↪  np.mean(longmax)}

def rel2abs(liste):
    rep = []
    for liste1 in liste:
        rep.append([i+j if j != 0 else -1 for i,j in enumerate(liste1)])
    return rep

def maketree(liste):
    arbre = [[liste.index(-1)]]
    profondeur = 1
    while True:
        nouvel_arbre = []
        liste_recherche = [j[-1] for j in arbre if len(j)==profondeur]
        if liste_recherche == []:
            return arbre
        for l in arbre:
            if l[-1] in liste_recherche and l[-1] in liste:
                indices = [i for i, x in enumerate(liste) if x == l[-1]]
                for k in indices:
                    nouvel_arbre.append(l+[k])
```

```
            else:
                nouvel_arbre.append(l)
        arbre = nouvel_arbre
        profondeur += 1
```

../RegressionMultiple.py

```
import numpy as np
import pandas as pd
from featuresAnalyser import make_df
import matplotlib.pyplot as plt
import seaborn as sns

### Plot ###
def my_plot(df,x_value,y):
    y = y["las"].values.tolist()
    x = df[x_value].values.tolist()

    plt.figure(figsize=(7, 4))
    sns.set_style("whitegrid")
    ax = sns.lineplot(x=x, y=y, color='red')
    sns.despine(left=True)
    ax.axhline(0, color='grey')
    # ax.set(xlabel='Année', ylabel="Taux (%)")
    # ax.legend(['Arabie Saoudite','Moyen-Orient et Afrique du Nord'])
    plt.show()

#### Data ####

#df_train = make_df("train")
#df_train.sort_values(by=['uas'], inplace=True)
#df_test = make_df("dev")
#df_test.sort_values(by=['uas'], inplace=True)

def mean_norm(df_input):
    return df_input.apply(lambda x: (x-x.mean())/ x.std(), axis=0)

df_train =
↪  pd.read_csv('results_train.csv',index_col='lang').sort_values(by=['uas'])
df_test =
↪  pd.read_csv('results_dev.csv',index_col='lang').sort_values(by=['uas'])

X_train = df_train[["taille_vocab" , "tailles_phrase" , "longueur_mots" ,
↪  "taux_non_projectivite" , "nombre_moyen_dep" , "longueur_moyenne_dep" ,
↪  "longueur_moyenne_max_dep", "Score"]]
X_train.fillna(X_train.mean(), inplace=True)
X_train = mean_norm(X_train)
y_train = df_train[["las"]]
X_test = df_test[["taille_vocab" , "tailles_phrase" , "longueur_mots" ,
↪  "taux_non_projectivite" , "nombre_moyen_dep" , "longueur_moyenne_dep" ,
↪  "longueur_moyenne_max_dep", "Score"]]
X_test.fillna(X_test.mean(), inplace=True)
X_test = mean_norm(X_test)
```

```
y_test = df_test[["las"]]

### Corrélations ###

corr_df = X_train.corr(method='pearson')

plt.figure(figsize=(8, 6))
sns.heatmap(corr_df, annot=True)
plt.tight_layout()
plt.show()

### Regression ###

# importing module
from sklearn.linear_model import LinearRegression
# creating an object of LinearRegression class
LR = LinearRegression()
# fitting the training data
LR.fit(X_train,y_train)

print(LR.intercept_)
print(LR.coef_)

y_prediction =  LR.predict(X_test)
y_prediction

# importing r2_score module
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
# predicting the accuracy score
score=r2_score(y_test,y_prediction)

print('r2 score is ', score)
print('MSE is ',mean_squared_error(y_test,y_prediction))
print('RMS error of is ',np.sqrt(mean_squared_error(y_test,y_prediction)))

for var in ["taille_vocab" , "tailles_phrase" , "longueur_mots" ,
↪  "taux_non_projectivite" , "nombre_moyen_dep" , "longueur_moyenne_dep" ,
↪  "longueur_moyenne_max_dep", "Score"]:
    my_plot(X_train,x_value=var,y=y_train)
```