Traitement du Langage Naturel et Linguistique

LE BELLEGO Victor, MARTZLOFF Alice, MOUSSOX Vincent

M2 IAAA 2021/2022

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*)

0.1 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.

0.1.1 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

En ce qui concerne des variables explicatives liées aux dépendances, comme la longueur moyenne des dépendances ou encore le nombre de dépendance, la longueur moyenne de la plus grande dépendance (ci-dessous), ce sont les variables les plus sujettes à relever de la cohérence de l'annotation pour chaque langue.

Nous évoquerons dans une autre section l'influence de la cohérence des annotations.

0.1.2 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 2 – 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 3 – 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 4 – 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...

0.2 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...

0.3 Conclusion

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

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

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

```
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

```
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"]))
```

```
        longueur_mots.append(np.mean([len(x) for x in my_dict["liste_mots"]]))
        tx_non_projectivite.append(1-taux_projectvite(filetype,lang))
        my_dict_dep = long_dependances(filetype,lang)
        nombre_moyen_dep.append(my_dict_dep['nombre_moyen_dep'])
        longueur_moyenne_dep.append(my_dict_dep['longueur_moyenne_dep'])

        ↪  longueur_moyenne_max_dep.append(my_dict_dep['longueur_moyenne_max_dep'])
    df = pd.DataFrame({'lang':list_lang,
                       'taille_vocab' : tailles_vocab,
                       'tailles_phrase' : np.round(tailles_phrases,2),
                       'longueur_mots' : np.round(longueur_mots,2),
                       'taux_non_projectivite':
                       ↪  np.round(tx_non_projectivite,2),
                       'nombre_moyen_dep':  np.round(nombre_moyen_dep,2),
                       'longueur_moyenne_dep':
                       ↪  np.round(longueur_moyenne_dep,2),
                       'longueur_moyenne_max_dep':
                       ↪  np.round(longueur_moyenne_max_dep,2),
                       })
    return df

def make_df(filetype):
    df = var_exp_df(filetype)
    df.set_index('lang', inplace=True)
    df2 = pd.read_csv(base_dir + '/out.csv', index_col='lang')
    df = df.join(df2)
    df2 = pd.read_csv(base_dir + '/complexityScore.csv', index_col='Code')
    df2.drop(['Language', 'Extrapolation'], axis = 1, inplace=True)
    df = df.join(df2)
    os.chdir(base_dir)
    df.to_csv('results_'+filetype+'.csv')
    return df

def taux_projectvite(filetype,lang):
    os.chdir(base_dir + '/data/')
    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)
        nbre_phrases = len(empty_lines)
    os.chdir(base_dir + '/expe/out/')
    with open(filetype + "_" + lang + "_proj.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)
        nbre_phrases_proj = len(empty_lines)
    return nbre_phrases_proj/nbre_phrases
```

```
94
95  def long_dependances(filetype,lang):
96      os.chdir(base_dir + '/expe/out/')
97      long, nbre, longmax = [],[],[]
98      with open(filetype + "_" + lang + "_pgle.mcf", 'r', encoding='utf8') as f:
99          ## construction des indices de phrases ##
100         lines = f.readlines()
101         sentences = [0]
102         for i,line in enumerate(lines):
103             line = line.split("\t")
104             if int(line[4]) == 1:
105                 sentences.append(i+1)
106         liste = []
107         for i in range(len(sentences) - 1):
108             liste2 = []
109             for line in lines[sentences[i]:sentences[i + 1]]:
110                 word = line.split("\t")
111                 if word[3] == 'root':
112                     liste2.append(0)
113                 else :
114                     liste2.append(int(word[2]))
115             liste.append(liste2)
116     liste_abs = rel2abs(liste)
117     for liste3 in liste_abs:
118         arbre = maketree(liste3)
119         nbre.append(len(arbre))
120         longueurs = [len(dep) for dep in arbre]
121         long.append(np.mean(longueurs))
122         longmax.append(np.max(longueurs))
123     return {"nombre_moyen_dep" : np.mean(nbre),
        ↪  "longueur_moyenne_dep":np.mean(long), "longueur_moyenne_max_dep" :
        ↪  np.mean(longmax)}
124
125 def rel2abs(liste):
126     rep = []
127     for liste1 in liste:
128         rep.append([i+j if j != 0 else -1 for i,j in enumerate(liste1)])
129     return rep
130
131 def maketree(liste):
132     arbre = [[liste.index(-1)]]
133     profondeur = 1
134     while True:
135         nouvel_arbre = []
136         liste_recherche = [j[-1] for j in arbre if len(j)==profondeur]
137         if liste_recherche == []:
138             return arbre
139         for l in arbre:
140             if l[-1] in liste_recherche and l[-1] in liste:
141                 indices = [i for i, x in enumerate(liste) if x == l[-1]]
142                 for k in indices:
143                     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"]]
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"]]
X_test = mean_norm(X_test)
y_test = df_test[["las"]]

```

```
41
42  ### Regression ###
43
44  # importing module
45  from sklearn.linear_model import LinearRegression
46  # creating an object of LinearRegression class
47  LR = LinearRegression()
48  # fitting the training data
49  LR.fit(X_train,y_train)
50
51  print(LR.intercept_)
52  print(LR.coef_)
53
54  y_prediction =  LR.predict(X_test)
55  y_prediction
56
57  # importing r2_score module
58  from sklearn.metrics import r2_score
59  from sklearn.metrics import mean_squared_error
60  # predicting the accuracy score
61  score=r2_score(y_test,y_prediction)
62
63  print('r2 score is ', score)
64  print('MSE is ',mean_squared_error(y_test,y_prediction))
65  print('RMS error of is ',np.sqrt(mean_squared_error(y_test,y_prediction)))
66
67  for var in ["taille_vocab" , "tailles_phrase" , "longueur_mots" ,
    ↪  "taux_non_projectivite" , "nombre_moyen_dep" , "longueur_moyenne_dep" ,
    ↪  "longueur_moyenne_max_dep"]:
68      my_plot(X_train,x_value=var,y=y_train)
```