

# NaijaSUD 实习总结



## 导师及主要实习人员：

- Sylvain Kahane : [sylvain@kahane.fr](mailto:sylvain@kahane.fr)
- Kim gerdes : [kim@gerdes.fr](mailto:kim@gerdes.fr)
- Marine Courtin : [rinema56@gmail.com](mailto:rinema56@gmail.com)
- Bernard Caron : [Bernard.I.caron@gmail.com](mailto:Bernard.I.caron@gmail.com), [Bernard.CARON@cnrs.fr](mailto:Bernard.CARON@cnrs.fr)
- Luigi Liu : [luigi.plurital@gmail.com](mailto:luigi.plurital@gmail.com)

## Github :

- [https://github.com/UniversalDependencies/UD\\_Naija-NSC](https://github.com/UniversalDependencies/UD_Naija-NSC)
- <https://gitlab.inria.fr/grew/SUD/blob/master/sud-treebanks-v2.2.md>

## • CONLL 格式（依存树库的 CoNLL 格式）：

个人理解：能够体现依存句法关系树的文本形式文件。

```
# elan_id = P_ABJ_GWA_06_024 P_ABJ_GWA_06_025
# sent_id = P_ABJ_GWA_06_Ugo-lifestory_PRO_11
# sent_translation = # That is when I called one of elder brothers who was in Lagos then I told that him that borther,
# text = # naim I call one of my elder broder wey dey Lagos dat time //
1      #          _      PUNCT      _      startali=35567|endali=35913      4      punct      _      _
2      naim      _      ADV      _      startali=35913|endali=36133      4      mod:periph      _      _
3      I          _      PRON      _      Case=Nom|PronType=Prs|Number=Sing|startali=36133|Person=1|endali=36203      4      subj      _      _
4      call      _      VERB      _      startali=36203|endali=36402      0      root      _      _
5      one      _      NUM      _      NumType=Card|startali=36402|endali=36608      4      comp:obj      _      _
6      of      _      ADP      _      startali=36608|endali=36661      5      dep      _      _
7      my      _      ADJ      _      PronType=Prs|Number=Sing|Poss=Yes|startali=36661|Person=1|endali=37118      9      det      _      _
8      elder      _      ADJ      _      Degree=Cmp|startali=37118|endali=37395      9      dep      _      _
9      broder      _      NOUN      _      startali=37395|endali=37673      6      comp      _      _
10     wey      _      SCONJ      _      startali=37673|endali=37833      9      dep      _      _
11     dey      _      VERB      _      PartType=Cop|startali=37833|endali=37953      10     comp      _      _
12     Lagos      _      PROPN      _      startali=37953|endali=38323      11     comp:pred      _      _
13     dat      _      DET      _      PronType=Dem|Number=Sing|startali=38323|endali=38443      14     det      _      _
14     time      _      NOUN      _      startali=38443|endali=38673      11     mod      _      _
15     //      _      PUNCT      _      startali=38673|endali=38703      4      punct      _      _
```

在 CONLL 格式中，每个词语占 1 行，1 行有 10 列，无值列用下划线 '\_' 代替，列的分隔符为制表符 '\t'，行的分隔符为换行符 '\n'；句子与句子之间用空行分隔。每列的定义：

1. ID：单词索引，每个新句子从 1 开始的整数；可能是多个词的标记的范围。
2. FORM：当前词语或标点。
3. LEMMA：当前词语（或标点）的原型或词干，在中文中，此列与 FORM 相同。
4. CPOSTAG：当前词语的词性（粗粒度）。
5. POSTAG：当前词语的词性（细粒度）。
6. FEATS：句法特征，在本次评测中。
7. HEAD：当前词语的中心词，数字代表该中心词的 ID。
8. DEPREL：当前词语与中心词的依存关系。
9. DEPS：二级依赖项列表（head-deprel 对）。
10. MISC：任何其他注释。

## CoNLL-U Format :

We use a revised version of [the CoNLL-X format](#) called CoNLL-U. Annotations are encoded in plain text files (UTF-8, using only the LF character as line break) with three types of lines:

1. Word lines containing the annotation of a word/token in 10 fields separated by single tab characters; see below.
2. Blank lines marking sentence boundaries.
3. Comment lines starting with hash (#).

Sentences consist of one or more word lines, and word lines contain the following fields:

1. ID: Word index, integer starting at 1 for each new sentence; may be a range for tokens with multiple words.
2. FORM: Word form or punctuation symbol.
3. LEMMA: Lemma or stem of word form.
4. UPOSTAG: [Universal part-of-speech tag](#) drawn from our revised version of the Google universal POS tags.
5. XPOSTAG: Language-specific part-of-speech tag; underscore if not available.
6. FEATS: List of morphological features from the [universal feature inventory](#) or from a defined [language-specific extension](#); underscore if not available.
7. HEAD: Head of the current token, which is either a value of ID or zero (0).
8. DEPREL: [Universal Stanford dependency relation](#) to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one.
9. DEPS: List of secondary dependencies (head-deprel pairs).
10. MISC: Any other annotation.

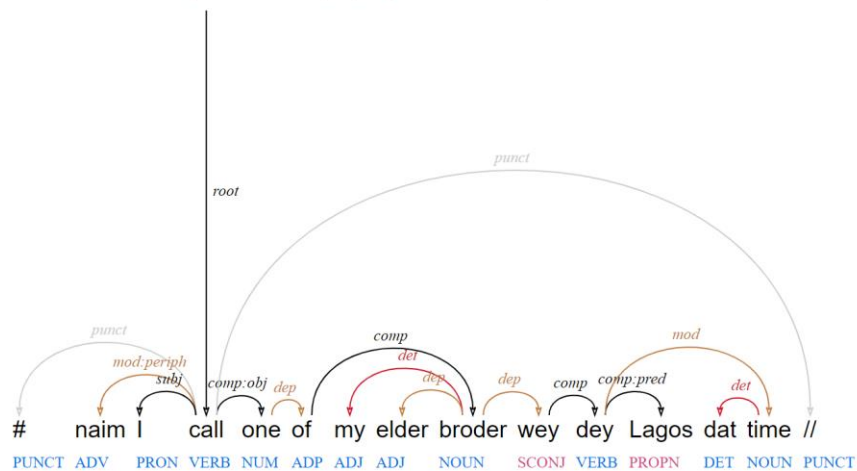
The fields DEPS and MISC replace the obsolete fields PHEAD and PDEPREL of the CoNLL-X format. In addition, we have modified the usage of the ID, FORM, LEMMA, XPOSTAG, FEATS and HEAD fields as explained below.

The fields must additionally meet the following constraints:

- Fields must not be empty.
- Fields must not contain space characters.
- Underscore ( ) is used to denote unspecified values in all fields except ID. Note that no format-level distinction is made for the rare cases where the FORM or LEMMA is the literal underscore – processing in such cases is application-dependent. Further, in UD treebanks the UPOSTAG, HEAD, and DEPREL columns are not allowed to be left unspecified.

## 依存句法树:

11: # naim I call one of my elder broder wey dey Lagos dat time //



P\_ABJ\_GWA\_06\_Ugo-lifestory\_PRO\_11

# That is when I called one of elder brothers who was in Lagos then I told that him that borthor,

## 任务 1: Arborator 系统与导出 conll 文件 (注意! 若数据库更新, 说明 conll 树被修改, 需重新导出 conll 文件)

### 1. 网站: <https://arborator.ilpga.fr/>

用户名: yuchen

密码: songyu

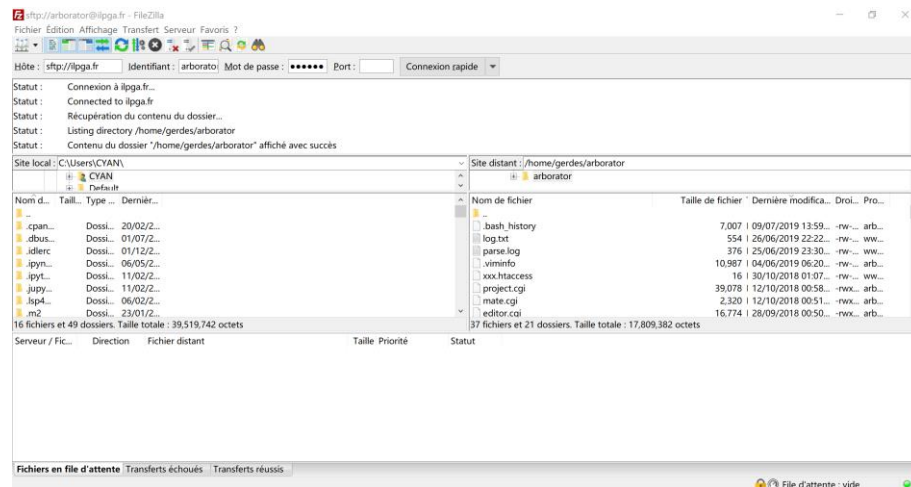
涉及项目: NaijaSUD

### 2. 服务器: 通过 FileZilla 进入

用户名: sftp://arborator@ilpga.fr:22 或者 fish://arborator@ilpga.fr:22

密码: annotate!t

项目路径: /home/gerdes/arborator/projects/NaijaSUD



### 3. 导出 conll 文件

- Arborator 网站直接导出：

The **NaijaSUD** Project has 80 texts and 7669 sentences.

text name	number of sentences	number of tokens	sentence length	annotators	validator	other trees	exo		
P_ABJ_GWA_03_Cost-of-living-in-Abuja_PRO.eaf.csv	126	1937	15.37	nobody yet	Bernard Caron	parser trees	no	assign	
P_ABJ_GWA_06_Ugo-lifestory_PRO	98	1651	16.85	nobody yet	Bernard Caron	parser trees			
P_ABJ_GWA_08_Davids-lifestory_PRO.eaf.csv	186	2680	14.41	nobody yet	Bernard Caron	parser trees			
P_ABJ_GWA_09_Journalism_PRO.eaf.csv	55	883	16.05	nobody yet	Bernard Caron	parser trees			
P_ABJ_GWA_10_Steven-lifestory_PRO	219	2962	13.53	nobody yet	Bernard Caron	parser trees			

#### Exporting P\_ABJ\_GWA\_09\_Journalism\_PRO.eaf.csv (text number 77 )

Exported 1 files and a total of 55 sentences.

Exported complete files for each assignment into P\_ABJ\_GWA\_09\_Journalism\_PRO.eaf.csv.user.complete.rhps.xml files.

The export file for most recent modification (including trees from yuchen, maigida, marine4maigida) is in the export folder **inside the project folder** on the server.

下载时可直接通过网站：

← → ↺ <https://arborator.ilpqa.fr/projects/NaijaSUD/export/?C=M;O=D>

## Index of /projects/NaijaSUD/export

Name	Last modified	Size	Description
Parent Directory	-	-	-
<a href="#">P_ABJ_GWA_09_Journalism_PRO.eaf.csv.most.recent.trees.conll10</a>	2019-07-09 10:23	71K	
<a href="#">P_ABJ_GWA_03_Cost.of.living.in.Abuja_PRO.eaf.csv.most.recent.trees.conll10</a>	2019-07-09 10:23	158K	

也可以通过 FileZilla: /home/gerdes/arborator/projects/NaijaSUD/export

Site distant: /home/gerdes/arborator/projects/NaijaSUD/export

Nom de fichier	Taille de fichier	Dernière modifica...	Droi...	Pro...
..				
...				
P_ABJ_GWA_09_Journalism_PRO.eaf.csv...	72,696	09/07/2019 16:23...	-rw...	ww...
P_ABJ_GWA_03_Cost.of.living.in.Abuja_...	161,590	09/07/2019 16:23...	-rw...	ww...
P_ABJ_GWA_06_Ugo.lifestory_PRO.mos...	136,539	08/07/2019 10:30...	-rw...	ww...
P_WAZP_07_Imonirhuas.Life.Story_PRO...	160,039	28/06/2019 10:27...	-rw...	ww...
P_WAZP_04_Ponzi.Scheme_PRO.most.r...	264,427	28/06/2019 10:26...	-rw...	ww...
P_WAZP_03_Education_PRO.most.recen...	141,934	28/06/2019 10:24...	-rw...	ww...
P_WAZL_15_MC.Abbey_PRO.most.rece...	225,307	28/06/2019 10:22...	-rw...	ww...
...				

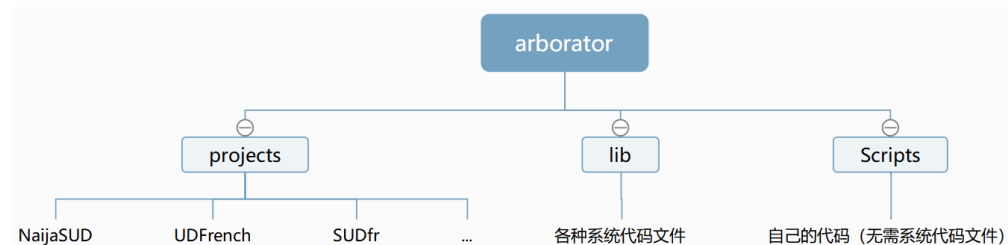
280 fichiers. Taille totale : 27,919,950 octets

- 通过 Marine 给的脚本: `yuchen_ExportNewestTree.py`
  - Python 2 版本, 运行: `py -2 yuchen_ExportNewestTree.py` (因涉及引入其他代码文件, 故需放在 `lib` 文件夹使用)
  - 漏洞:
    - 近一半以上的 text 在导出时会报错, 无法生成导出文件。
    - 会重新命名 `sent_id`, 导致在上传单句时无法通过数据库建立联系。
    - 不建议使用, 推荐直接在 Arborator 上直接导出下载。

#### 4. 重命名导出的 conll 文件

- 通过 Arborator 网站导出的 conll 文件以 `.most.recent.trees.conll10` 为后缀。
- 通过 Marine 的脚本导出的 conll 文件以 `.most.recent.trees.conll10` 为后缀 `.most.recent.trees.with.feats.conllu` 或 `.most.recent.trees.with.feats.conllu_reordered` 为后缀。
- 需改为 `.conll` 为后缀。
- 代码文件: 用于重命名文件 `.py`

#### 5. 文件目录结构:



## 任务 2：构建该语种词典，根据核对后的词典进行查错改错。范围：词(mot)，词性(cat)，句法特征(trait)。

### 1. 构建词典(Construire un dictionnaire)：

- 代码文件： *Get\_lexique* 可询问版本.py
- 思路：从 conll 格式的文件中提取词列、词性列、词干列和句法特征列，写入格式为 tsv 的文件（以制表符划分）中。

	A	B	C	D	E
1	mot	cat	trait	lemme	commentaires
2	!//	PUNCT	-	-	
3	#	PUNCT	-	-	
4	&	PUNCT	-	-	
5	&	X	Aspect=Im	-	
6	&	X	-	-	
7	//	PUNCT	-	-	
8	//+	PUNCT	-	-	
9	//=	PUNCT	PronType=-	-	

第 1 列：词 mot  
第 2 列：词性 catégorie  
第 3 列：句法特征 trait  
第 4 列：词干 lemme  
第 5 列：注释 commentaires

#### • 如何提取并写入信息：

- 按行读取 conll 文件，以制表符为界限将字符串分割为列表。

11 wheder SCONJ 10 comp:obj 15000|15770

- 以制表符为界，提取列表中的目标项组成一行新的字符串，即

terme = mot + "\t" + cat + "\t" + trait + "\t" + lemme + "\n", 写入词典文件。

#### • 函数功能：

- 构建全新的词典，包含所有词条。
- 构建只含有新词条的词典，故在写入信息时将与已构建的词典进行比对，只向新词典中写入已构建词典中未出现的词条。
- 合并词典，将新词典的词条写入旧词典。



## 2. 构建词典并计算词频 (Construire un dictionnaire et calculer la fréquence) :

- 代码文件: *Get\_lexique* 可询问版本加词频. *py*
- 思路: 从 conll 格式的文件中提取词列、词性列、词干列和句法特征列, 计算词频, 写入格式为 tsv 的文件中。

	A	B	C	D	E	F
1	mot	cat	trait	lemme	fréquence	commentaires
2	l//	PUNCT	-	-	1	
3	#	PUNCT	-	-	143	
4	&	PUNCT	-	-	3	
5	&	X	Aspect=Im	-	1	
6	&	X	-	-	5	
7	//	PUNCT	-	-	94	
8	//+	PUNCT	-	-	1	
9	//=	PUNCT	PronType=	-	1	
10	//=	PUNCT	-	-	5	

第 1 列: 词 mot  
第 2 列: 词性 catégorie  
第 3 列: 句法特征 trait  
第 4 列: 词干 lemme  
第 5 列: 词频 fréquence  
第 6 列: 注释 commentaires

- 如何计算频率:
  - 按行读取 conll 文件, 以制表符为界限将字符串分割为列表。  
11 wheder                SCONJ                10 comp:obj                15000|15770
  - 以制表符为界, 提取列表中的目标项组成一行新的字符串, 即  
 $terme = mot + "\t" + cat + "\t" + trait + "\t" + lemme + "\n"$ 。
  - 将所有词条写入一个列表, 计算每个元素在列表中的出现次数即为词频。以字典方式记录, 即 {词条: 词频}。
  - 依次读取字典中的 key 和 value, 写入 tsv 格式的词典文件中。
- 函数功能:
  - 构建全新的词典, 包含所有词条。
  - 构建只含有新词条的词典, 故在写入信息时将与已构建的词典进行比对, 只向新词典中写入已构建词典中未出现的词条。因涉及计算词频, 若使用此功能时, 将会额外生成一个只包含非新词的词典文件, 为合并词典时词频叠加用。
  - 合并词典, 将新词典的词条写入旧词典。  
若两个词典的指定 conll 文件完全相同, 即两次运行均使用同样的 conll 文件, 无需词频叠加, 故可直接将新词典的词条写入旧词典。  
若两次运行使用完全不同的 conll 文件, 将对两次使用的 conll 文件中共有的旧词进行词频叠加, 再写入新词条。  
注意! 若第二次输入的 conll 文件既包含第一次输入的又包含新的 conll 文件, 词频将计算混乱。

### 3. 查错改错（以 conll 文件查看）：

- 发现新词，改正 mot（大写或首字母大写）&cat&trait, 人工改 mot 拼写错误，cat 一对多模糊。

- 代码文件： *Correction\_conll\_version.py*

- 思路：

- 构建三个列表, 用于分别比对一致性:
  - lexique = [[mot1, cat1, trait1], [mot2, cat2, trait2], ...]
  - mot = [[mot1], [mot2], ...]
  - mot\_catégorie = [[mot1, cat1], [mot2, cat2], ...]
- 按行读取文件，检查与词典中的词条是否一致，若一致，直接写入新文件，若不一致，查错改错并写入新文件。

按行读取目标conll文件，按行写入同名新conll文件，改正即重写文件。

```
if mot correct:
    if mot&cat correct:
        if mot&cat&trait correct:
            写入此行
        else:
            改正trait并写入
    else:
        if 词典中的该词只有一个词性:
            改正cat并写入
        elif 词典中的该词有多个词性:
            将错误写入cat_ambiguïté.txt文件，需人工核对后在conll文件直接修改。并写入原行
elif 该词的大写在mot列表中:
    改成大写，写入此行
elif 该词的首字母大写在mot列表中:
    改成首字母大写，写入此行
else:
    发现新词写入mot_à_vérifier.txt文件进行核对，或者拼写错误需人工在conll文件修改。
    将所在行写入position_mot.txt文件，以便在conll文件中找到位置并修改。
    写入原行
```

- 漏洞：

- 不能自动修改本该为小写的词，在 conll 文件中大写或首字母大写。
- 省略了发现新的 cat 这一步。
- 不能发现新的 trait。

- 功能：

- 读取目录里所有文件（该文件夹只能有 conll 文件），进行查错改错。
- 读取指定 conll 文件（在同一文件夹），进行改错查错。

#### 4. 查错改错（点击链接在 arborator 上查看）：

- 发现新词，改正 mot（大写或首字母大写）&cat&trait, 人工改 mot 拼写错误，cat 一对多模糊。
- 代码文件： *Correction\_link\_version.py*
- 思路：
  - 构建三个列表, 用于分别比对一致性:
    - lexique = [[mot1, cat1, trait1], [mot2, cat2, trait2], ...]
    - mot = [[mot1], [mot2], ...]
    - mot\_catégorie = [[mot1, cat1], [mot2, cat2], ...]
  - 依次将 conll 文件中一句话的行写入临时文件，该文件只包含一句 conll，按行读取临时文件，检查与词典中的词条是否一致，若一致，直接写入新文件，若不一致，查错改错并写入新文件。与 Correction\_conll\_version.py 不同，本代码并不是按行一下子处理 conll 文件，而是按句处理，以便查询其在数据库中的所需信息，从而建立链接。

按句读取目标conll文件，通过文件名在数据库中找到相应的textid，每句话写入临时文件，按行读取临时文件，查错改错写入同名新conll文件，改正即重写文件。  
在临时文件中，包含一个句子，通过textL，textid找到句子编号sentenceid。  

```
if mot correct:
    if mot&cat correct:
        if mot&cat&trait correct:
            写入此行
        else:
            改正trait并写入
    else:
        if 词典中的该词只有一个词性:
            改正cat并写入
        elif 词典中的该词有多个词性:
            将错误写入cat_ambiguïté.txt文件。
            建立句子链接link = "https://arborator.ilpqa.fr/editor.cgi?project=NaijaSUD&textid="+textid+"&opensentence="+
            sent_id, 写入cat_ambiguïté.txt文件。
            并写入原行
elif 该词的大写在mot列表中:
    改成大写，写入此行
elif 该词的首字母大写在mot列表中:
    改成首字母大写，写入此行
else:
    发现新词写入mot_à_vérifier.txt文件进行核对，或者拼写错误需人工修改。
    建立句子链接link = "https://arborator.ilpqa.fr/editor.cgi?project=NaijaSUD&textid="+textid+"&opensentence="+sent_id, 写入
    position_mot.txt文件。
    写入原行"
```

- 漏洞：
  - 不能自动修改本该为小写的词，在 conll 文件中大写或首字母大写。
  - 省略了发现新的 cat 这一步。
  - 不能发现新的 trait。

### 任务 3：整理数据库 /home/gerdes/arborator/projects/NaijaSUD/arborator.db.sqlite

#### 1. 整理 sent\_id：

- 要求：

- Renuméroter les phrases, c'est-à-dire les sentenceid doivent être unique et s'enchaînés (1, 2, 3, ...).
- A la fin, toute phrase doit avoir un bon sent\_id.
- Attention : sent\_id est un code du type comme "WAZK\_11\_004", sentenceid est un numéro.
- 即每个句子都有其唯一的 sent\_id, 句子的 sent\_id 是连续的, 在 Arborator 上的体现为, 序号为 1 的句子, 其 sent\_id 以 001 结尾, 序号为 178 的句子, 其 sent\_id 以 178 结尾, 从 1 开始, 直到最后一句。
- 涉及表: sentences 和 sentencefeatures, 共有 key => sentences.rowid = sentencefeatures.sentenceid

- 数据库的问题：

- (1). Arborator 上的 text 和导出的 conll 文件的#text 均来自于数据库中的 sentences 表的 sentence 栏, sent\_id 来自于 sentencefeautres 表。然而 sentences.sentence 并不全部存在于 sentencefeatures 表, 即 sentences.rowid 在 sentencefeatures.sentenceid 中不存在; 或者即使存在, 在 sentences.rowid = sentencefeatures.sentenceid 的情况下, sentences.sentence 和 sentencefeatures.text 不一样 (少了个空格或者其他情况)。

此时会导致 Arborator 上的句子和导出的 conll 文件的句子没有 sent\_id。因为通过 sentences.sentence 在 sentencefeatures 找 text 找不到, 故联系不到 sent\_id。比如：

数据库结构		浏览数据	编辑备注	执行 SQL
表(T): sentences				
nr	sentence	textid		
过滤	as I talk before < " ehñ " de born me for Jos //	过滤		
1 2	as I talk before < " ehñ " de born me for Jos //	1		

```
1 select rowid,*
2 from sentences
3 where sentence = 'as I talk before < " ehñ " de born me for Jos //'
4
5
```

rowid	nr	sentence	textid
1 178	2	as I talk before < " eh...	1

其 rowid 为 178, 但在 sentencefeatures 表里, 没有 sentenceid=178。

```
1 SELECT *
2 FROM sentencefeatures
3 where sentenceid = 178
```

```
Result: 0 行返回, 耗时 38ms
At line 1:
SELECT *
FROM sentencefeatures
where sentenceid = 178
```

- (2). Arborator 上的句子的 sent\_id 不连续, 出现断层, 比如:

26	text	me don see sey [ if you find ten married women for Nigeria now < " mstch...
26	sent_id	WAZK_11_022 WAZK_11_023 WAZK_11_024 WAZK_11_025 WAZK_11_0...
26	sent_translation	I have seen that, if you find ten married women in Nigeria now, mstchew I...
27	text	di remaining ones na im >+ de dey draw deir daughters ear dat [" eh " no...
27	sent_id	WAZK_11_026
27	sent_translation	will be happy in the marriage. The remaining ones they are the one that ar...
28	text	so < you go see woman wey be sey [ im pikin dey marry tomorrow ]//< e ...
28	sent_id	WAZK_11_027 WAZK_11_028
28	sent_translation	So, you will see a woman that her child is marrying tomorrow, she will no...
29	text	mstchew //
29	sent_id	WAZK_11_029
29	sent_translation	Mstchew.
30	text	X just lock your mind //
30	sent_id	WAZK_11_030 WAZK_11_031
30	sent_translation	[ inaudible words ] just lock your mind.
31	text	take di way you fader don do me //
31	sent_id	WAZK_11_032

WAZK\_11\_026 和 WAZK\_11\_027 WAZK\_11\_028 两句对于 sentences 表中的一句, 而在 sentencefeatures 表中被拆分了, 所以在 Arborator 上的句子来自 sentences 表是完整的, 但是 sent\_id 只能对应 WAZK\_11\_026, 因为其 sentenceid 和 sentences.rowid 对应, 即此情况的 sentences.rowid 只和涉及到的分割句子的第一分割部分的 sentenceid 对应。对应的可以修改, 不对应的则忽略, 即被拆分的非第一部分的句子的 sent\_id 可被忽略, 不做修改, 因为不会显示在 Arborator 和 conll 文件上。

- 修改方法:

- 首先, 需要在向 sentencefeatures 表里加入 sentences.rowid 不存在于 sentencefeatures.sentenceid 的句子, 对于每个要添加的句子, 向 sentencefeatures 表加两行, "sentenceid","text","phrase"和"sentenceid","sent\_id", "nr de 'sentences'"。

```

1  import sqlite3
2
3  conn = sqlite3.connect("arborator.db.sqlite")
4  c = conn.cursor()
5  cursor = c.execute("SELECT rowid,nr,sentence,textid FROM sentences WHERE not EXISTS( SELECT sentenceid FROM
   sentencefeatures WHERE sentencefeatures.sentenceid=sentences.rowid)")
6
7  #插入rowid不在sent_id的句子 Inserer les phrases de 'sentences' dont le rowid n'est pas dans 'sentencefeatures'
8  for row in cursor:
9      c = conn.cursor() #必须每次都重新连一次
10     sentenceid = row[0]
11     sent_id = row[1]
12     texte = row[2]
13     textid = row[3]
14     c.execute("INSERT INTO sentencefeatures VALUES (?,?),(sentenceid,'text',texte)")
15     c.execute("INSERT INTO sentencefeatures VALUES (?,?),(sentenceid,'sent_id',sent_id)")

```

- 重命名 sent\_id (只修改 sentences.rowid=sentencefeatures.sentenceid 的句子)。在 sentences 表中, nr 表示每篇语料的句子序号, 是从 1 开始连续且唯一的, 所以重命名的 sent\_id 由 text name 和 nr 组成。

```

17 #重命名sent_id renommer sent_id
18 c.execute("""update sentencefeatures
19 set value=
20 (select texts.textname from texts where texts.rowid=(select sentences.textid from sentences where
21 sentences.rowid=sentencefeatures.sentenceid) )
22 ||
23 "-"
24 ||
25 (select sentences.nr from sentences where sentences.rowid=sentencefeatures.sentenceid)
26 where attr="sent_id"
27 and sentencefeatures.sentenceid in (select rowid from sentences);""")
28
29 #nettoyer du sent_id, supprimer '_TRANS.eaf.csv' et '.eaf.csv'
30 #删除sent_id中的"_TRANS.eaf.csv"
31 c.execute("""update sentencefeatures
32 set value=replace(value,"_TRANS.eaf.csv","")
33 where attr="sent_id";""")
34
35 #删除sent_id中的".eaf.csv"(主要在以PRO结尾的文件)
36 c.execute("""update sentencefeatures
37 set value=replace(value,".eaf.csv","")
38 where attr="sent_id";""")
39
40 conn.commit()
41 conn.close()

```

结果, 所有显示在 arborator 上的句子的 sent\_id 是连续且唯一的。

1	1	text	anyway <{ me    me } na person wey grow up for Jos //
2	1	sent_id	WAZK_11_M-Chiagozies-Life-Story_1
3	1	sent_translation	Anyway I, I am a person that grew up in Jos. As I said before, I grew up in Jos.
4	2	text	as I talk before < " ehm " dey born me for Jos //
5	2	sent_id	WAZK_11_002
6	2	sent_translation	a person that grew up in Jos. As I said before, I grew up in Jos.
7	3	text	na dis work sha < na im >+ carry me come Kano //
8	3	sent_id	WAZK_11_003 WAZK_11_004
9	3	sent_translation	It is really this work that brought me to Kano. I came to this town in two thousan...
10	4	text	I come dis town two thousand and eleven //
11	4	sent_id	WAZK_11_M-Chiagozies-Life-Story_4

- 统一 sentences.sentence = sentencefeatures.text : 统一 sentence 和 text.sql

```

update sentencefeatures
set value=

```

```

    (select sentences.sentence from sentences where sentences.rowid=sentencefeatures.sentenceid)
where attr="text"
and sentencefeatures.sentenceid in (select rowid from sentences);

```

- 代码文件: *Modify\_sent\_id.py*

## 2. 整理时间标记 (Unifier les alignements temporels) :


- Dans le tableau "features", s'il y a une pipe dans value misc, par exemple : 325651|325839. Pour chaque misc, insert startali, insert endali, delete misc et xpos, par exemple : startali=325651, endali=325839, mettre startali et endali sur tous les nœuds, ensuite tout arbre sur tous les noeuds doit avoir ces deux traits ! Il n'y a plus de traits misc ou xpos.
- 涉及情况及修改方法：
  - 情况 1: xpos 和 misc 的值为 "\_", startali 和 endali 均有数值，只需删除 xpos 和 misc 两行。

表(T): features

	treeid	nr	attr	value
	过滤	过滤	过滤	过滤
1	4	1	xpos	_
2	4	1	misc	_
3	4	1	startali	9524
4	4	1	lemma	_
5	4	1	tag	_
6	4	1	t	I
7	4	1	id	1
8	4	1	endali	9736

- 情况 2: 没有 startali 和 endali，词的时间起始结束标记在 misc 值，需取相应数值添加 startali 和 endali 两行，删除 xpos 和 misc。

230148	12463	3	index	3
230149	12463	3	xpos	_
230150	12463	3	misc	406000 406305
230151	12463	3	lemma	_
230152	12463	3	tag	PROPN
230153	12463	3	t	Buhari
230154	12463	3	id	3

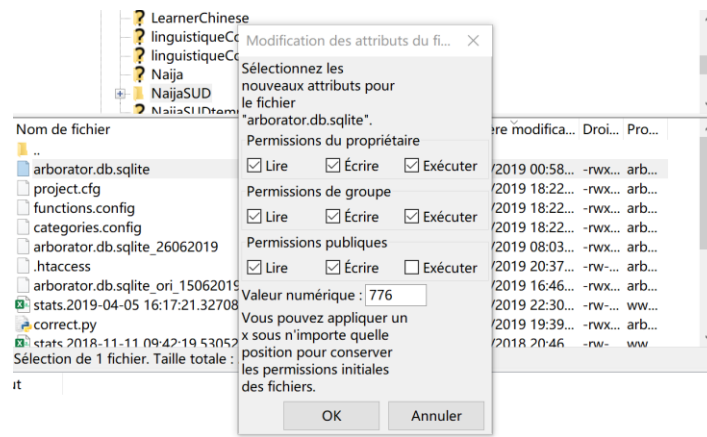


12463	3	index	3
12463	3	lemma	_
12463	3	tag	PROPN
12463	3	t	Buhari
12463	3	id	3
12463	3	startali	406000
12463	3	endali	406305

- 代码文件：
  - 添加 startali 和 endali: *add\_ali.py*
  - 删除 xpos 和 misc: *delete\_misc\_xpos.py*

### 3. 上传已修改的数据库到 Arborator:

- 通过 FileZilla 链接 Arborator 服务器，找到 NaijaSUD 所在路径/home/gerdes/arborator/projects/NaijaSUD。
- 可选择保存旧数据库文件，需重命名该文件，比如 arborator.db.sqlite\_ori\_日期。
- 拖入已修改的数据库文件 arborator.db.sqlite 上传。
- 修改上传文件的权限。右键 Droits d'accès au fichier ...



- 再导出的 conll 文件的呈现效果:

```
# elan_id = P_ABJ_GWA_06_024 P_ABJ_GWA_06_025
# sent_id = P_ABJ_GWA_06_Ugo-lifestory_PRO_11
# sent_translation = # That is when I called one of elder brothers who was in Lagos then I told that him that borther,
# text = # naim I call one of my elder broder wey dey Lagos dat time //
1      #          PUNCT          startali=35567|endali=35913      4      punct      -
2      naim      ADV      startali=35913|endali=36133      4      mod:periph
3      I          PRON      Case=Nom|PronType=Prs|Number=Sing|startali=36133|Person=1|endali=36203      4      subj      -
4      call      VERB      startali=36203|endali=36402      0      root
5      one      NUM      NumType=Card|startali=36402|endali=36608      4      comp:obj      -
6      of      ADP      startali=36608|endali=36661      5      dep
7      my      ADJ      PronType=Prs|Number=Sing|Poss=Yes|startali=36661|Person=1|endali=37118      9      det      -
8      elder      ADJ      Degree=Cmp|startali=37118|endali=37395      9      dep      -
9      broder      NOUN      startali=37395|endali=37673      6      comp      -
10     wey      SCONJ      startali=37673|endali=37833      9      dep      -
11     dey      VERB      PartType=Cop|startali=37833|endali=37953      10     comp      -
12     Lagos      PROPN      startali=37953|endali=38323      11     comp:pred      -
13     dat      DET      PronType=Dem|Number=Sing|startali=38323|endali=38443      14     det      -
14     time      NOUN      startali=38443|endali=38673      11     mod      -
15     //      PUNCT      startali=38673|endali=38703      4      punct      -
```



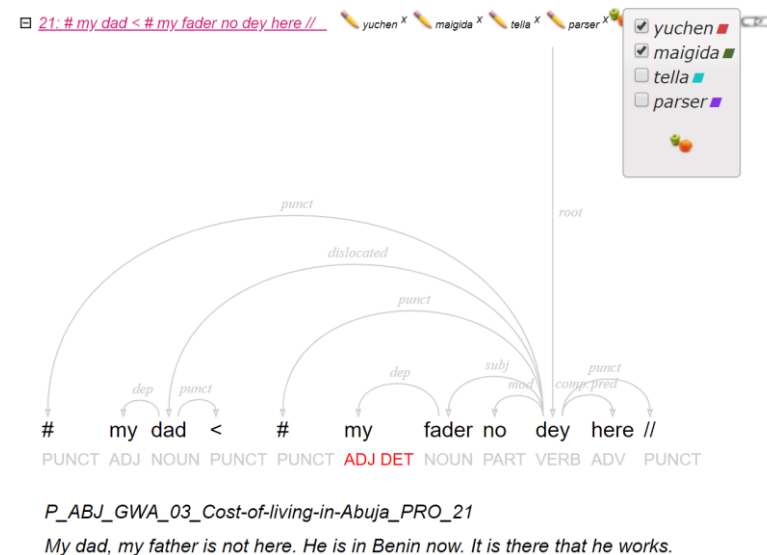
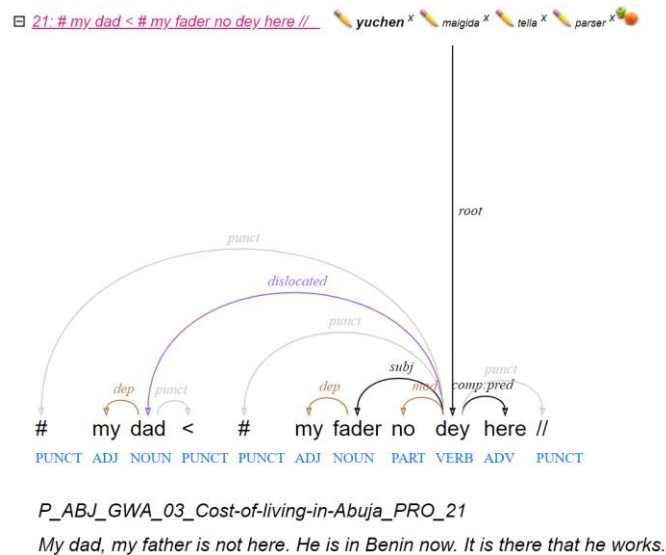
## 任务 4：只上传已修改的 conll 树，以 yuchen 命名 (Uploader tout arbre modifié sous le nom de yuchen)

### • 思路：

- 识别出已修改的句子：
  - 按句读取已修改的 conll 文件，将一句话的 conll 写入变量 `phrase`。
  - 读取相对应的（通过文件名建立关系）未修改的 conll 文件全部内容，写入变量 `contenu`。
  - 如果 `phrase` 不存在 `contenu` 中，说明该句已做修改，将写入数据库。
- 使用 `conllFile2trees` 函数，转换格式。
- 使用 `simpleEnterSentences` 函数，将句子写入数据库。

• 代码文件： `yuchen_EntreOnlyModifiedTree_NaijaSUD.py` (因涉及引入其他代码文件，故需放在 `lib` 文件夹使用)

• 呈现结果： `yuchen` 的已修改树已上传到相应的句子位置，通过比对，可看到被修改处，若通过树比对无区别，则说明可能在 `trait` 位置有修改。



## 任务 5：构建法语词典 (改写 Lefff)

- 思路：找到 Lefff 和 UD/SUD 的 `cat` 和 `trait` 之间可转换的共同点，进行转换，特殊项可手动修改。  
详见 Rapport-Dictionnaire\_FR.docx。
- 代码文件： *fonction\_dict.py*
- 注释：
  - SUD 和 UD 的 `trait` 的区别？ 无区别
  - `trait` 在 `arbre` 中的体现？ 用于 transformation de SUD au UD : `mod => a:mod`

## 未进行任务：

### 1. Identifier les noyaux

- 格式：prénoyau < prénoyau < noyau > postnoyau //
- 例子：

```
# elan_id = ABJ_GWA_03_M_013
# sent_id = P_ABJ_GWA_03_Cost-of-living-in-Abuja_PRO_8
# sent_translation = it is here that my mama and my papa gave birth to me. So, it's here that I grew up in Abuja.
# text = so na here >+ { I || I } grow up for Abuja //
```

1	so	-	ADV	-	2	discourse	-	16330 16500	MacroPré=B	
2	na	-	PART	-	0	root	-	16500 16688	MacroPré=I	
3	here	-	ADV	-	2	comp:pred	-	16688 16791	MacroPré=N	
4	>+	-	PUNCT	-	10	punct	-	16791 16821		
5	{	-	PUNCT	-	6	punct	-	16791 16821	MacroPost=B	
6	I	-	PRON	-	Case=Nom PronType=Prs Person=1 Number=Sing	10	subj	-	MacroPost=I	
7		-	PUNCT	-	8	punct	-	17010 17040	MacroPost=I	
8	I	-	PRON	-	Case=Nom PronType=Prs Person=1 Number=Sing	6	conj:dicto	-	17040 17210	MacroPost=I
9	}	-	PUNCT	-	6	punct	-	17210 17240	MacroPost=I	
10	grow	-	VERB	-	2	comp:cleft	-	17240 17480	MacroPost=I	
11	up	-	ADP	-	10	compound:prr	-	17480 17702	MacroPost=I	
12	for	-	ADP	-	10	mod	-	17702 17860	MacroPost=I	
13	Abuja	-	PROPN	-	12	comp	-	17860 18457	MacroPost=L	
14	//	-	PUNCT	-	2	punct	-	18457 18487		

```
# sent_id = P_BEN_34_Tale_PRO_2
# sent_translation = # So, she cooked porridge.
# text = # so < e con cook porridge //
```

1	#	-	PUNCT	-	5	punct	-		MacroPré=U
2	so	-	ADV	-	5	discourse	-		
3	<	-	PUNCT	-	2	punct	-		
4	e	-	PRON	-	Case=Nom Number=Sing Person=3 PronType=Prs	5	subj	-	MacroN=B
5	con	-	AUX	-	0	root	-		MacroN=I
6	cook	-	VERB	-	5	comp:aux	-		MacroN=I
7	porridge	-	NOUN	-	6	comp:obj	-		MacroN=L
8	//	-	PUNCT	-	5	punct	-		

### 2. 用 Grew 进行 conll 文件的查错改错（需学习 grew 语法并改写 python 的代码）

- 思路：def lexigrew(lexico\_ori, lexico\_new):
  - grammaire de réécriture grew déterministe (pour les erreurs sans ambiguïté)
  - grammaire de réécriture grew non déterministe (pour les erreurs d'ambiguïté)

