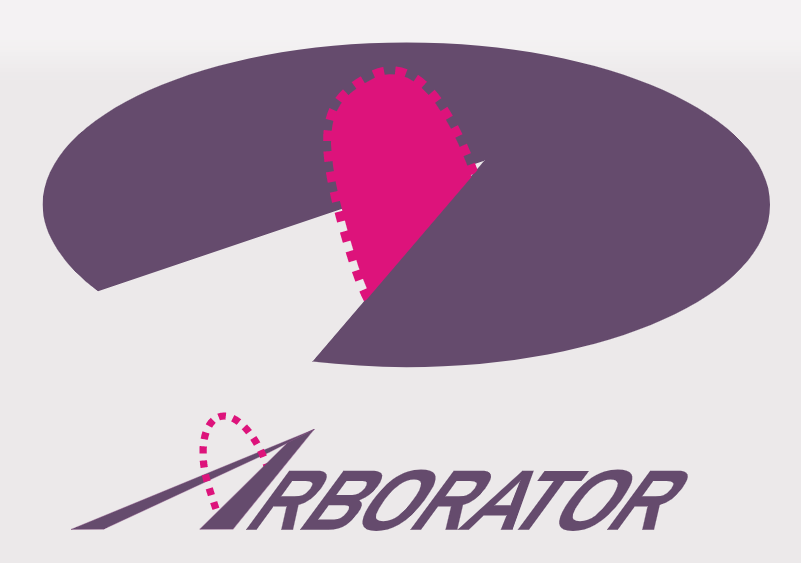
**NaijaSUD实习总结**



**导师及主要实习人员：**

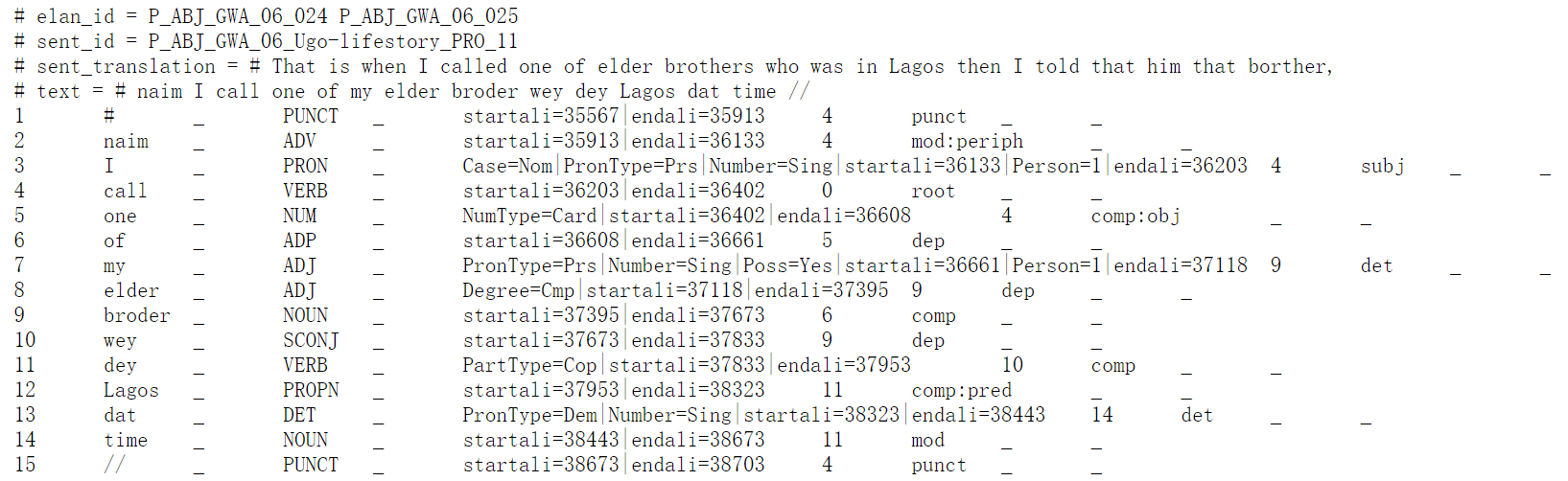
* Sylvain Kahane : [sylvain@kahane.fr](mailto:sylvain@kahane.fr)
* Kim gerdes : [kim@gerdes.fr](mailto:kim@gerdes.fr)
* Marine Courtin : [rinema56@gmail.com](mailto:rinema56@gmail.com)
* Bernard Caron : [Bernard.I.caron@gmail.com](mailto:Bernard.I.caron@gmail.com), [Bernard.CARON@cnrs.fr](mailto:Bernard.CARON@cnrs.fr)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

**Github :**

* <https://github.com/UniversalDependencies/UD_Naija-NSC>
* <https://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。



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**](https://universaldependencies.org/docs/format.html) **:**

We use a revised version of [the CoNLL-X format](http://ilk.uvt.nl/conll/#dataformat) 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:

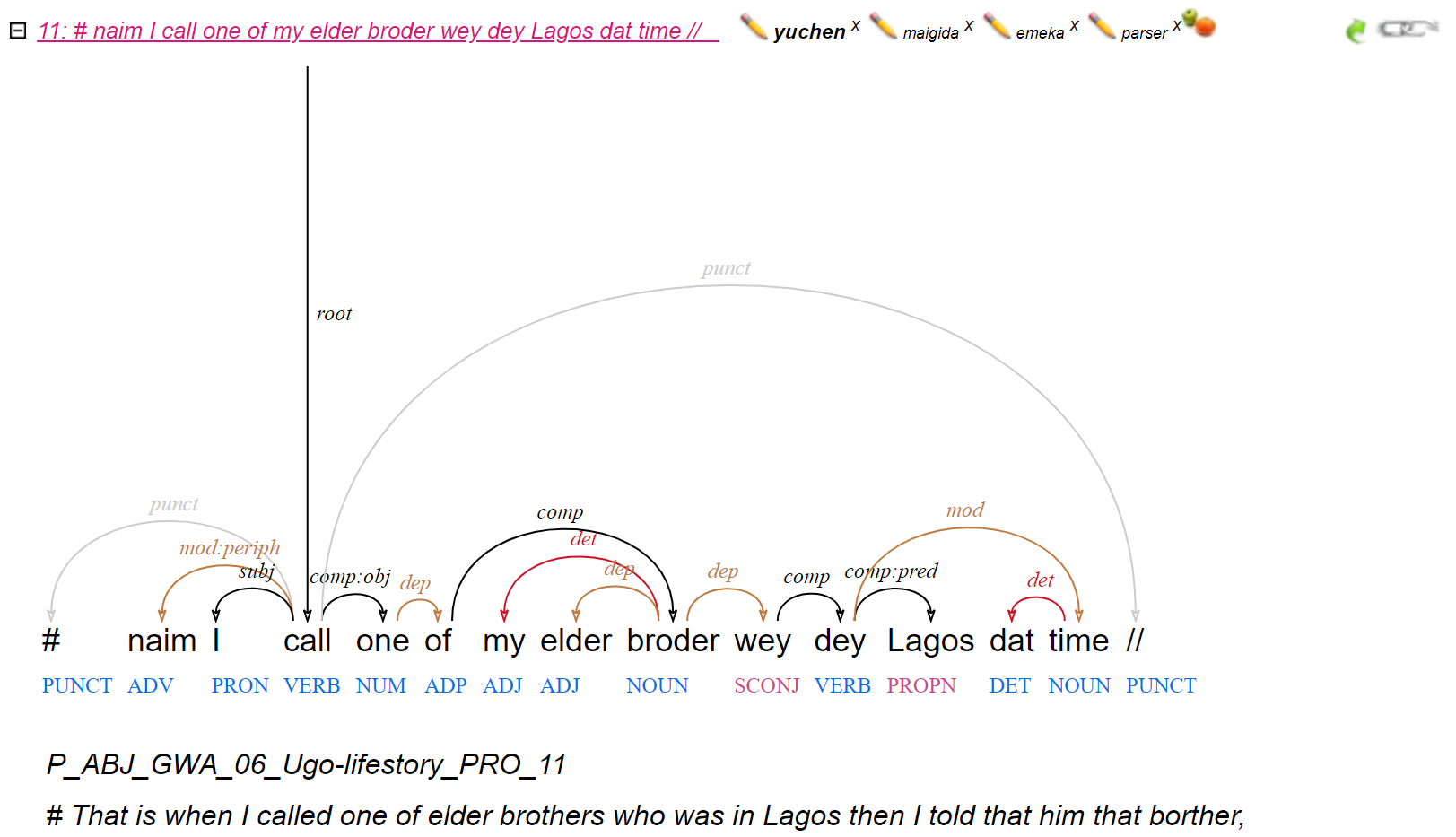
1. ID: Word index, integer starting at 1 for each new sentence; may be a range for tokens with multiple words.
2. FORM: Word form or punctuation symbol.
3. LEMMA: Lemma or stem of word form.
4. UPOSTAG: [Universal part-of-speech tag](https://universaldependencies.org/docs/u/pos/index.html) drawn from 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](https://universaldependencies.org/docs/u/feat/index.html) or from a defined [language-specific extension](https://universaldependencies.org/docs/ext-feat-index.html); 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](https://universaldependencies.org/docs/u/dep/index.html) 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：**

[](https://arborator.ilpga.fr/editor.cgi?project=NaijaSUD&textid=115&opensentence=1)

**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/**](https://arborator.ilpga.fr/)

用户名：yuchen

密码：songyu

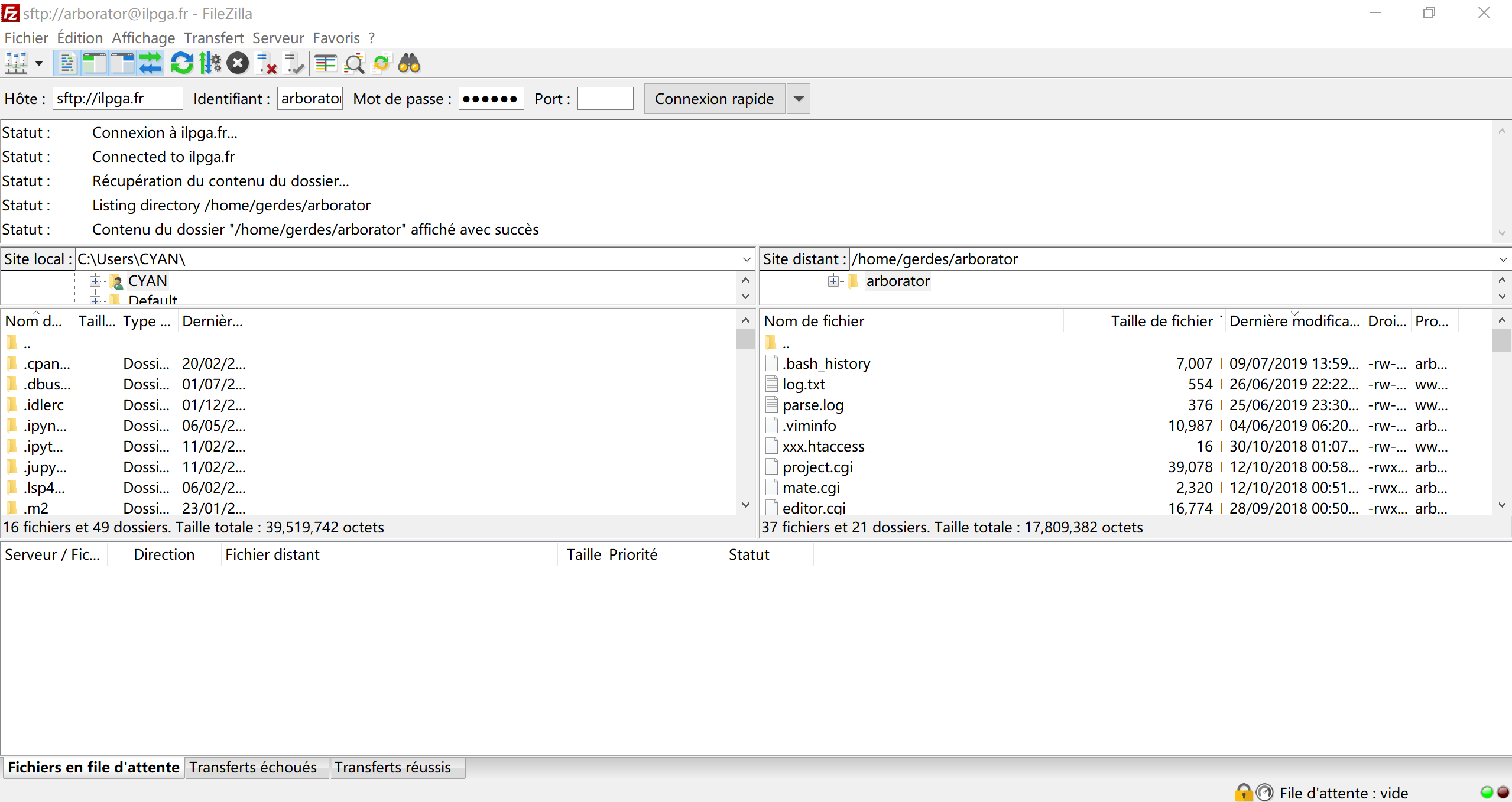
涉及项目：NaijaSUD

1. **服务器：通过FileZilla进入**

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

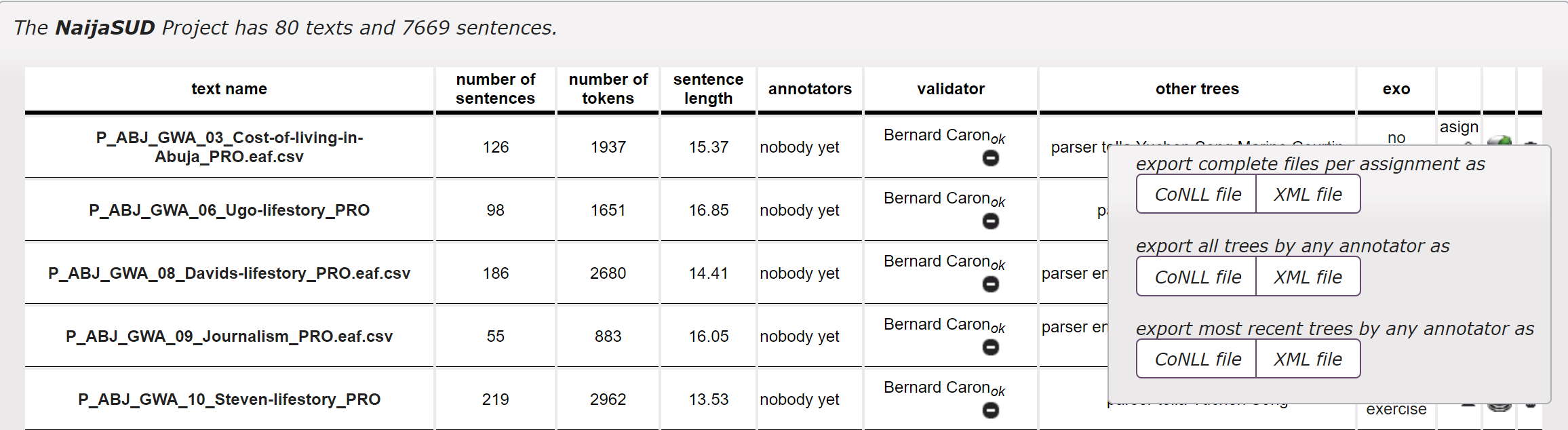
密码：annotate!t

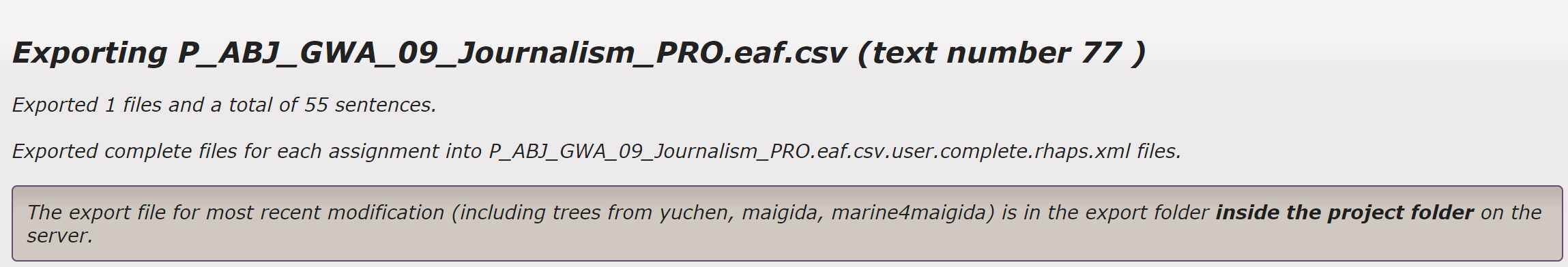
项目路径：/home/gerdes/arborator/projects/NaijaSUD



1. **导出conll文件**

• Arborator网站直接导出：

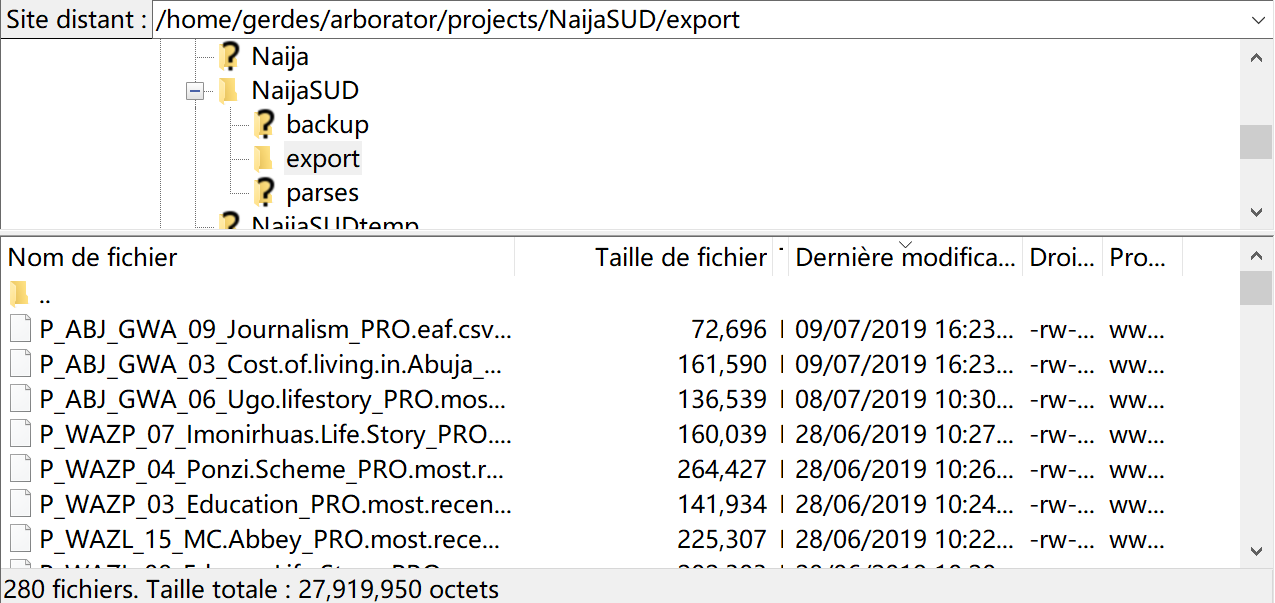




下载时可直接通过网站：



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



• 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。

1. **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。

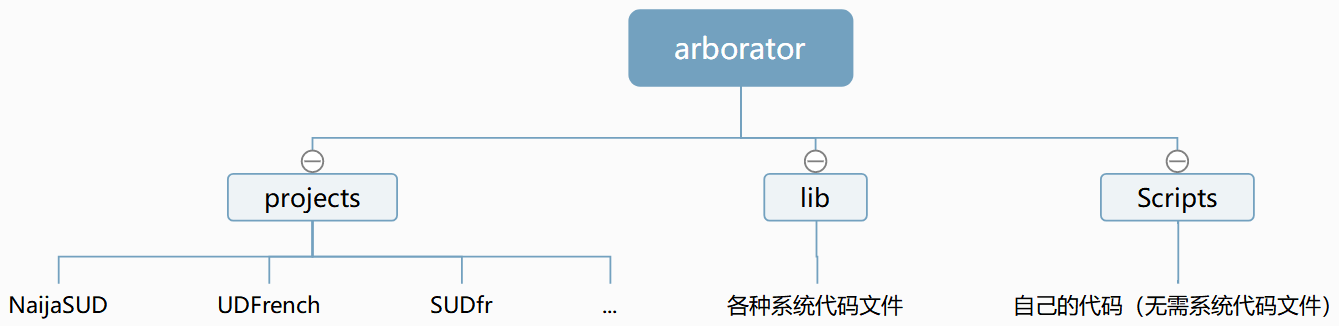
• 通过Marine的脚本导出的conll文件以.most.recent.trees.conll10为后缀.most.recent.trees.with.feats.conllu

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

1. **文件目录结构：**

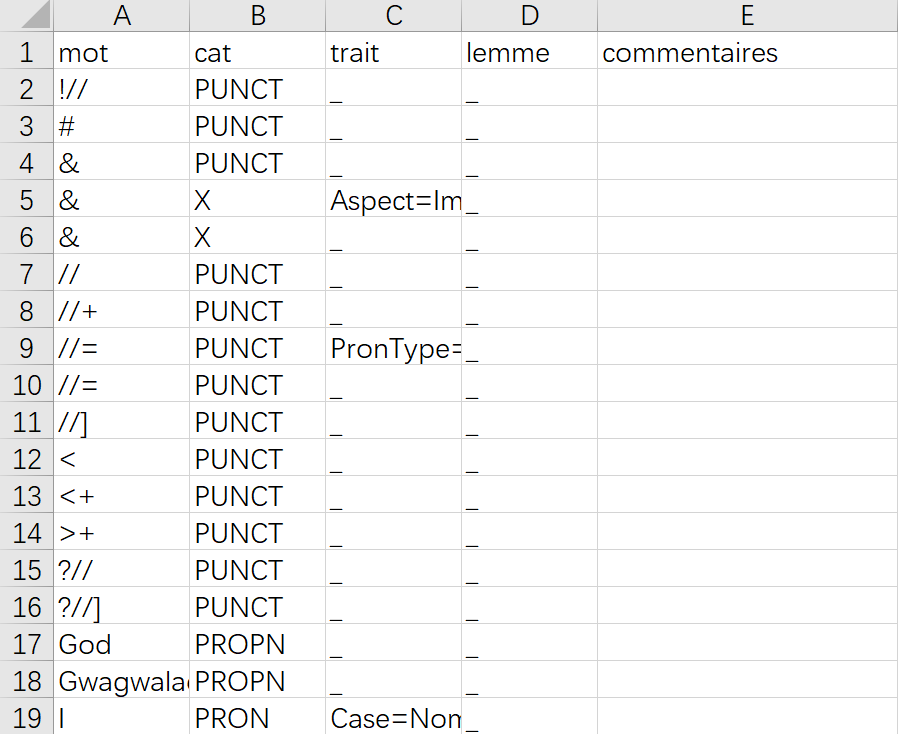


**任务2：构建该语种词典，根据核对后的词典进行查错改错。范围：词(mot)，词性(cat)，句法特征(trait)。**

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。



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。



* + 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。

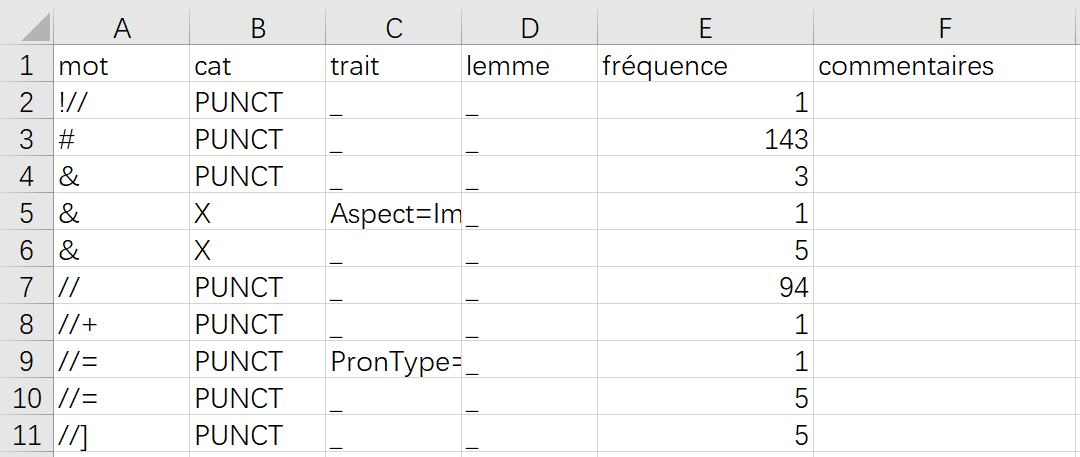
• 函数功能：

* + 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。

1. **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。



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。



* + 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。

1. **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：

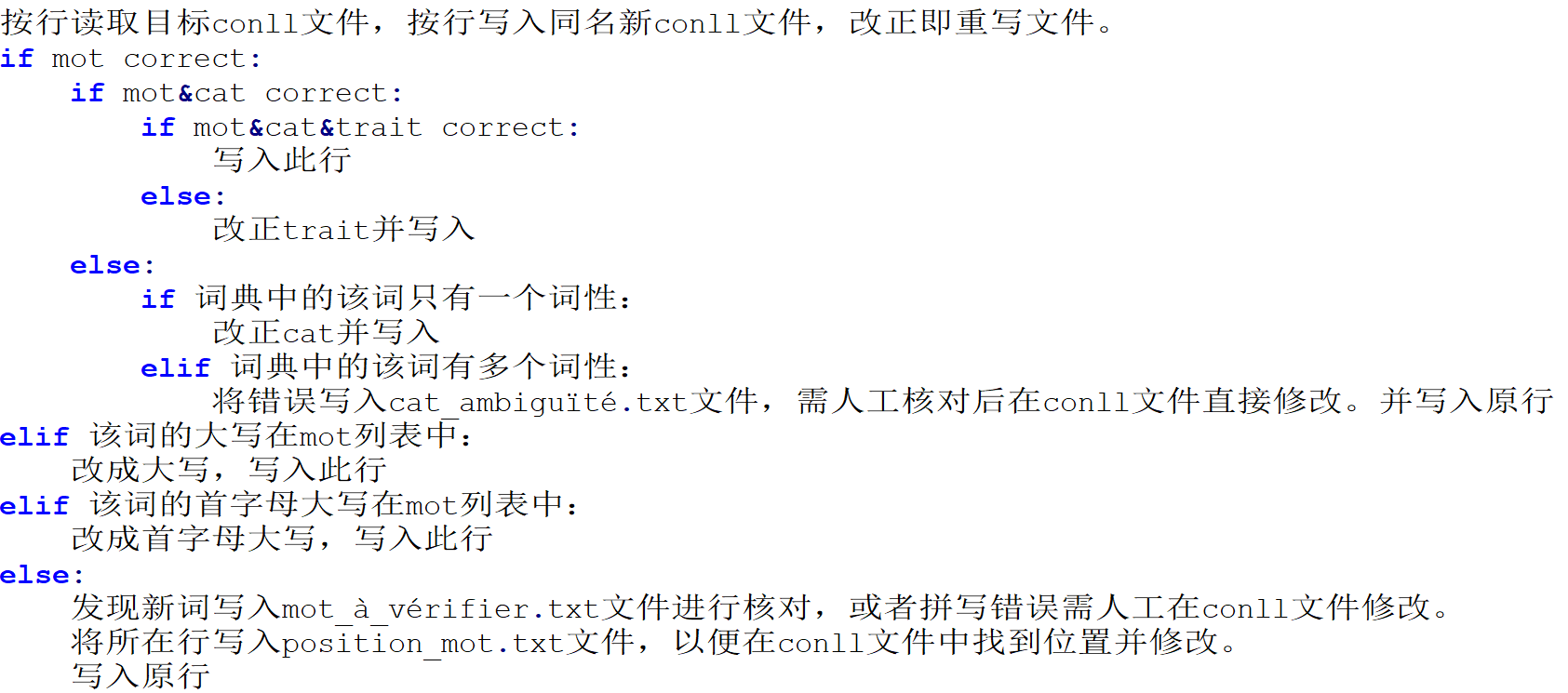
* + 构建三个列表,用于分别比对一致性：

- 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。



• 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。

1. **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。



• 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 :**

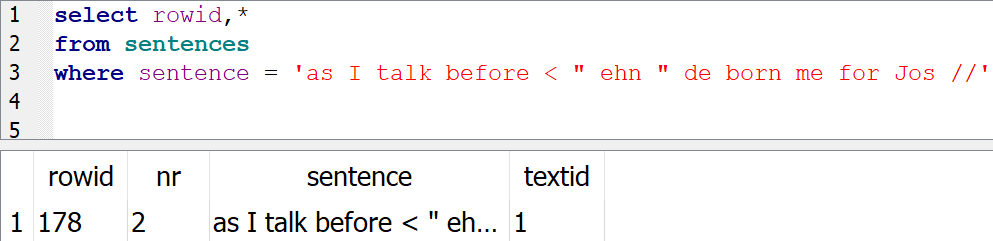
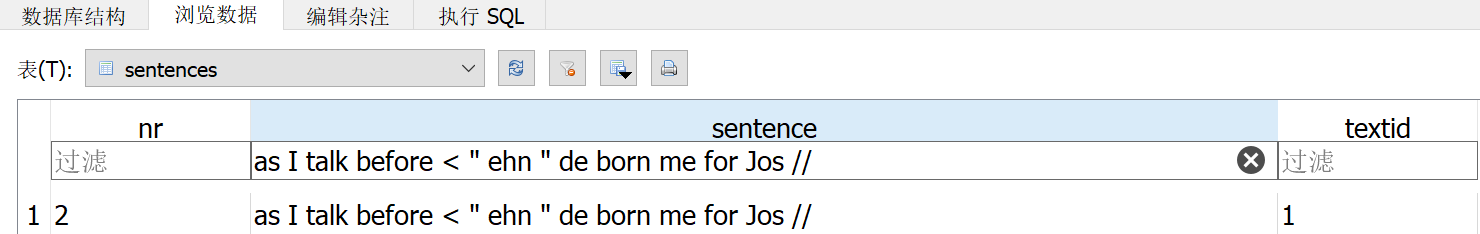
• 要求：

* + 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

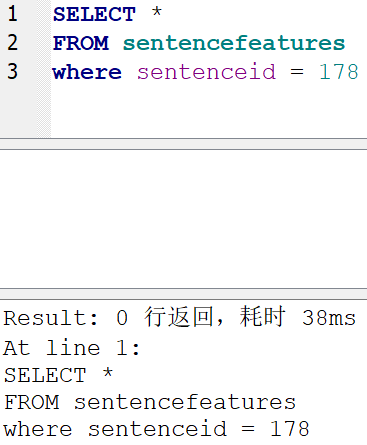
• 数据库的问题：

* + (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）。

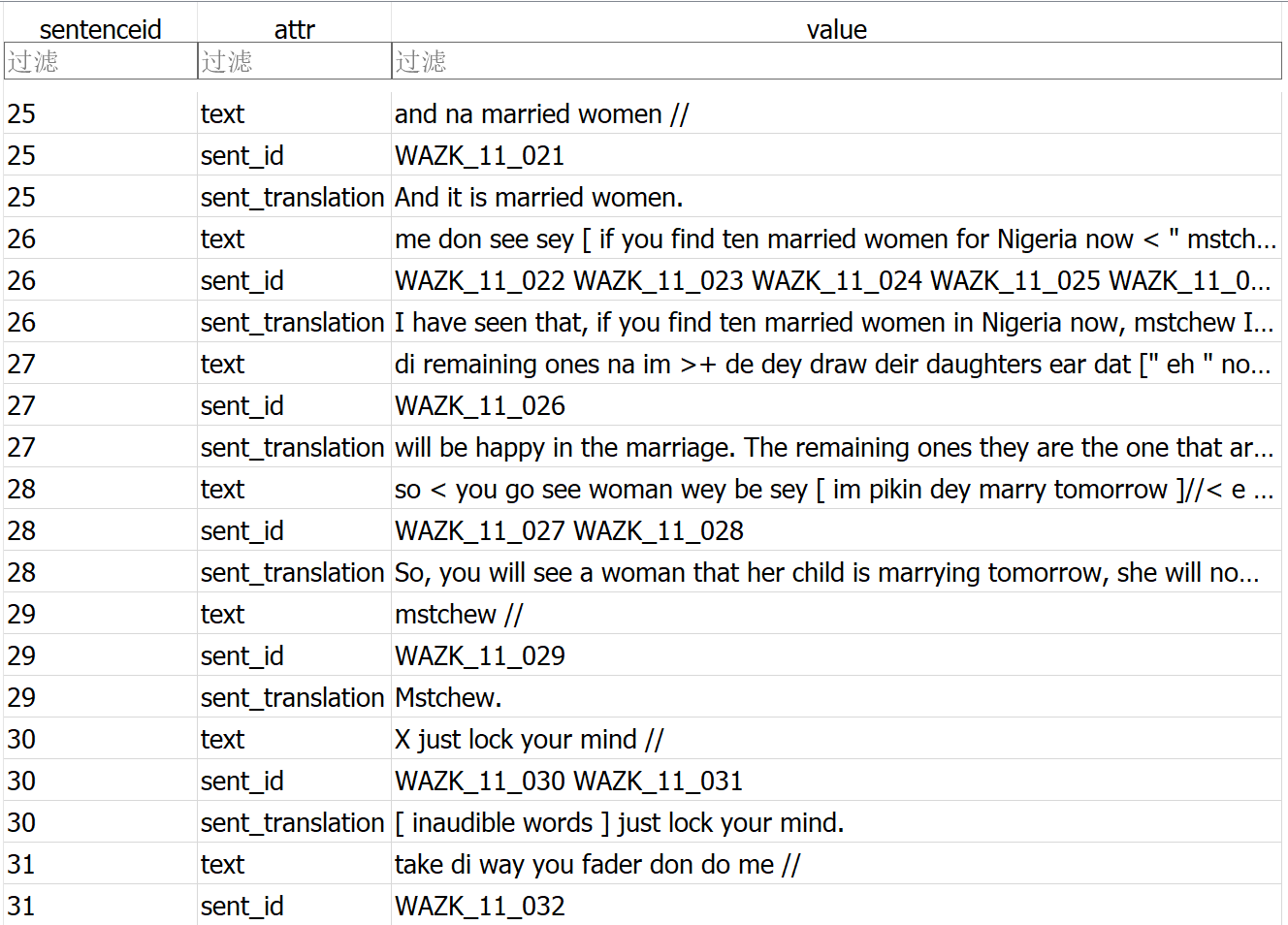
此时会导致Arborator上的句子和导出的conll文件的句子没有sent\_id。因为通过sentences.sentence在sentencefeatures找text找不到，故联系不到sent\_id。比如：



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



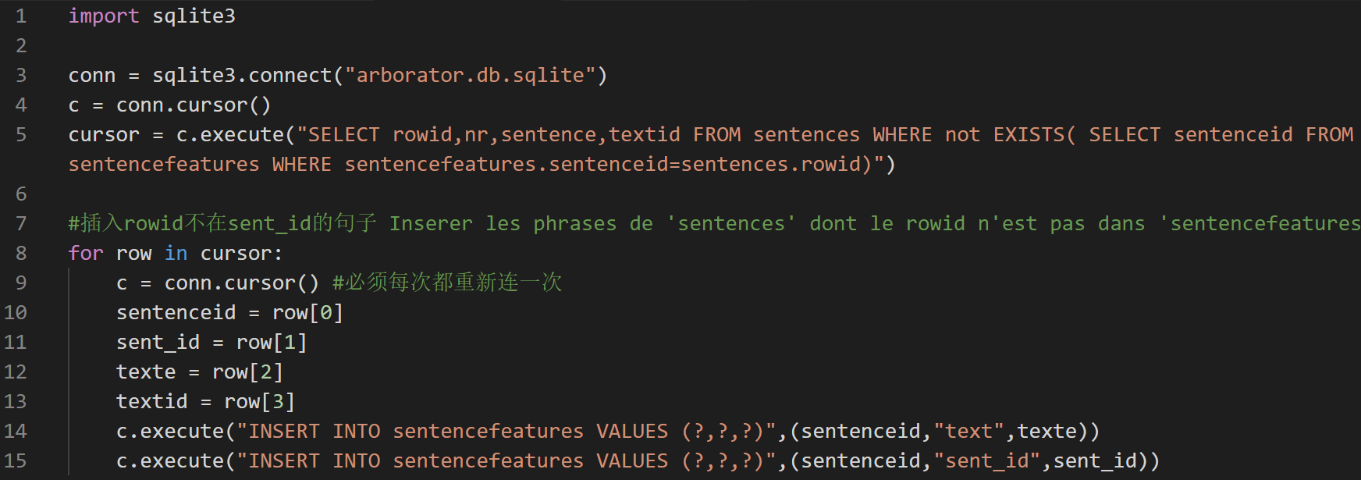
* + **(2).** Arborator上的句子的sent\_id不连续，出现断层，比如 :



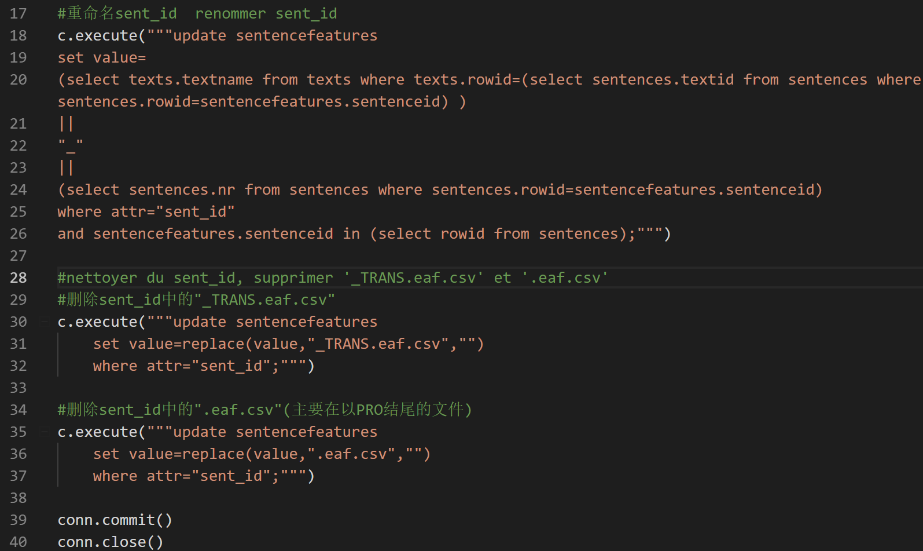
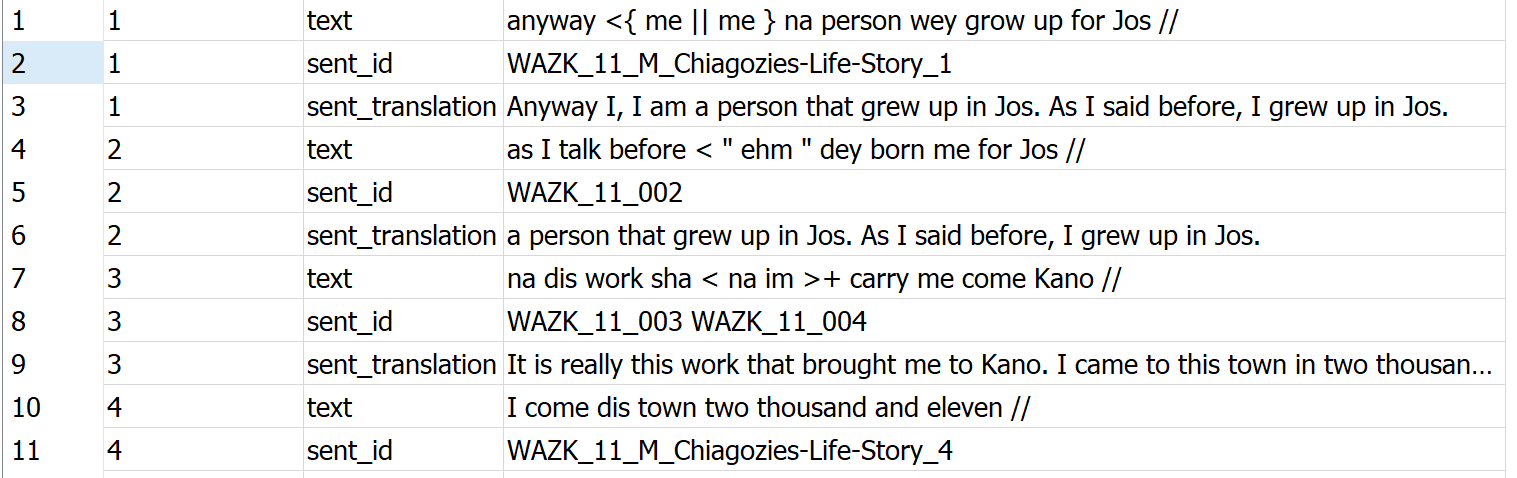
WAZK\_11\_026和WAZK\_11\_027 WAZK\_11\_028两句对于sentences表中的一句，而在sentencefeatures表中被拆分了，所以在Arborator上的句子来自sentences表是完整的，但是sent\_id只能对应WAZK\_11\_026，因为其sentenceid和sentences.rowid对应，即此情况的sentences.rowid只和涉及到的分割句子的第一分割部分的sentenceid对应。对应的可以修改，不对应的则忽略，即被拆分的非第一部分的句子的sent\_id可被忽略，不做修改，因为不会显示在Arborator和conll文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

• 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'"。

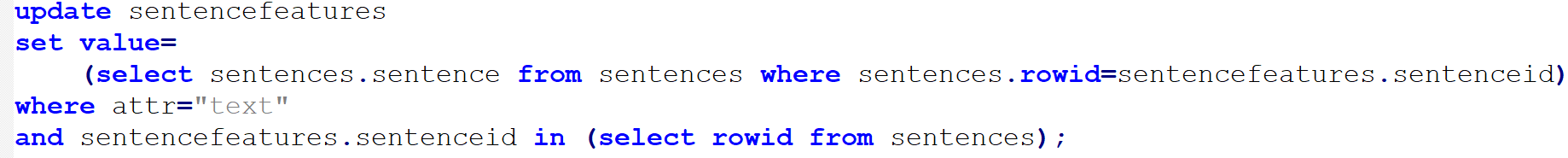


* + **重命名sent\_id（只修改sentences.rowid=sentencefeatures.sentenceid的句子）。在sentences表中，nr表示每篇语料的句子序号，是从1开始连续且唯一的，所以重命名的sent\_id由text name和nr组成。**



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

* + **统一sentences.sentence = sentencefeatures.text : *统一sentence和text.sql***

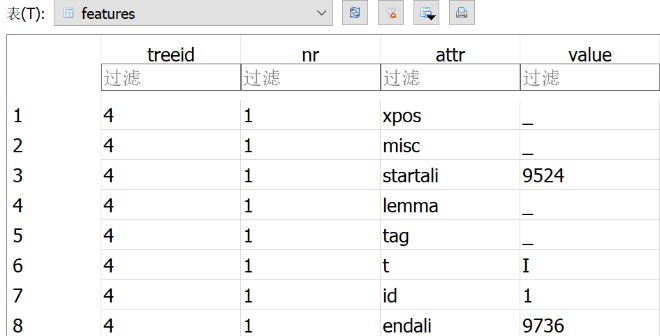


• 代码文件：*Modify\_sent\_id.py*

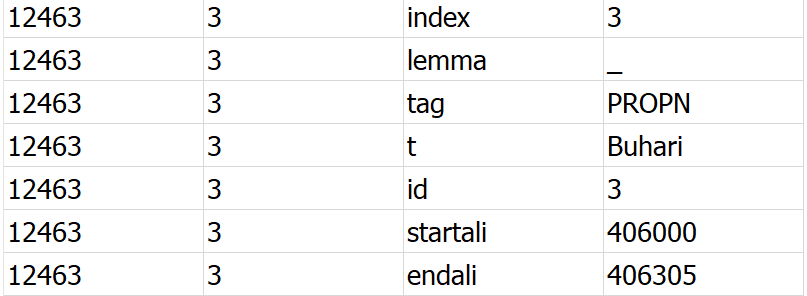
1. **整理时间标记 (Unifier les alignements temporels) :**
   * Dans le tableau "features", s'il y a une pipe dans value misc, par example : 325651|325839. Pour chaque misc, insert startali, insert endali, delete misc et xpos, par exemple : startali=325651, endali=325839, mettre startali et endali sur tous 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。



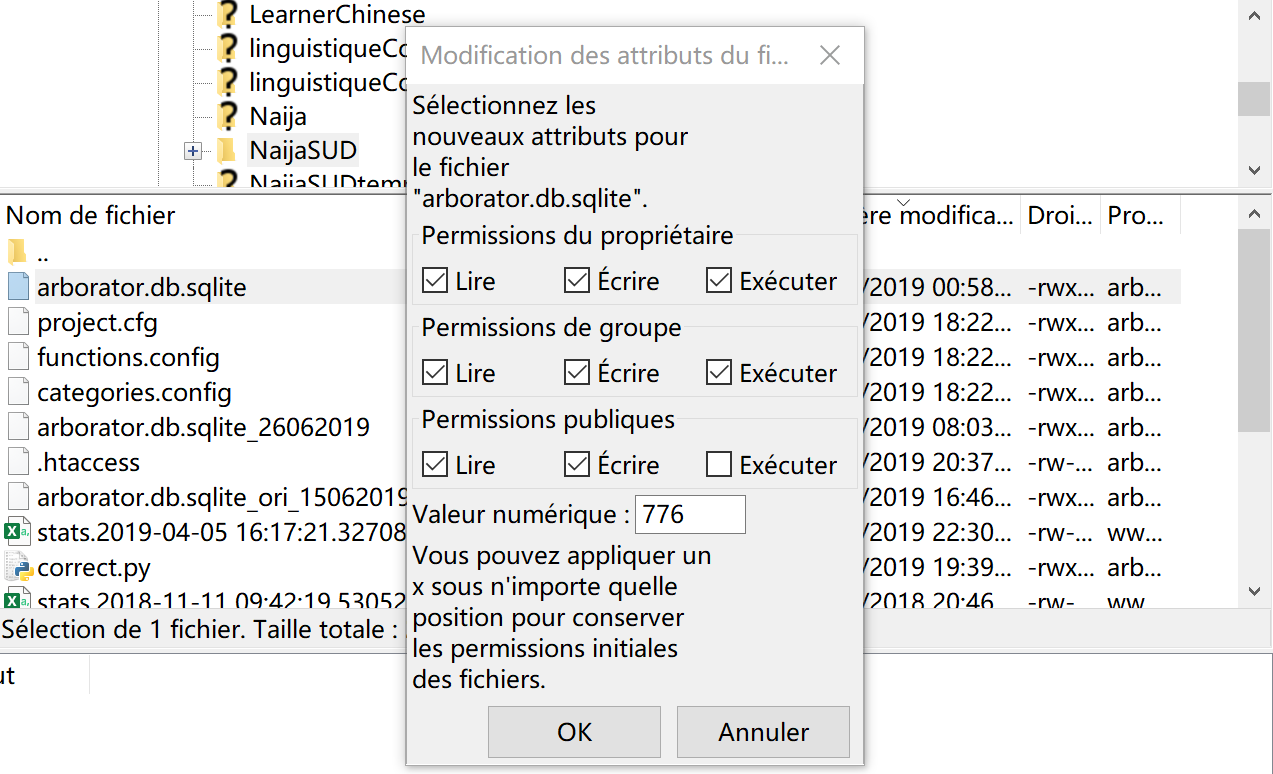
* + 情况2：没有startali和endali，词的时间起始结束标记在misc值，需取相应数值添加startali和endali两行，删除xpos和misc。

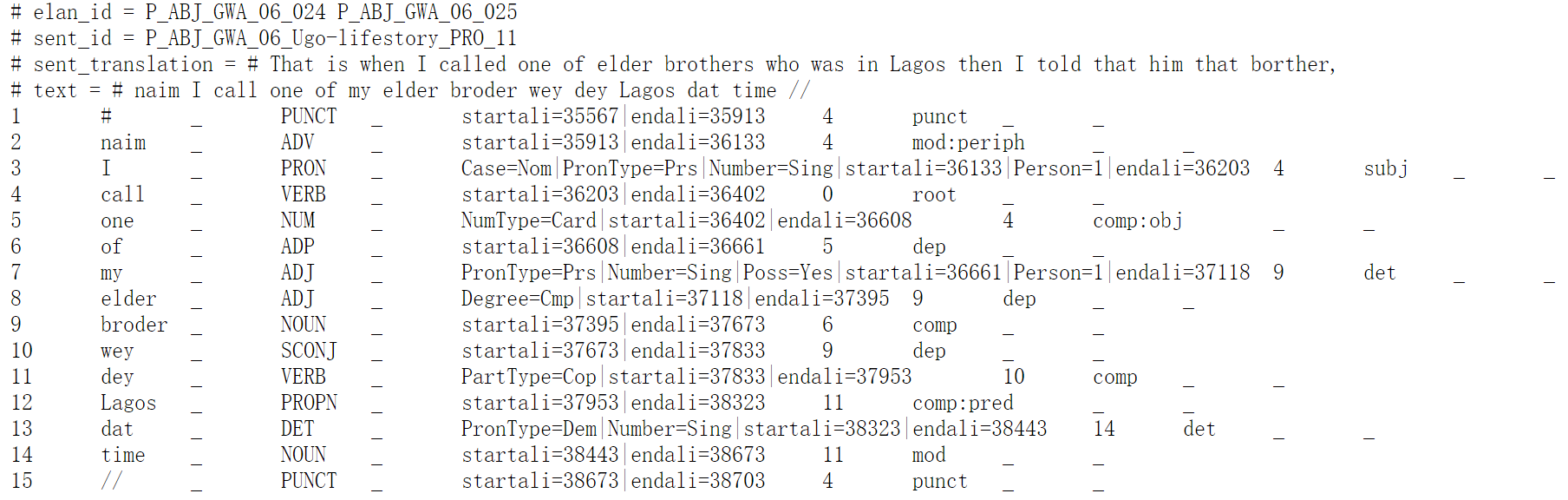
• **代码文件**：

* + 添加startali和endali：*add\_ali.py*
  + 删除xpos和misc*：delete\_misc\_xpos.py*

1. **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 ficher …



* + 再导出的conll文件的呈现效果：



**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。

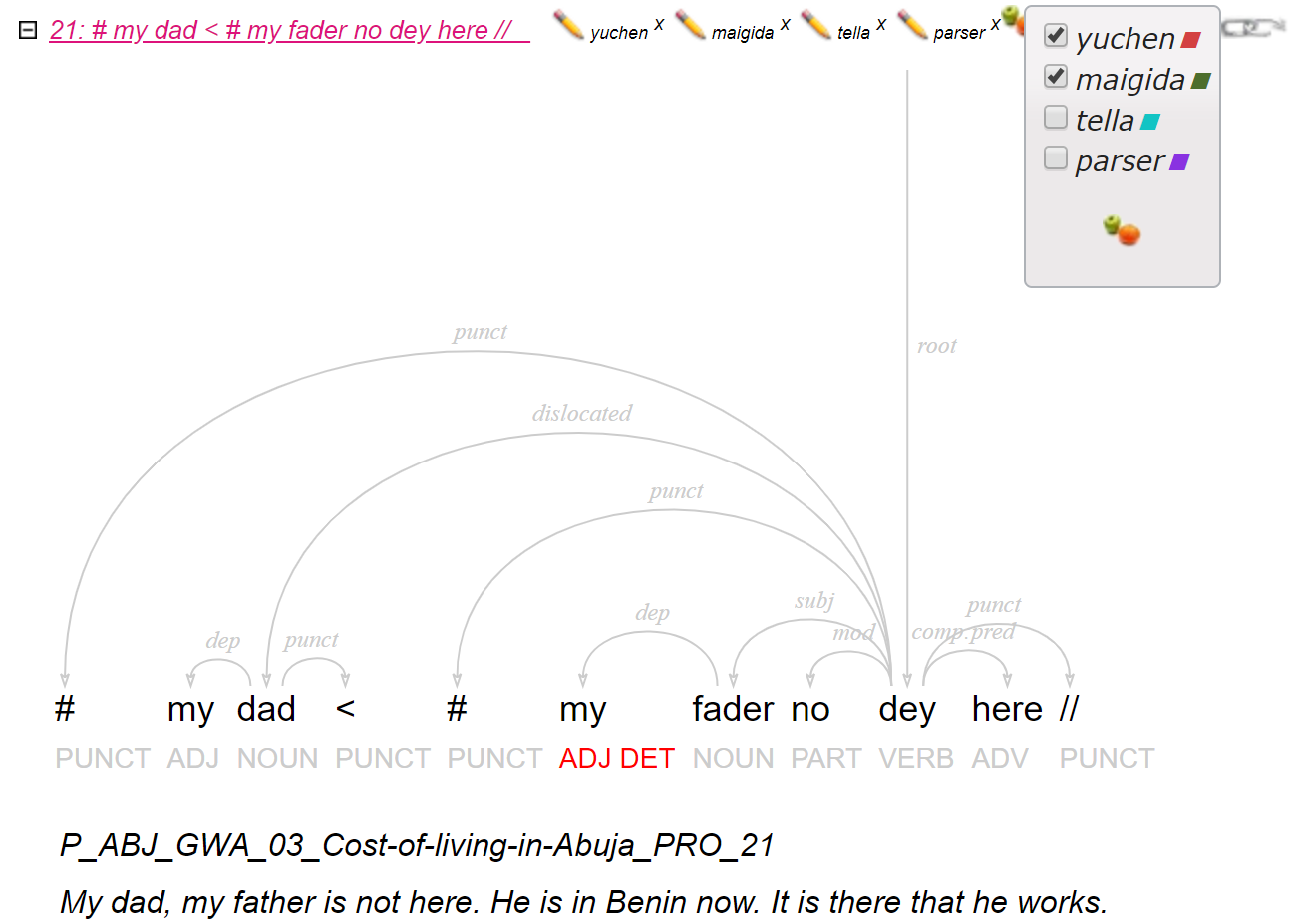
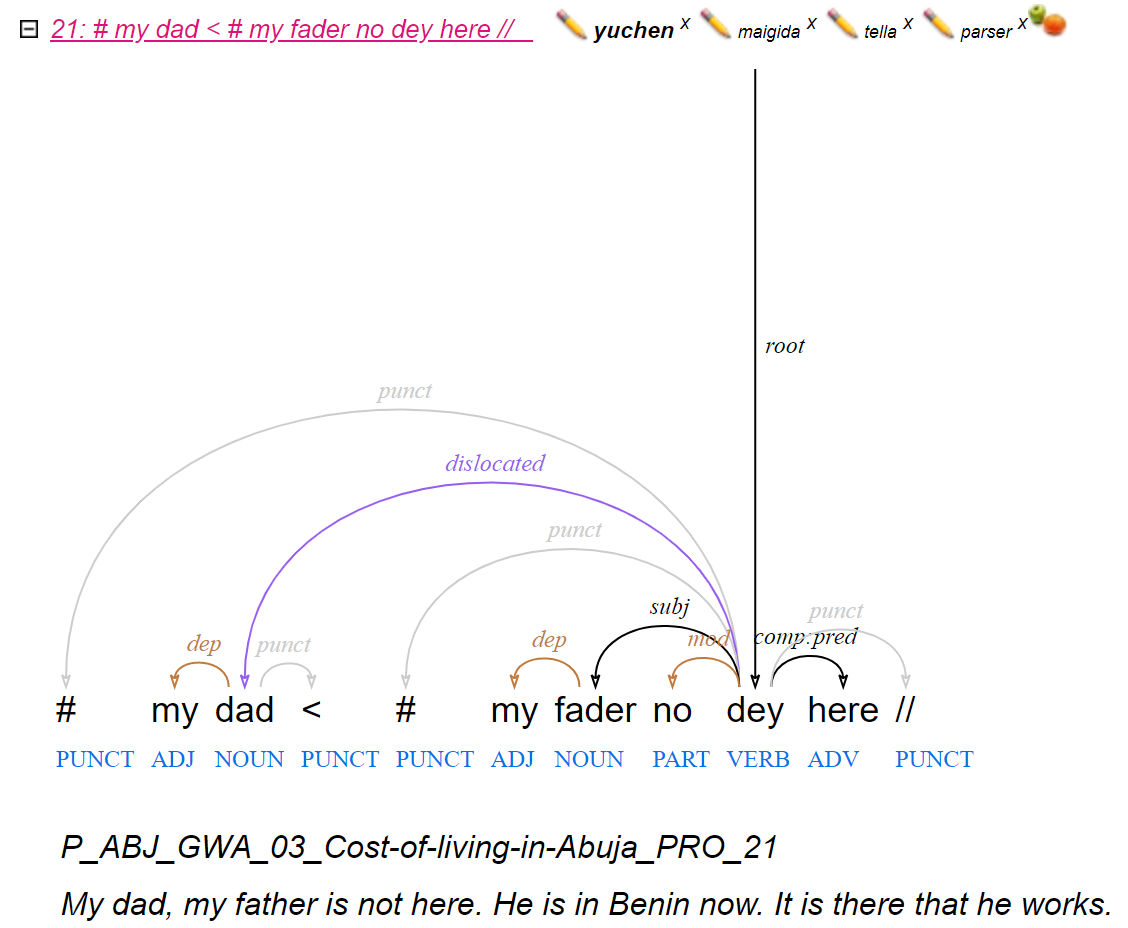
- 如果phrase不存在contenu中，说明该句已做修改，将写入数据库。

* + 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：**

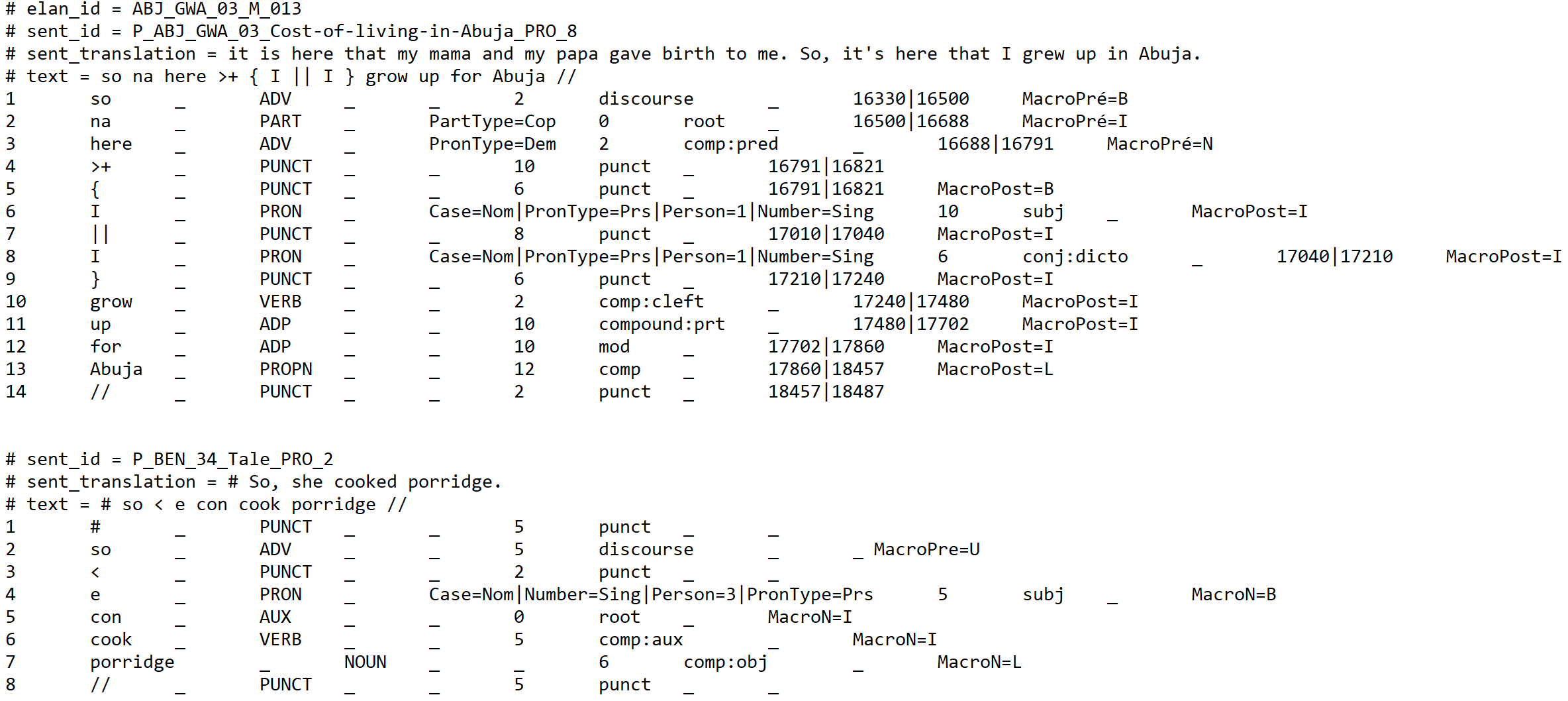
* + SUD和UD的trait的区别？ 无区别
  + trait在arbre中的体现？ 用于transformation de SUD au UD : mod => a:mod

**未进行任务：**

1. **Identifier les noyaux**

• **格式：**prénoyau < prénoyau < noyau > postnoyau //

• **例子：**



1. **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)