Fiche intervention pour Galla Septembre 2023
Pour prise en compte et estimation pour le CAP
YS/ASTN/LLL/CNRS
Septembre 2023

Expression de besoins : 19 septembre 2023

La demande formulée par Galla consiste à faire le point et à vérifier dans les transcriptions qu'elle a produites sur la langue ngubu au fil des années qu'il n'y a pas eu de dérive dans les descriptions au fur et à mesure de sa compréhension de la langue. Et si jamais, il y a eu des variations au cours de temps de pouvoir les estimer et ensuite de pouvoir harmoniser l'ensemble afin d'avoir une traduction cohérente.

Les données :

Une centaine de fichiers (113) eaf produits par ElanCorpA et un fichier du dernier lexique à partir de son poste transférés ensuite EXTRABOX. Problème de récupération un peu pénible car par possibilité de sélectionner l'ensemble des fichiers d'un coup. Régler par la suite par un envoi d'un fichier archive.

Solution et démarche proposée à l'intéressée à chaud : 19/09/2023

Vérifier la faisabilité de l'extraction et de l'harmonisation en examinant les fichiers eaf produits par ELAN. A priori ce sont des structures XML. Ensuite avant d'harmoniser, faire le point sur le contenu des fichiers en essayant de faire l'inventaire par motif/forme des descriptions associées et de permettre à Galla d'en faire l'analyse. A définir de quelle façon. De pouvoir éventuellement associer à cet inventaire le dernier lexique utilisé.

Dans un deuxième temps :
Il faudra peut-être voir comment importer et récupérer l'ensemble de ce corpus complet ainsi que son expertise pour les mettre en valeur dans un contexte LLL.

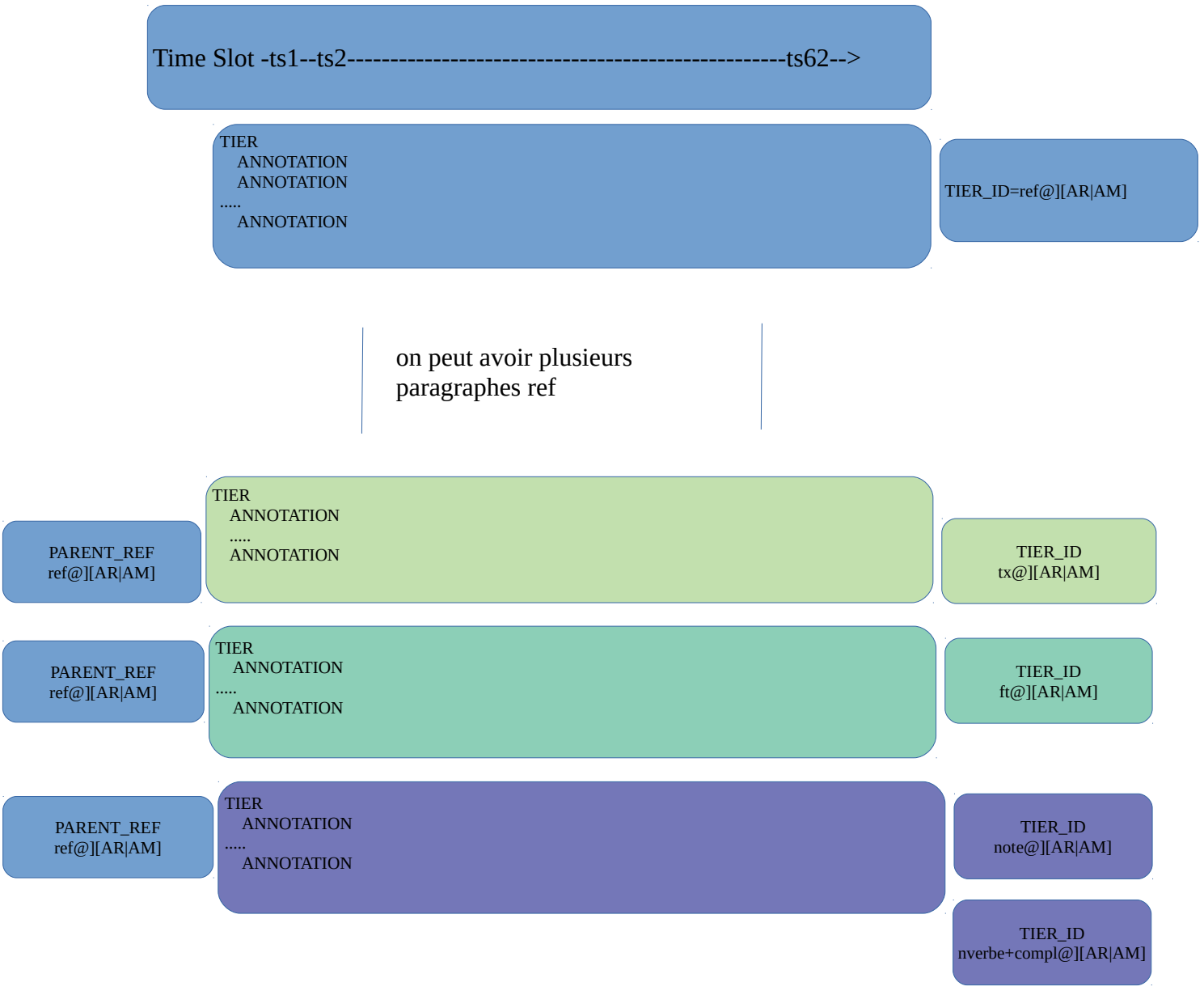
Début de l'intervention : mercredi 20/09/2023

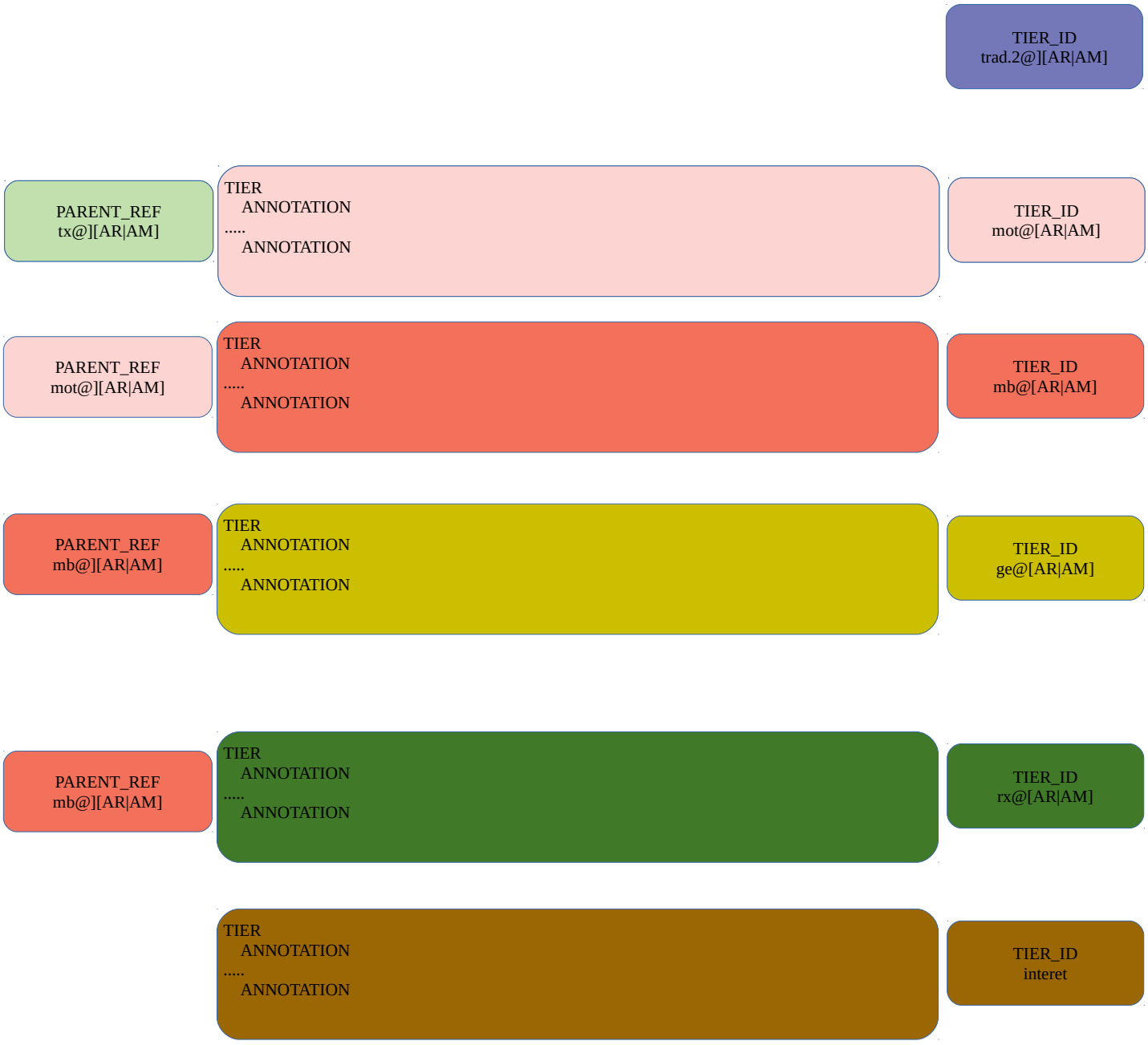
Analyse des fichiers produits dans ELAN par Galla pour connaître la structure interne et de voir comment procéder à l'extraction des données.

Les fichiers ELAN sont structurés à l'aide de TAG de type TIER et de tags de type ANNOTATION de la façon suivante :

Une première partie qui donne les Time Slot de l'enregistrement avec une valeur de TEMPS.
Ensuite arrive les paragraphes pour les annotations avec une description des dépendances entre les différentes valeurs contenues. Chaque paragraphe est associé à une référence à l'aide de l'attribut TIER_ID.

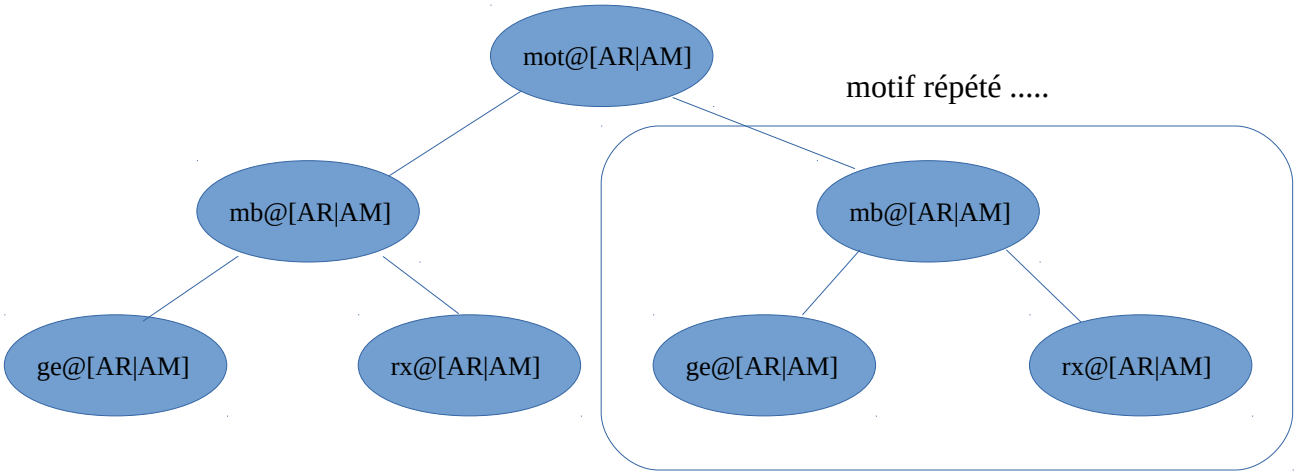
Représentation de la structure et des dépendances associées ...





Que faut-il extraire et comment reconstruire les éléments en respectant la hiérarchie et les dépendances ?

On peut commencer à extraire les éléments à partir du mot et toutes les informations qui décrivent et caractérisent ces mots (mb, ge et rx).
Ce qui nous donne le schéma suivant :



il te faut un contexte visuel pour examiner les différentes formes et les valider ou les modifier

LEXIQUE ACTUEL

mb --> mot1, mot2	mb --> mot1, mot2
--> ge1	--> ge1
--> rx1	--> rx1
--> mot3, mot4	--> mot3, mot4
--> ge2	--> ge2
--> rx2	--> rx2

se poser la question :
c'est Ok ou on change

on produit un script (python) qui effectue les changements directement dans les fichiers eaf pour harmoniser l'ensemble ?????

Exemple de mise en œuvre sur un exemple : tó

Fichier	Phrase/Expression	mot	mb	ge	rx
AM_presentation.eaf	a31 : àlà_ à **tó** nə́ pākēdə́ mà gɔ̄ŋgbúgù jú nú gbɛ́	a102 : tó	a634 : tó	a637 : dire\H	a640 :verbe
		a102 : tó	a635 : ʁə́	a638 : 3INAN\ CMPL	a641 : pronom
AM_presentation.eaf	a58 : bɔ̀ mɛ̀rsí áɲē ɲē **tɔ́** nʤə̄pō ndə́ kə́ zà ʁə̄à ŋgbúgù làfó	a174 : tɔ́	a454 : tó	a508 : dire\H	a562 : verbe
AR_brochettes.eaf	a80 : kə́ pù sà ē **tó** wò	a345 :tó	a983 :tə́	a989 : nom	a986 :corps
AR_brochettes.eaf	a89 : ē pá \ ē **tó** wò nī̄	a396 :tó	a439 :tə́	a1139 : nom	a1145 :corps
AR_elicit_11.eaf	a45 : à **tó** pā mà gɔ̄ŋgbúgù jú nə́ gbɛ́	a932 : tó			
AR_elicit_11.eaf	a46 : à tó nə́ pākēdə́ mà ŋgbúgù jú nə́ gbɛ́	a949 : tó			
AR_SickGirl.eaf	a65 : àndʒē **tó** sə̄bèʃē nə̄pādə́ àɲē\ ʧè wùtù	a256 : tó	a925 : tó	a941 : originer\H	a957 :verbe
			a926 : ʁə́	a942 : 3INAN\ CMPL	a958 : pronom
AR_SickGirl.eaf	a70 : áɲē bèʃē nə̄ **tó** sǎ sə̄ àndʒé pākēdə́	a272 : tó	a973 : tó	a982 : dire\H	a991 :verbe
AR_SickGirl.eaf	a95 : ʧè **tó** sə̄ àndʒé pàkēdə́	a366 : tó	a1269 :tó	a1276 : dire\H	a1283 : verbe
			a1270 : ʁə́	a1277 : 3INAN\ CMPL	a1284 : pronom
AR_SweekCocoon1.eaf	a36 bɔ̀ / pō nə́ ŋgá kwūlà kā / nə́ / ʧè ɲí / nə́ kə́ mbò té / **tó** kwómbōlē	a194 : tó	a428 : tə́	a454 : nom	a441 : corps

Analyse et interprétation des fichiers eaf fournis :

il y a une relation hiérarchique entre mot -> tx
il y a une relation hiérarchique entre mb -> mot
il y a une relation hiérarchique entre ge -> mb
il y a une relation hiérarchique entre rx -> mb

} constitue le gloss

plusieurs mb peuvent pointer sur un même mot

Dans certains fichiers, il peut y avoir des informations non complétées et manquantes

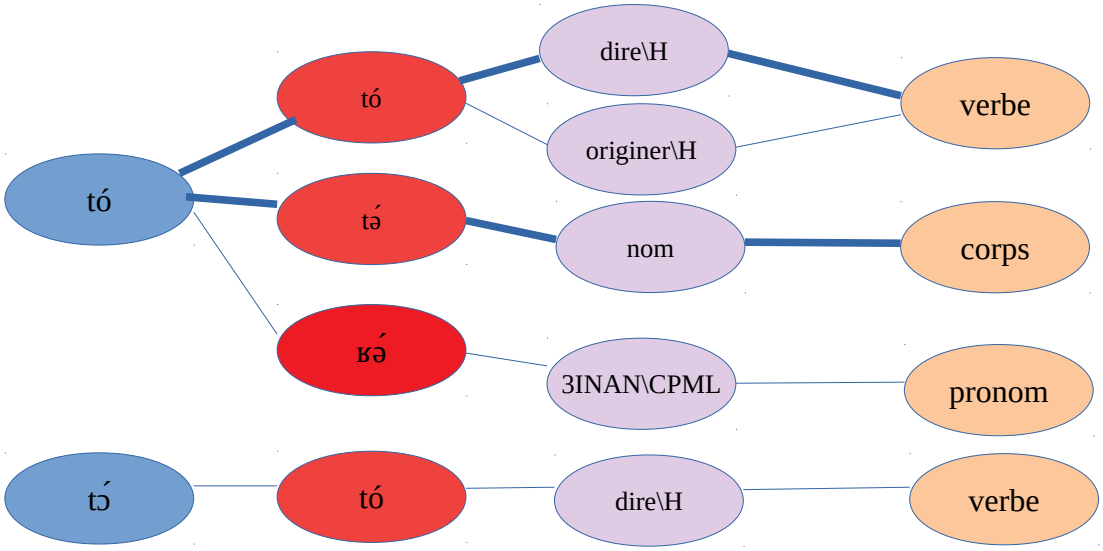
pas de mb --> mot ==> erreur à indiquer dans le fichier de log

Dans certains fichiers eaf, il n'y a pas d'annotation

Interprétations possibles :

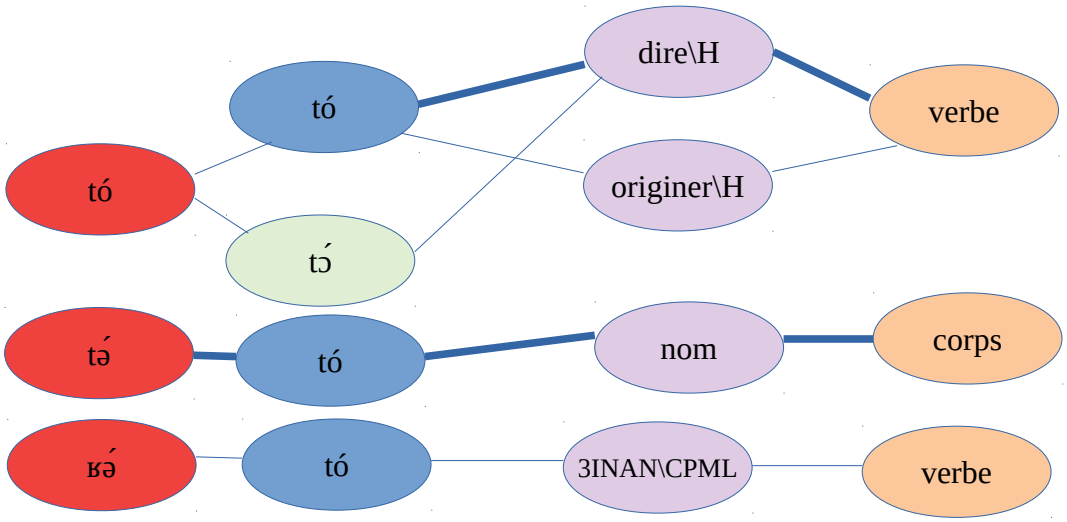
Première solution déduite : départ sur mot

mot		mb		nb		ge rx
tó	===>	tó	==>	nb 3	==>	dire\H -- verbe
tó	===>	tə́	==>	nb 3	==>	nom -- corps
tó	===>	tó	==>	nb 1	==>	originer\H -- verbe
tó	===>	ʁə́	==>	nb 3	==>	3INAN\CPML -- pronom
tɔ́	===>	tó	==>	nb 1	==>	dire\H -- verbe



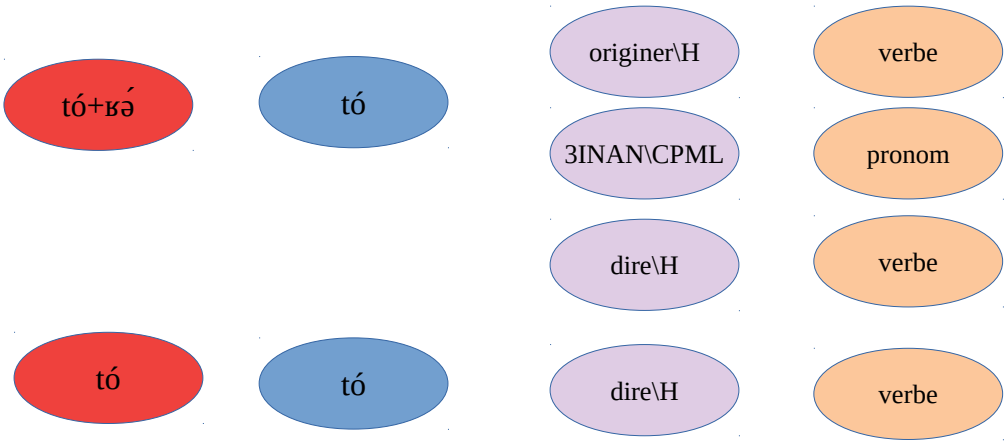
Deuxième solution déduite : départ sur mb

mb	mot	ge	rx

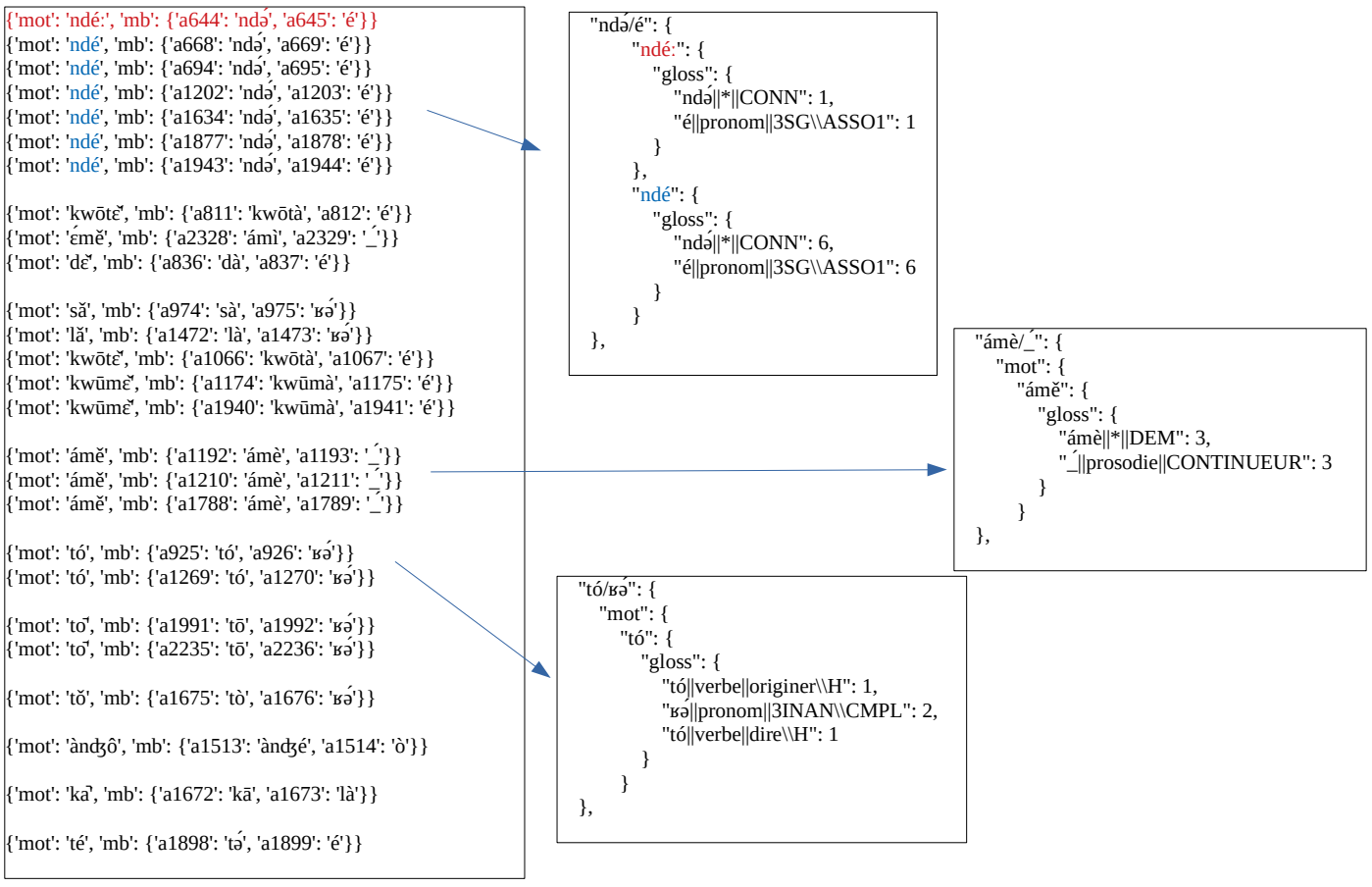


Suite aux derniers commentaires de Galla concernant les MB consécutifs pour un même mot

AR_SickGirl.eaf	a65 : àndʒē **tó** sə̄ bèʃē nə̄ pādə́ àɲē\ ʧè wùtù	a256 : tó	a925 : tó	a941 : originer\H	a957 :verbe
AR_SickGirl.eaf			a926 : ʁə́	a942 : 3INAN\ CMPL	a958 : pronom
AR_SickGirl.eaf	a70 : áɲē bèʃē nə̄ **tó** sǎ sə̄ àndʒé pākēdə́	a272 : tó	a973 : tó	a982 : dire\H	a991 :verbe
AR_SickGirl.eaf	a95 : ʧè **tó** sə̄ àndʒé pàkēdə́	a366 : tó	a1269 :tó	a1276 : dire\H	a1283 : verbe
AR_SickGirl.eaf			a1270 : ʁə́	a1277 : 3INAN\ CMPL	a1284 : pronom



Exemple d'extraction sur le fichier uniquement les mots qui ont plusieurs mb associés à partir du fichier AR_SickGirl.eaf



Description de la procédure d'harmonisation : processus global

Les différentes phases de développement

Phase 1 :

Extraire le lexique actuel pour en faire le constat (situation actuelle) à partir des fichiers eaf

Phase 2 :

Extraire le lexique cible à partir du fichier eafl

Pour être sûr de la cible

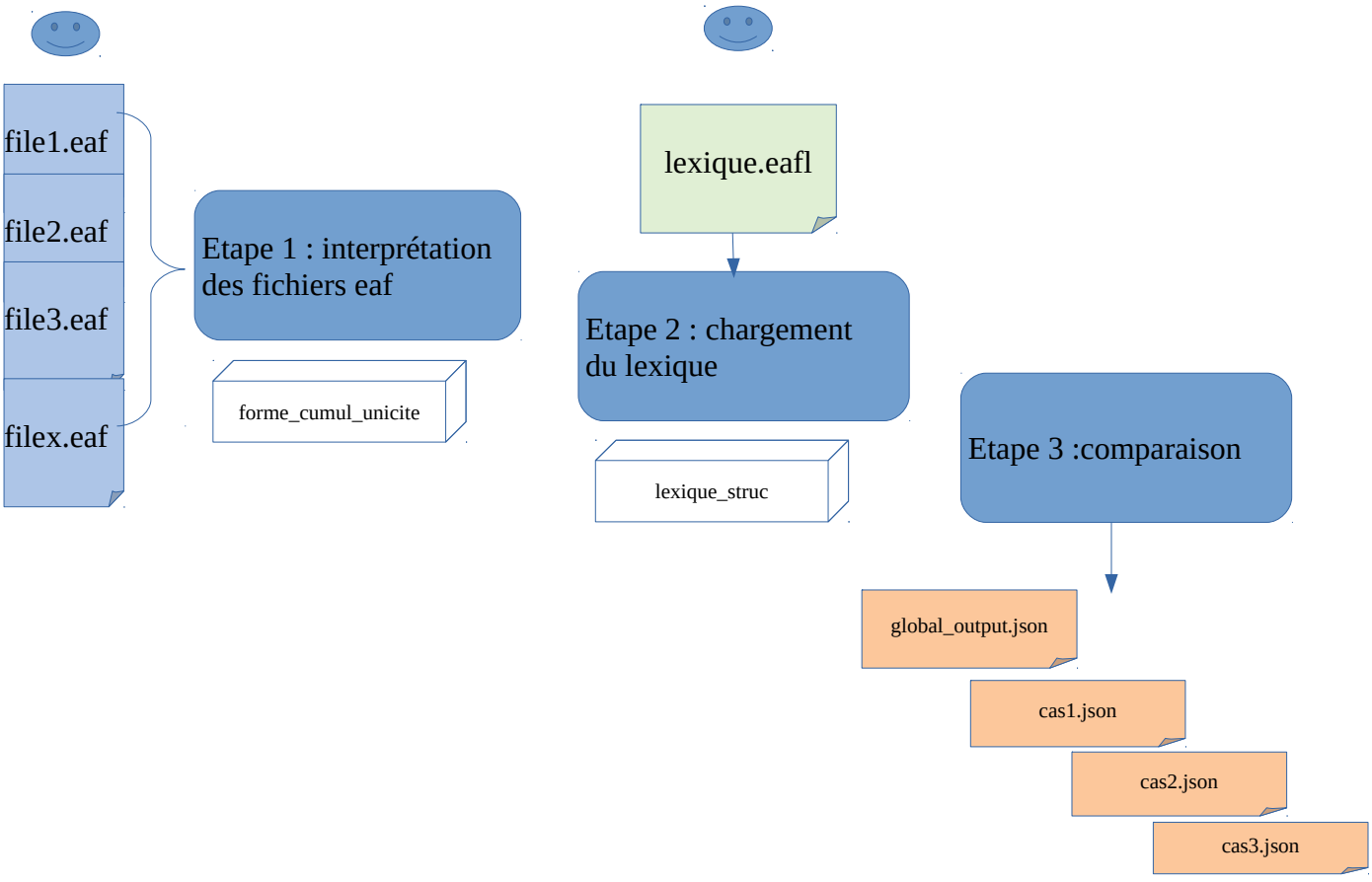
Phase 3 :

Implémenter les correspondances entre les deux et générer les scripts nécessaires pour appliquer les modifs *

Phase 4 : contrôle

Reprendre la phase 1 et comparer avec le lexique phase 2

le processus doit être itératif afin de permettre d'ajuster au fil de l'eau en fonction des modifications à apporter au niveau des fichiers eaf et au niveau du lexique.



Il faut fournir pour chaque forme mb, les descriptions associées en un seul exemplaire pour améliorer la lecture et l'analyse

Dans ce programme on procède en plusieurs étapes :
Phase 1 :

Extraction des paragraphes XML représentants les structures à analyser

Paragraphe_tx
Paragraphe_mot
Paragraphe_mb
Paragraphe_ge
Paragraphe_rx

Appel de la fonction remplir avec en argument le nom du tag à sélectionner et à charger. retourne une chaine de caractère représentant la structure XML

Si il manque un des paragraphes, le script ressort une erreur et arrête l'interprétation du fichier
Si Ok
Exploite les différents paragraphes

Paragraphe tx

tx[ref_ID]= { "ref_annotation":anno, "value" : valeur }

```
<ANNOTATION>
  <REF_ANNOTATION ANNOTATION_ID="a2" ANNOTATION_REF="a1">
    <ANNOTATION_VALUE>ŋgá ɛʃē nà bèʃē ndé: kā át</ANNOTATION_VALUE>
  </REF_ANNOTATION>
</ANNOTATION>
```

tx["a1"]= { "ref_annotation":"a2", "value" : "ŋgá ɛʃē nà bèʃē ndé: kā á" }

Paragraphe mot :

mots[ref_ID]= { "ref_":tx, "mot" : valeur, mb:{} }

```
<ANNOTATION>
  <REF_ANNOTATION ANNOTATION_ID="a163" ANNOTATION_REF="a2">
    <ANNOTATION_VALUE>ŋgá</ANNOTATION_VALUE>
  </REF_ANNOTATION>
</ANNOTATION>
```

mots["a163"]= { "ref_":"a2", "mot" : "ŋga", mb:{} }
On prépare pour chaque mot la structure d'accueil qui va être compléter par la suite à la lecture des mb pour conserver le lien mot --> mb

Paragraphe mb :

mb[ref_ID]={"ref_mot" : ref_mot, "mb_value":valeur}
mots["ref_mot"]["mb"]["ref_mb"]=valeur

```
<ANNOTATION>
  <REF_ANNOTATION ANNOTATION_ID="a634" ANNOTATION_REF="a163">
    <ANNOTATION_VALUE>ŋgá</ANNOTATION_VALUE>
  </REF_ANNOTATION>
</ANNOTATION>
```

mb["a634"]={"ref_mot" : "a163", "mb_value":"ŋgá"}
mots["a163"]["mb"]["a_634"]=valeur

Paragraphe ge : on complète la structure mb

mb[ref_mb]["ref_ge"]=ref_ge
mb[ref_mb]["ge_value"]=valeur

```
<ANNOTATION>
  <REF_ANNOTATION ANNOTATION_ID="a637" ANNOTATION_REF="a634">
    <ANNOTATION_VALUE>INDEF</ANNOTATION_VALUE>
  </REF_ANNOTATION>
</ANNOTATION>
```

mb["a634"]["ref_ge"]="a637"
mb["a634"]["ge_value"]="INDEF"

Paragraphe rx : on complète la structure mb
mb[ref_mb]["ref_rx"]=ref_rx
mb[ref_mb]["rx_value"]=valeur

```
<ANNOTATION>
  <REF_ANNOTATION ANNOTATION_ID="a640" ANNOTATION_REF="a634">
    <ANNOTATION_VALUE>*</ANNOTATION_VALUE>
  </REF_ANNOTATION>
</ANNOTATION>
```

mb["a634"]["ref_rx"]="a634"
mb["a634"]["rx_value"]="*"

Affichage des grandeurs de chaque éléments

Analyse et construction des clés en fonction de la relation mb --> mot
Si plusieurs mb pointent vers le même mot, il faut composer une nouvelle clé qui va être la concaténation des mb
Sinon on stocke le mb dans sa forme initial
dans tous les cas on rajoute à la structure mots un attribut "cle" et on rajoute dans un attribut "detail" le contenu du ou des mb associés. Ainsi la structure mot devient complète/

Pour une référence de mot a616 avec plusieurs mb:

```
{'mot': 'tɔ̄', 'ref':'a151', 'mb': {'a2235': 'tō', 'a2236': 'ʁə́'}, 'cle': 'tō/ʁə́',
  'detail': [
    {'ref_mot': 'a616', 'mb_value': 'tō', 'ref_ge': 'a2251', 'ge_value': 'dire\\M', 'ref_rx': 'a2267', 'rx_value': 'verbe'},
    {'ref_mot': 'a616', 'mb_value': 'ʁə́', 'ref_ge': 'a2252', 'ge_value': '3INAN\\CMPL', 'ref_rx': 'a2268', rx_value': 'pronom'}
  ]
}
```

Pour une référence de mot a624 avec un mb:

```
{'mot': 'kə́', 'ref':'a152', 'mb': {'a2244': 'kə́'}, 'cle': 'kə́'
        'detail': [{'ref_mot': 'a624', 'mb_value': 'kə́', 'ref_ge': 'a2260', 'ge_value': 'TAM', 'ref_rx': 'a2276', 'rx_value': '*'}],
}
```

Déduction du lexique à partir de mots et de forme_cumul_unicite
On liste chaque mot de la structure mots pour le fichier en cours
On vérifie que le mot est bien associé à au moins un mb
on récupère la clé
on vérifie si elle est présente dans forme_cumul_unicite
sinon on créer la structure d'accueil des données
on complète la structure forme_cumul_unicite

Représentation de la structure forme_cumul_unicite

```
forme_cumul_unicite[Cle]
                    ["mot"]
                            [mot1]
                                    ["expressions"]
                                    ["gloss"]
                                            [cle_detail1]=nb1
                                            [cle_detail2]=nb2
                            [mot2]
                                    ["expressions"]
                                    ["gloss"]
                                            [cle_detail1]=nb1
                                            [cle_detail2]=nb2
```

Exemple de contenu produit

```
# exemple pour un mb et un mot
  "ŋwù": {
    "mot": {
      "ŋwù": {
        "expression": [
          "áɲē ɓá lə́ ndà::: / ɓá lə́ ndà jò ɓáɛ̀ àʁə́má nə́ bèʃē nə̄ / àndʒē nə́ nə́ kə́ ŋwù ʧē||./AR_SickGirl.eaf",
          "àndʒē ŋwù sà kēkē nə́ àɲé||./AR_SickGirl.eaf",
          "ɓá ópìtâl ámě ʔàʁə́má_ \\ ʔàʁə́má ndé ámě àndʒē nə̄ nə́ kə́ ŋwù ʧē ||./AR_SickGirl.eaf",
          "ʧèʧē ò / ʧè ŋwù ʧú ndʒōní áɲē kə́ nə̀ tá||./AR_SickGirl.eaf",
          "àʁə́má ndé ámɛ̀ ná ə̀kɔ̂ nə́ kə́ ŋwù ʧē / nə́ kə́ pàdə́ àndʒē wùtə̀||./AR_SickGirl.eaf",
          "àʁə́má ndə́ ámɛ / ndə̀_ ndə́ bèʃē nə̄ / àndʒē ná / nə́ kə́ ŋwù ʧē nə́ kə́ pà àndʒē wùtə̀||./AR_SickGirl.eaf"
        ],
        "gloss": {
          "ŋwù||verbe||voir\\B": 6
        }
      }
    }
  },
# exemple pour un mb et plusieurs mots
  "pākēdə́": {
    "mot": {
      "pādə́": {
        "expression": [
          "àndʒē tó sə̄ bèʃē nə̄ pādə́ àɲē\\ ʧè wùtù||./AR_SickGirl.eaf",
          "àɲī nə̄ kə̄ kə́ tŏ sə̄ ʧē pādə́||./AR_SickGirl.eaf",
          "ʧè tŏ sə̄ àndʒé pādə / àɲé wùtə̀ bá tè à||./AR_SickGirl.eaf"
        ],
        "gloss": {
          "pākēdə́||*||introducteur DR": 3
        }
      },
      "pākēdə́": {
        "expression": [
          "áɲē bèʃē nə̄ tó să sə̄ àndʒé pākēdə́||./AR_SickGirl.eaf"
        ],
        "gloss": {
          "pākēdə́||*||introducteur DR": 1
        }
      }
    }
  },
```

Phase 2 :

On peut aborder la phase 2, qui va consister à charger le lexique à partir du fichier eafl

Rien d'exceptionnel dans cette phase, on lit le fichier eafl et on restructure l'ensemble XML en un dictionnaire au sens Python. Cette lecture va permettre de stocker le résultat dans le dictionnaire lexique_struc sous la forme suivante : la clé de la structure est sur la forme des expressions mb

```
lexique_struc[lexeme]
        ["gloss"]
```

["segments"]
["altForms"]

On ne profite pour cumuler les différents formes dans une même rubrique ce qui va rendre plus simple l'exploitation des données lors de l'étape suivante.

```
lexique_struc['lə̀mùndʒú']
{'altForms': [], 'gloss': [{'tierX': 'Nom de personne', 'text': 'Lemoundjou'}]}

lexique_struc['pɛ̀rsílə̀']
{'altForms': ['pèʁsélə̀', 'pɛ̀ʁsílə̀', 'pèrsílə̀', 'pɛ̀rsîl'], 'gloss': [{'tierX': 'nom (français)', 'text': 'persil'}]}

lexique_struc['bə̀ɛ̀']
{'altForms': [], 'gloss': [], 'segments': ['ábò', 'à']}
```

```
<lexicalEntry id="2315" dt="08/Mar/2023">
 <Lexeme typ="lem">lə̀mùndʒú</Lexeme>
 <form />
 <sense>
  <Gloss lang="en" tierX="Nom de personne">Lemoundjou</Gloss>
 </sense>
</lexicalEntry>
```

```
<lexicalEntry id="2175" dt="04/Jan/2023">
 <Lexeme typ="lem">pɛ̀rsílə̀</Lexeme>
 <form>
  <altForm>
   <WForm>pèʁsélə̀</WForm>
  </altForm>
  <altForm>
   <WForm>pɛ̀ʁsílə̀</WForm>
  </altForm>
  <altForm>
   <WForm>pèrsílə̀</WForm>
  </altForm>
  <altForm>
   <WForm>pɛ̀rsîl</WForm>
  </altForm>
 </form>
 <sense>
  <Gloss lang="en" tierX="nom (français)">persil</Gloss>
 </sense>
</lexicalEntry>
```

```
<lexicalEntry id="2187" dt="09/Jan/2023">
 <Lexeme typ="wf">bə̀ɛ̀</Lexeme>
 <form />
 <sense />
 <morph>
  <Segment ref="1156">ábò</Segment>
  <Segment ref="1157">à</Segment>
 </morph>
</lexicalEntry>
```

Phase 3 :

On peut démarrer la phase 3 qui va consister à comparer les deux structures de données ainsi constituées lexique_struc avec forme_cumul_unicite et de produire des fichiers résultats de cette comparaison.
Ensuite il faudra déterminer comment répercuter les modifications dans les fichiers eaf.

Suite à l'analyse des correspondances il s'avère qu'il y a trois cas à distinguer dans la comparaison :
On démarre la vérification à partir du lexique recomposé :

Pour chaque clé :

Cas 1 : clé composite

il faut pour chaque mot attaché à cette clé que l'on retrouve dans le lexique des segments qui doivent recomposer cette clé

Erreurs possibles :

CAS1_error_1 : le mot n'est pas présent dans le lexique

CAS1_error_2 : pas de segment définis dans le lexique pour le mot
CAS1_error_3 : les segments ne recomposent pas la clé

Cas 2 : clé simple un seul mot

CAS2_error_1 : le mot n'est pas présent dans le lexique
CAS2_error_2 : pas de gloss définis
CAS2_error_3 : Comparaison des gloss non conforme

Cas 3 : clé simple plusieurs mots

CAS3_error_1 : le mot n'est pas présent dans le lexique
CAS3_error_2 : pas de gloss définis
CAS3_error_3 : comparaison des mots de la liste non conforme
CAS1_error_4 : Comparaison des gloss non conforme

Analyse du lexique de Référence issu du fichier eafl :

Dans le lexique plusieurs formes voisines sont présentes et sont sans doute issues d'une variation au cours du temps de la formulation. Elles n'ont pas de raison a priori d'exister. ce qui va nécessiter un remaniement de ce type de données ;

Première étape de cette exploitation c'est l'insertion du lexique dans une base de données afin d'en extraire certaines représentation afin d'analyse son contenu et de repèrer d'éventuelles anomalie.

Récupération du fichier eafl et insertion dans la base de données MongoDB, importation des tag xml vers Mongodb

```
<lexicalEntry id="125" dt="27/Apr/2020">
  <Lexeme typ="lem">éŋgà</Lexeme>
  <form>
    <altForm>
      <WForm>ʔéngà</WForm>
    </altForm>
    <altForm>
      <WForm>jíŋgà</WForm>
    </altForm>
    <altForm>
      <WForm>íŋgà</WForm>
    </altForm>
  </form>
  <sense>
    <Gloss lang="en" der="H" tierX="verbe (sango)">connaître</Gloss>
    <Gloss lang="en" der="H" tierX="verbe (sango)">considérer, respecter</Gloss>
  </sense>
</lexicalEntry>
```

```
{
   "_id" : ObjectId("6512c56941efa7e2b8bda86a"),
   "id" : "125",
   "dt" : "27-04-2020",
   "Lexeme" : {"typ" : "lem","#text" : {"data" : "éŋgà"}},
   "form" : {
      "altForm" : [
         {"WForm" : {"#text" : {"data" : "ʔéngà"}}},
         {"WForm" : {"#text" : {"data" : "jíŋgà"}}},
         {"WForm" : {"#text" : {"data" : "íŋgà"}}}
      ]
   },
   "sense" : {
      "Gloss" : [
         {
            "lang" : "en","der" : "H","tierX" : "verbe (sango)",
            "#text" : {"data" : "connaître"}
         },
         {
            "lang" : "en","der" : "H","tierX" : "verbe (sango)",
            "#text" : {"data" : "considérer, respecter"}
         }
      ]
   }
}
```

```
<lexicalEntry id="935" dt="09/Apr/2021">
  <Lexeme typ="lem">_́</Lexeme>
  <form />
  <sense>
    <Gloss lang="en" tierX="prosodie">CONTINUEUR</Gloss>
  </sense>
</lexicalEntry>
```

```
{
            "_id": "650db24b5b654d572ebcd7c1",
            "id": "935",
            "dt": "09/Apr/2021",
            "Lexeme": {"typ": "lem","#text": {"data": "_́"}},
            "form": {},
            "sense": {
                  "Gloss": {
                                    "lang": "en","tierX": "prosodie",
                                    "#text": {"data": "CONTINUEUR"}
                              }
            }
}
```

Plusieurs problèmes dans ce transfert :
Pb:1 :
Les structures internes pour un même attribut peuvent varier en fonction du contenu du XML. Exemple pour Gloss qui peut se trouver dans le base Mongodb sous deux formes

Structure
Tableau --> structure

Si il y a plusieurs occurrences de gloss dans la balise Sense. Donc à la formulation des requêtes pour les extraction.

pb2 : formalisme de la date

Attention changement du format date car le format <lexicalEntry id="783" dt="19/Jan/2021"> pose des problèmes par la suite dans mongodb pour trier sur les dates, car ne le reconnait pas comme une date. Donc on va le convertir sous la forme DD-MM-YYYY.

Explication du script d'importation vers mongodb de la structure eafl

```
# ---------------------------------------------
# Connexion a la base de donnees
# ---------------------------------------------
myclient =
pymongo.MongoClient("mongodb://localhost:27087/",username="",password="",authSource="bddictionnairique",authMechanism='SCRAM-SHA-1')
mydb = myclient["galla"]
mycol = mydb["lexique"]

# ---------------------------------------------
# lecture du fichier lexique
# ---------------------------------------------
_file="lexique_ngbg.eafl"

# fonction de substitution du mois en lettre par le numérique correspond
mois={"Jan":"01","Feb":"02","Mar":"03","Apr":"04","May":"05","Jun":"06","Jul":"07","Aug":"08","Sep":"09","Oct":"10","Nov":"11","Dec":"12"}
p=re.compile('[0-9]{2}/[A-Za-z]{3}/[0-9]{4}')  # format recherché 19/Jan/2021

# ---------------------------------------------------------------------------------------------
# Fonction de lecture et de préparation de la structure json pour le stockage dans mongodb
# attention la fonction est récursive pour explorer l'arbre XML complet
# ---------------------------------------------------------------------------------------------
def parse_element(element):
    global p
    dict_data = dict()
    if element.nodeType == element.TEXT_NODE:
        dict_data['data'] = element.data
    if element.nodeType not in [element.TEXT_NODE, element.DOCUMENT_NODE,element.DOCUMENT_TYPE_NODE]:
        for item in element.attributes.items():
            # pour adapter le format date
            nouv=item[1]
            m=p.match(nouv)
            if m:
                tab=nouv.split("/")
                nouv=tab[0]+"-"+mois[tab[1]]+"-"+tab[2]
            dict_data[item[0]] = nouv
    if element.nodeType not in [element.TEXT_NODE, element.DOCUMENT_TYPE_NODE]:
        for child in element.childNodes:
            child_name, child_dict = parse_element(child)
            if child_name in dict_data:
                try:
                    if "data" not in child_dict:
                        dict_data[child_name].append(child_dict)
                except AttributeError:
                    dict_data[child_name] = [dict_data[child_name], child_dict]
            else:
                if "data" in child_dict:
                    if "\n" not in child_dict["data"]:
                        dict_data[child_name] = child_dict
                else:
                    dict_data[child_name] = child_dict
    return element.nodeName, dict_data
# chargement du fichier dans une structure mémoire à l'aide de #from xml.dom import minidom
dom=minidom.parse(_file)
# appel de la fonction de conversion en dict de cette structure
toto,structure_json=parse_element(dom)
# exploitation de la structure obtenue pour insertion dans la base MongoDb
for entry in structure_json["lexicon"]["lexicalEntry"]:
    x = mycol.insert_one(entry)
```

Analyse du contenu à partir de quelques requêtes sous MongoDB/

db.lexique.aggregate([{"$addFields":{"created_at":{"$dateFromString":
{"dateString":"$dt","format":"%d-%m-%Y"}}}},{$group:{_id:{tierX:"$created_at"},compte:
{$sum:1}}},{$sort:{"_id.tierX":-1}},{$project:{"_id":1,compte:2}}])

Requête Mongodb pour extraction et analyse divers de ce lexique

db.lexique.aggregate([{$group:{_id:{tierX :"$sense.Gloss.tierX"},compte:{$sum:1}}}])
Dans l'ordre décroissant
db.lexique.aggregate([{$group:{_id:{tierX:"$sense.Gloss.tierX"},compte:{$sum:1}}},{$sort:{compte:-1}},{$project:{"_id.tierX":1,compte:2}}])

```
{ "_id" : { "tierX" : "verbe np" }, "compte" : 1 }
{ "_id" : { "tierX" : [ "verbe (sango)", "verbe (sango)" ] }, "compte" : 1 }
{ "_id" : { "tierX" : "verbe (sango)" }, "compte" : 6 }
{ "_id" : { "tierX" : "verbe (français)" }, "compte" : 33 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "verbe", "verbe", "verbe", "verbe", "verbe", "verbe" ] }, "compte" : 1 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "B" ] }, "compte" : 1 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "verbe", "verbe", "verbe", "verbe", "verbe" ] }, "compte" : 2 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "verbe", "verbe", "verbe", "verbe" ] }, "compte" : 5 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "verbe", "verbe", "verbe" ] }, "compte" : 11 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "verbe", "verbe" ] }, "compte" : 25 }
{ "_id" : { "tierX" : [ "verbe", "verbe", "verbe" ] }, "compte" : 38 }
{ "_id" : { "tierX" : [ "verbe", "verbe" ] }, "compte" : 68 }
{ "_id" : { "tierX" : "verbe" }, "compte" : 267 }

{ "_id" : { "tierX" : "préposition?" }, "compte" : 1 }

{ "_id" : { "tierX" : "prosodie" }, "compte" : 17 }

{ "_id" : { "tierX" : "pronom ?" }, "compte" : 1 }
{ "_id" : { "tierX" : "pronom+TAM" }, "compte" : 2 }
{ "_id" : { "tierX" : "pronom inter" }, "compte" : 1 }
{ "_id" : { "tierX" : "pronom ? " }, "compte" : 1 }
{ "_id" : { "tierX" : "pronom ?" }, "compte" : 2 }
{ "_id" : { "tierX" : [ "gram nominal", "pronom", "pronom", "pronom" ] }, "compte" : 1 }
{ "_id" : { "tierX" : "pronom" }, "compte" : 77 }

{ "_id" : { "tierX" : "preposition" }, "compte" : 2 }

{ "_id" : { "tierX" : "particule énonciative ? " }, "compte" : 1 }
{ "_id" : { "tierX" : "particule de fin" }, "compte" : 1 }

{ "_id" : { "tierX" : "numéral (sango)" }, "compte" : 1 }
{ "_id" : { "tierX" : "numéral (français)" }, "compte" : 4 }
{ "_id" : { "tierX" : "numéral" }, "compte" : 8 }

{ "_id" : { "tierX" : [ "nom", "numéral" ] }, "compte" : 1 }
{ "_id" : { "tierX" : "numeral (français)" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom?/adjectif?/adverbe?" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom?" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom preposition" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom de recette" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom de personnes (français)" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom de personnes" }, "compte" : 5 }
{ "_id" : { "tierX" : "nom de personne/lieu" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom de personne (français)" }, "compte" : 39 }
{ "_id" : { "tierX" : "nom de personne" }, "compte" : 44 }
{ "_id" : { "tierX" : "nom de lieu (français)" }, "compte" : 10 }
{ "_id" : { "tierX" : "nom de lieu" }, "compte" : 35 }
{ "_id" : { "tierX" : "nom composé ? " }, "compte" : 2 }
{ "_id" : { "tierX" : "nom ? adverbe ? " }, "compte" : 1 }
{ "_id" : { "tierX" : "nom ? adjectif ? " }, "compte" : 1 }
{ "_id" : { "tierX" : "nom ?" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom (sango ?)" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom (sango?)" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom (sango)" }, "compte" : 14 }
{ "_id" : { "tierX" : "nom (sango ?)" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom (français ?)" }, "compte" : 2 }
{ "_id" : { "tierX" : "nom (français?)" }, "compte" : 1 }
{ "_id" : { "tierX" : [ "nom (français)", "nom (français)" ] }, "compte" : 1 }
{ "_id" : { "tierX" : "nom (français)" }, "compte" : 116 }
{ "_id" : { "tierX" : "nom (composé?)" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom (composé)" }, "compte" : 1 }
{ "_id" : { "tierX" : "nom (composé ?)" }, "compte" : 1 }
{ "_id" : { "tierX" : [ "nom", "nom", "nom", "nom", "nom", "nom", "nom", "nom", "nom", "nom" ] }, "compte" : 1 }
{ "_id" : { "tierX" : [ "nom", "*" ] }, "compte" : 1 }
{ "_id" : { "tierX" : [ "nom", "nom", "nom" ] }, "compte" : 6 }
{ "_id" : { "tierX" : [ "nom", "nom" ] }, "compte" : 21 }
```


quand il ya plusieurs occurrences c'est qu'il y a plusieurs gloss dans sense

```
<lexicalEntry id="125" dt="27/Apr/2020">
  <Lexeme typ="lem">éŋgà</Lexeme>
  <form>
    <altForm>
      <WForm>ʔéngà</WForm>
    </altForm>
    <altForm>
      <WForm>jíŋgà</WForm>
    </altForm>
    <altForm>
      <WForm>íŋgà</WForm>
    </altForm>
  </form>
  <sense>
    <Gloss lang="en" der="H" tierX="verbe (sango)">connaître</Gloss>
    <Gloss lang="en" der="H" tierX="verbe (sango)">considérer, respecter</Gloss>
  </sense>
</lexicalEntry>
```

{ "_id" : { "tierX" : ["*", "nom"] }, "compte" : 1 }
{ "_id" : { "tierX" : "nom" }, "compte" : 311 }
{ "_id" : { "tierX" : ["locatif", "locatif"] }, "compte" : 1 }
{ "_id" : { "tierX" : "locatif" }, "compte" : 16 }
{ "_id" : { "tierX" : "interruption" }, "compte" : 2 }
{ "_id" : { "tierX" : "interjection (français)" }, "compte" : 4 }
{ "_id" : { "tierX" : ["interjection", "interjection"] }, "compte" : 1 }
{ "_id" : { "tierX" : "interjection" }, "compte" : 10 }
{ "_id" : { "tierX" : "gram?" }, "compte" : 1 }
{ "_id" : { "tierX" : "gram/verbe" }, "compte" : 1 }
{ "_id" : { "tierX" : "gram/prep" }, "compte" : 1 }
{ "_id" : { "tierX" : "gram verbal ?" }, "compte" : 1 }
{ "_id" : { "tierX" : "gram verbal" }, "compte" : 11 }
{ "_id" : { "tierX" : "gram verb" }, "compte" : 3 }
{ "_id" : { "tierX" : ["*", "gram nominal"] }, "compte" : 1 }
{ "_id" : { "tierX" : "gram nominal" }, "compte" : 5 }
{ "_id" : { "tierX" : "gram" }, "compte" : 1 }
{ "_id" : { "tierX" : "adverbe ?" }, "compte" : 1 }
{ "_id" : { "tierX" : ["adverbe ?", "adverbe ? "] }, "compte" : 1 }
{ "_id" : { "tierX" : "adverbe ? " }, "compte" : 5 }
{ "_id" : { "tierX" : "adverbe ?" }, "compte" : 3 }
{ "_id" : { "tierX" : ["adverbe ?", "adverbe ?"] }, "compte" : 1 }
{ "_id" : { "tierX" : ["*", "adverbe (sango) ? "] }, "compte" : 1 }
{ "_id" : { "tierX" : ["adverbe", "adverbe"] }, "compte" : 1 }
{ "_id" : { "tierX" : "adverbe" }, "compte" : 7 }
{ "_id" : { "tierX" : "adjectif\\verbe" }, "compte" : 1 }
{ "_id" : { "tierX" : "adjectif/verbe" }, "compte" : 1 }
{ "_id" : { "tierX" : "adjectif/nom (sango)" }, "compte" : 1 }
{ "_id" : { "tierX" : "adjectif/nom" }, "compte" : 5 }
{ "_id" : { "tierX" : "adjectif adverbe" }, "compte" : 1 }
{ "_id" : { "tierX" : "adjectif ? " }, "compte" : 2 }
{ "_id" : { "tierX" : ["adjectif ?", "adjectif ? "] }, "compte" : 1 }
{ "_id" : { "tierX" : "adjectif ?" }, "compte" : 1 }
{ "_id" : { "tierX" : ["adjectif (?)", "adjectif (?)"] }, "compte" : 1 }
{ "_id" : { "tierX" : ["adjectif", "adjectif"] }, "compte" : 1 }
{ "_id" : { "tierX" : ["adjectif", "adjectif", "adjectif"] }, "compte" : 2 }
{ "_id" : { "tierX" : "adjectif" }, "compte" : 18 }
{ "_id" : { "tierX" : "adj/adv" }, "compte" : 1 }
{ "_id" : { "tierX" : "adj" }, "compte" : 1 }
{ "_id" : { "tierX" : "TAM" }, "compte" : 1 }
{ "_id" : { "tierX" : "Pronom ?" }, "compte" : 2 }
{ "_id" : { "tierX" : "Nom de personne" }, "compte" : 1 }
{ "_id" : { "tierX" : "Nom (français)" }, "compte" : 1 }
{ "_id" : { "tierX" : "H" }, "compte" : 1 }
{ "_id" : { "tierX" : "?" }, "compte" : 1 }
{ "_id" : { "tierX" : "* français" }, "compte" : 2 }
{ "_id" : { "tierX" : "* (sango)" }, "compte" : 5 }
{ "_id" : { "tierX" : ["* (sango)", "* (sango)"] }, "compte" : 1 }
{ "_id" : { "tierX" : "* (sango ?)" }, "compte" : 2 }
{ "_id" : { "tierX" : ["* (français)", "* (français)"] }, "compte" : 2 }
{ "_id" : { "tierX" : "* (français)" }, "compte" : 59 }
{ "_id" : { "tierX" : "*" }, "compte" : 141 }
{ "_id" : { "tierX" : ["*", "*"] }, "compte" : 4 }
{ "_id" : { "tierX" : ["*", ""] }, "compte" : 1 }
{ "_id" : { "tierX" : "" }, "compte" : 1 }
{ "_id" : { "tierX" : null }, "compte" : 251 }

Extraction pour vérification des "typ" renseignés

```
> db.lexique.find({"Lexeme.typ":"lem"}).count()
1553
> db.lexique.find({"Lexeme.typ":"wf"}).count()
251
```

Interaction sur le lexique sur la durée


```
{ "_id" : { "tierX" : ISODate("2023-08-15T00:00:00Z") }, "compte" : 10 }
{ "_id" : { "tierX" : ISODate("2023-08-04T00:00:00Z") }, "compte" : 8 }
{ "_id" : { "tierX" : ISODate("2023-07-31T00:00:00Z") }, "compte" : 12 }
{ "_id" : { "tierX" : ISODate("2023-07-25T00:00:00Z") }, "compte" : 8 }
{ "_id" : { "tierX" : ISODate("2023-07-24T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2023-07-21T00:00:00Z") }, "compte" : 18 }
{ "_id" : { "tierX" : ISODate("2023-07-20T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-07-19T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-07-18T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-07-17T00:00:00Z") }, "compte" : 16 }
{ "_id" : { "tierX" : ISODate("2023-07-13T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2023-07-12T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2023-07-10T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-07-08T00:00:00Z") }, "compte" : 7 }
{ "_id" : { "tierX" : ISODate("2023-07-07T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-07-05T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2023-07-04T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2023-07-03T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2023-06-29T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2023-06-06T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-06-02T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2023-06-01T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2023-05-30T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2023-05-29T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-05-20T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-05-19T00:00:00Z") }, "compte" : 8 }
{ "_id" : { "tierX" : ISODate("2023-05-18T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2023-05-03T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-04-28T00:00:00Z") }, "compte" : 7 }
{ "_id" : { "tierX" : ISODate("2023-04-27T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-04-20T00:00:00Z") }, "compte" : 19 }
{ "_id" : { "tierX" : ISODate("2023-04-19T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2023-03-22T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-03-17T00:00:00Z") }, "compte" : 9 }
{ "_id" : { "tierX" : ISODate("2023-03-16T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2023-03-15T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2023-03-14T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-03-08T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2023-03-06T00:00:00Z") }, "compte" : 40 }
{ "_id" : { "tierX" : ISODate("2023-02-22T00:00:00Z") }, "compte" : 7 }
{ "_id" : { "tierX" : ISODate("2023-02-21T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2023-02-17T00:00:00Z") }, "compte" : 27 }
{ "_id" : { "tierX" : ISODate("2023-02-16T00:00:00Z") }, "compte" : 14 }
{ "_id" : { "tierX" : ISODate("2023-02-13T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2023-02-09T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-01-18T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2023-01-10T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2023-01-09T00:00:00Z") }, "compte" : 13 }
{ "_id" : { "tierX" : ISODate("2023-01-04T00:00:00Z") }, "compte" : 12 }
{ "_id" : { "tierX" : ISODate("2022-12-28T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2022-12-19T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2022-11-04T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2022-11-03T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2022-11-02T00:00:00Z") }, "compte" : 22 }
{ "_id" : { "tierX" : ISODate("2022-09-09T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2022-09-08T00:00:00Z") }, "compte" : 15 }
{ "_id" : { "tierX" : ISODate("2022-09-07T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2022-07-25T00:00:00Z") }, "compte" : 9 }
{ "_id" : { "tierX" : ISODate("2022-07-19T00:00:00Z") }, "compte" : 9 }
```

```
{ "_id" : { "tierX" : ISODate("2022-07-18T00:00:00Z") }, "compte" : 15 }
{ "_id" : { "tierX" : ISODate("2022-07-17T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2022-07-12T00:00:00Z") }, "compte" : 9 }
{ "_id" : { "tierX" : ISODate("2022-07-11T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2022-07-09T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2022-07-08T00:00:00Z") }, "compte" : 33 }
{ "_id" : { "tierX" : ISODate("2022-07-07T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2022-07-06T00:00:00Z") }, "compte" : 14 }
{ "_id" : { "tierX" : ISODate("2022-06-24T00:00:00Z") }, "compte" : 8 }
{ "_id" : { "tierX" : ISODate("2022-06-22T00:00:00Z") }, "compte" : 7 }
{ "_id" : { "tierX" : ISODate("2022-06-21T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2022-06-20T00:00:00Z") }, "compte" : 17 }
{ "_id" : { "tierX" : ISODate("2022-06-09T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2022-06-08T00:00:00Z") }, "compte" : 17 }
{ "_id" : { "tierX" : ISODate("2022-06-05T00:00:00Z") }, "compte" : 11 }
{ "_id" : { "tierX" : ISODate("2022-06-04T00:00:00Z") }, "compte" : 20 }
{ "_id" : { "tierX" : ISODate("2022-06-02T00:00:00Z") }, "compte" : 17 }
{ "_id" : { "tierX" : ISODate("2022-05-02T00:00:00Z") }, "compte" : 40 }
{ "_id" : { "tierX" : ISODate("2022-04-20T00:00:00Z") }, "compte" : 68 }
{ "_id" : { "tierX" : ISODate("2022-04-05T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2022-03-23T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2022-03-17T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2022-03-16T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2022-02-18T00:00:00Z") }, "compte" : 13 }
{ "_id" : { "tierX" : ISODate("2022-02-16T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2022-02-15T00:00:00Z") }, "compte" : 8 }
{ "_id" : { "tierX" : ISODate("2022-02-14T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2022-02-11T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2022-02-10T00:00:00Z") }, "compte" : 14 }
{ "_id" : { "tierX" : ISODate("2022-02-09T00:00:00Z") }, "compte" : 6 }
{ "_id" : { "tierX" : ISODate("2022-02-08T00:00:00Z") }, "compte" : 3 }
{ "_id" : { "tierX" : ISODate("2022-02-03T00:00:00Z") }, "compte" : 7 }
{ "_id" : { "tierX" : ISODate("2022-02-02T00:00:00Z") }, "compte" : 5 }
{ "_id" : { "tierX" : ISODate("2022-01-28T00:00:00Z") }, "compte" : 27 }
{ "_id" : { "tierX" : ISODate("2022-01-27T00:00:00Z") }, "compte" : 2 }
{ "_id" : { "tierX" : ISODate("2022-01-26T00:00:00Z") }, "compte" : 19 }
{ "_id" : { "tierX" : ISODate("2021-11-26T00:00:00Z") }, "compte" : 60 }
{ "_id" : { "tierX" : ISODate("2021-11-21T00:00:00Z") }, "compte" : 1 }
{ "_id" : { "tierX" : ISODate("2021-11-18T00:00:00Z") }, "compte" : 12 }
{ "_id" : { "tierX" : ISODate("2021-11-05T00:00:00Z") }, "compte" : 4 }
{ "_id" : { "tierX" : ISODate("2021-11-04T00:00:00Z") }, "compte" : 1 }
```

Travail sur la comparaison entre lexique recomposé et le lexique de référence

Cas 1 : simple

```
"jú": {
   "mot": {
      "jú": {
         "expression": [
            "àndʒē jú à kwūlà nī lə́ sà lə́bà nə́ àndʒé (??)",
            "àlà_ à tó nə́ pākēdə́ mà gɔ̄ŋgbúgù jú nú gbɛ́"
         ],
         "gloss": {
            "jú||plonger\\H||verbe": 1,
            "jú||disparaître\\HAUT||verbe": 1
         }
      }
   }
},
```

```
<lexicalEntry id="831" dt="19/Feb/2021">
 <Lexeme typ="lem">jú</Lexeme>
 <sense>
  <Gloss lang="en" der="H" tierX="verbe">disparaître</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">plonger</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">demander</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">apporter</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">aiguiser</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">saluer</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">être perdu</Gloss>
  <Gloss lang="en" der="H" tierX="verbe">poignarder</Gloss>
 </sense>
 <form />
</lexicalEntry>
<lexicalEntry id="1532" dt="26/Nov/2021">
 <Lexeme typ="lem">jú</Lexeme>
 <sense>
  <Gloss lang="en" tierX="adjectif/nom">femelle</Gloss>
 </sense>
 <form />
</lexicalEntry>
```

```
"kpétè": {
   "mot": {
      "kpétè": {
         "expression": [
            "àndʒē kpétè lə́bà nə̄"
         ],
         "gloss": {
            "kpétè||presser\\H||verbe": 1
         }
      },
      "kpítè": {
         "expression": [
            "àndʒē kpítè lə́bà nə̄"
         ],
         "gloss": {
            "kpétè||presser\\H||verbe": 1
         }
      }
   }
},
```

```
<lexicalEntry id="2421" dt="18/May/2023">
  <Lexeme typ="lem">kpétè</Lexeme>
  <form>
   <altForm>
    <WForm>kpítè</WForm>
   </altForm>
  </form>
  <sense>
   <Gloss lang="en" der="H" tierX="verbe">presser</Gloss>
  </sense>
</lexicalEntry>
```

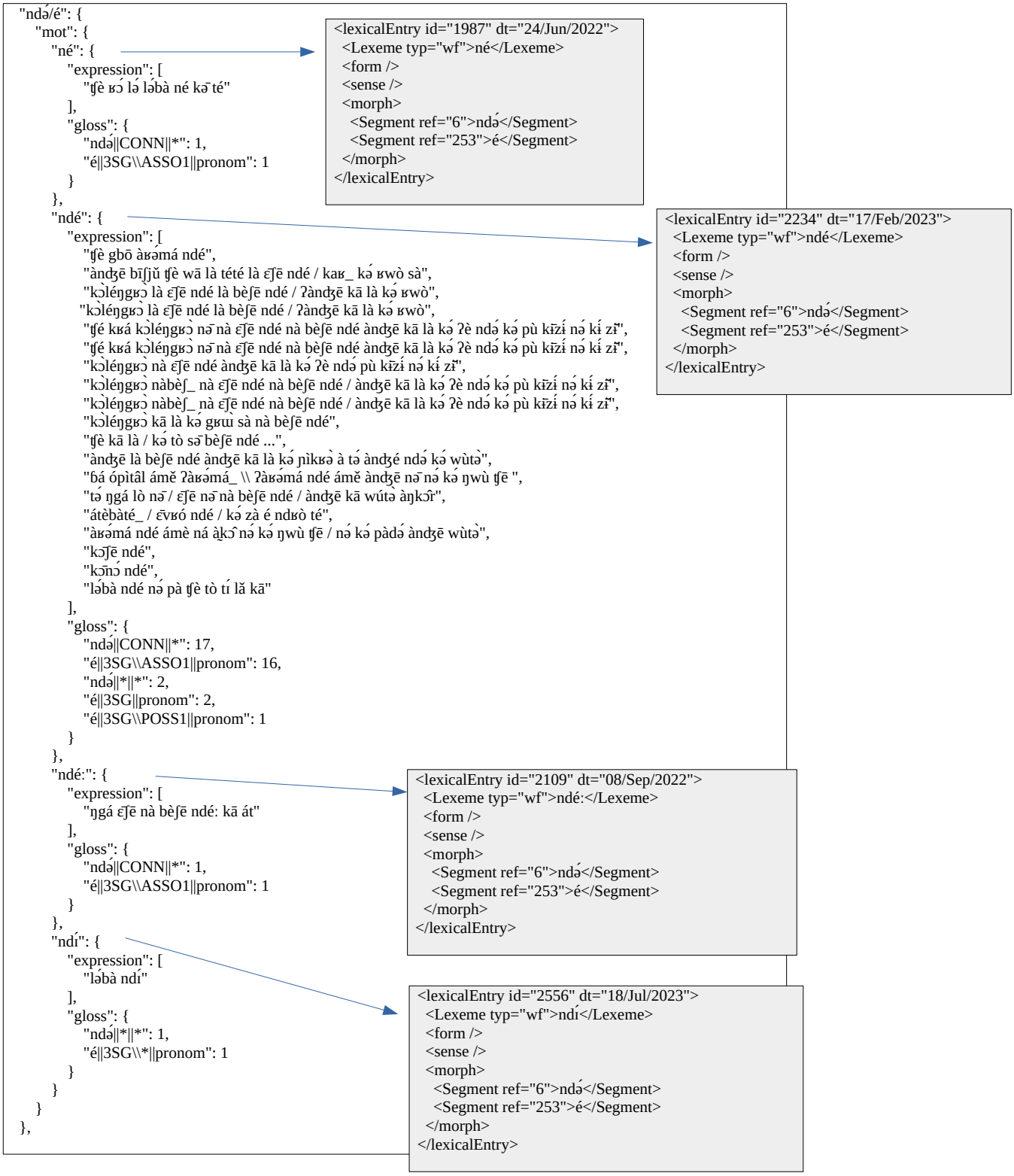
Exemple pour l'analyse des correspondances entre les différents lexiques


```
"ndà": {
   "mot": {
      "ndà": {
         "expression": [
            "... ē tə́ ndà kēkē nə̄||./AR_Caused_positions1.eaf",
            "ʧè zā ŋgā pí ndà kēkē nə̄||./AR_Caused_positions1.eaf",
            "ʧè zā sà kàpù nə̄ kə̄ zá ɓá lə́ ndà kēkē nə̄||./AR_Caused_positions1.eaf",
            "áɲē ɓá lə́ ndà::: / ɓá lə́ ndà jò ɓáɛ̀ àʁə́má nə́ bèʃē nə̄ / àndʒē nə́ nə́ kə́ ŋwù ʧē||./AR_S
            "mī kā là kə́ kɔ̀ ndà||./AR_elicit_13.eaf",
            "mī kā là kə́ kɔ̀ ndà||./AR_elicit_13.eaf",
            "mī kɔ́ ndà nə̄ nə́||./AR_elicit_13.eaf",
            "mī kā là tə́ kə́ kɔ / ndà||./AR_elicit_13.eaf",
            "kɔ̄ ndà lámì sə̄ mí||./AR_elicit_13.eaf"
         ],
         "gloss": {
            "ndà||maison||nom": 9
         }
      },
      "ndà:::": {
         "expression": [
            "áɲē ɓá lə́ ndà::: / ɓá lə́ ndà jò ɓáɛ̀ àʁə́má nə́ bèʃē nə̄ / àndʒē nə́ nə́ kə́ ŋwù ʧē||./AR_SickGirl.eaf"
         ],
         "gloss": {
            "ndà||maison||nom": 1
         }
      }
   }
},
```

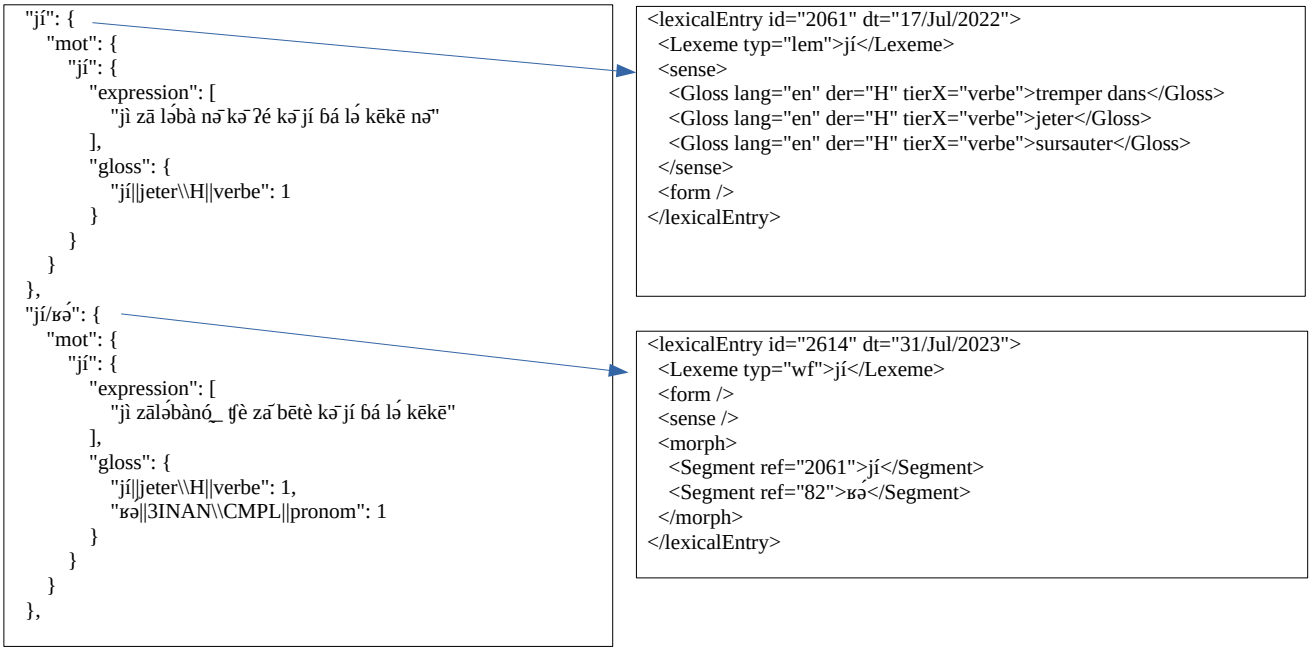


```
<lexicalEntry id="260" dt="04/May/2020">
  <Lexeme typ="lem">ndà</Lexeme>
  <form>
    <altForm>
      <WForm>ndà:::</WForm>
    </altForm>
  </form>
  <sense>
    <Gloss lang="en" tierX="nom">maison</Gloss>
  </sense>
</lexicalEntry>
```

Cas d'une recomposition avec ndə́/é comment la relier au lexique ????? car il n'y a pas d'entrée spécifique ==> les entrées à vérifier sont dans l'autre sens, il faut partir des mots et vérifier si ils sont décrits par des segments qui recomposent la clé



Autre exemple :



```
"jí": {
   "mot": {
      "jí": {
         "expression": [
            "jì zā lə́bà nə̄ kə̄ ʔé kə̄ jí ɓá lə́ kēkē nə̄"
         ],
         "gloss": {
            "jí||jeter\\H||verbe": 1
         }
      }
   }
},
"jí/ʁə́": {
   "mot": {
      "jí": {
         "expression": [
            "jì zālə́bànó_ ʠè zə̄ bētè kə̄ jí ɓá lə́ kēkē"
         ],
         "gloss": {
            "jí||jeter\\H||verbe": 1,
            "ʁə||3INAN\\CMPL||pronom": 1
         }
      }
   }
},
```

```
<lexicalEntry id="2061" dt="17/Jul/2022">
  <Lexeme typ="lem">jí</Lexeme>
  <sense>
    <Gloss lang="en" der="H" tierX="verbe">tremper dans</Gloss>
    <Gloss lang="en" der="H" tierX="verbe">jeter</Gloss>
    <Gloss lang="en" der="H" tierX="verbe">sursauter</Gloss>
  </sense>
  <form />
</lexicalEntry>
```

```
<lexicalEntry id="2614" dt="31/Jul/2023">
  <Lexeme typ="wf">jí</Lexeme>
  <form />
  <sense />
  <morph>
    <Segment ref="2061">jí</Segment>
    <Segment ref="82">ʁə́</Segment>
  </morph>
</lexicalEntry>
```

```
<lexicalEntry id="82" dt="12/Apr/2020">
 <Lexeme typ="lem">ʁə́</Lexeme>
 <form>
   <altForm>
   <WForm>ʁó</WForm>
   </altForm>
    <altForm>
    <WForm>ʁá</WForm>
   </altForm>
   <altForm>
     <WForm>ʁ́</WForm>
   </altForm>
   <altForm>
     <WForm>ʁ</WForm>
   </altForm>
 </form>
 <sense>
   <Gloss lang="en" der="CMPL" tierX="pronom">3INAN</Gloss>
 </sense>
</lexicalEntry>
```

Mise en place du traitement pour la reconstitution du lexique à partir de l'existant et le chargement du lexique de référence à partir du fichier eafl

Reconstruction du lexique à partir de l'existant

Prendre le contenu des différents fichiers eaf et reconstituer la chaîne entre les différents données en partant des mb --> mot --> ge --> rx

Chargement du lexique de référence
Lecture du fichier eafl et chargement de la structure mémoire sous la forme suivante :

Comparaison des deux lexiques à partir de leur structure mémoire

A la lecture et analyse des différents fichiers fournis, on distingue trois cas qu'il faut rapprocher :

Premier cas : simple mb --> mot --> [gloss]

Simple : Exp
--> Mot
--> forme
--> expressions
--> gloss [forme condensée]

Simple : Lexeme (Exp)
--> gloss TierX et Text

Deuxième cas : simple mb --> mots --> [gloss]

Simple : Exp
--> Mot
--> forme1
--> expressions[]
--> gloss[forme condensée]
--> forme2
--> expressions[]
--> gloss[forme condensée]
--> .....

Simple : Lexeme (Exp)
--> altforms [forme1, forme2]
--> gloss TierX et Text

Troisième cas : mb composite --> mots --> [gloss]

Simple : Exp1/Exp2/.../ExpX
--> Mot
--> forme1
--> expressions[]
--> gloss[forme condensée]
--> forme2
--> expressions[]
--> gloss[forme condensée]
--> .....

Simple : Lexeme (forme1)
--> morph [Exp1, Exp2,...., ExpX]
--> gloss TierX et Text

Simple : Lexeme (forme2)
--> morph [Exp1, Exp2,..., ExpX]
--> gloss TierX et Text

Renforcement du code pour les lectures inattendues dans les fichiers eaf non conformes
Soit ils ne sont pas annotés, soit il manque des paragraphes

Contrôle de la présence des blocs à analyse tx, mb, ge, rx si unn de ces blocs est vide (None) on sort en anomalie

```
paragraphe_tx=fonction_remplir("tx")
paragraphe_mot=fonction_remplir("mot")
paragraphe_mb=fonction_remplir("mb")
paragraphe_ge=fonction_remplir("ge")
paragraphe_rx=fonction_remplir("rx")
if paragraphe_mot==None or paragraphe_mb==None or paragraphe_ge==None or paragraphe_rx==None:
   write_log("!!!!!!!!!!!")
   write_log("problème dans la structure du fichier : il manque des paragraphes")
   write_log("!!!!!!!!!!!")
   return
```

Adaptation du code python pour éviter les "deprecated" lors de l'utilisation des getChildren()[0] dans l'exploration des structures XML

```
# Première écriture pour l'exploration du paragraphe qui fournit le bloc XML à analyser
# import xml.etree.ElementTree as ET
   tree = ET.ElementTree(ET.fromstring(paragraphe_tx))
   root=tree.getroot()
   tx=dict()

   list_enfants_tx=root.getchildren()
   for indice in range(0,len(list_enfants_tx)):
      ref_mot=list_enfants_tx[indice].getchildren()[0].attrib['ANNOTATION_REF']
      ref_ID=list_enfants_tx[indice].getchildren()[0].attrib['ANNOTATION_ID']
      valeur=list_enfants_tx[indice].getchildren()[0].getchildren()[0].text
      tx[ref_ID]={"ref_mot":ref_mot,"value":valeur}
   print(tx)



# réécriture pour éviter les "deprecateds" avec getchildren()[0]
   list_enfants_tx=root
   for indice in list(list_enfants_tx):
      for indice2 in list(indice):
         ref_mot=indice2.attrib['ANNOTATION_REF']
         ref_ID=indice2.attrib['ANNOTATION_ID']
         for indice3 in list(indice2):
            valeur=indice3.text
         tx[ref_ID]={"ref_mot":ref_mot,"value":valeur}
   print(tx)
```

Annexes

Questions et orientations possibles :

- Comment et quelles sont les règles e constructions d'un lexique de ce type ?
  - Existe t-il une procédure à suivre ?
- Quels sont les outils ?
- Quelles sont les difficultés
- Pourquoi le lexique est en français uniquement ?
  - lang="en" à l'intérieur des éléments

- Comment valoriser ce type de travail ? et comment l'exposer ?
  - Pour la communauté ?
  - Pour le Labo ?