Mémoire de Master 2
Ingénierie Linguistique

Université Paris III - La Sorbonne Nouvelle
Département de Langues, Linguistique et Didactique

Uyên-To Doan-Rabier
utdoanrabier@gmail.com

INTERFACE DE RECHERCHE WEB POUR L'EXPLORATION DE CORPUS ORAUX ANNOTÉS

Mémoire dirigé par Pierre Zweigenbaum
Stage effectué au CNRS/LLACAN UMR 8135

21 février 2013

Table des matières

Introduction

Dans les dernières décennies s'est développé un intérêt particulier pour les productions orales dans le domaine du traitement automatique des langues qui a pour objectif de leur faire prendre un statut de matière de recherche valorisable.

Comme le dit Blanche-Benveniste citée dans [Dister and Simon, 2008],« l'oral ne peut être étudié par l'oral », et ne devient objet d'étude à part entière qu'à partir de sa mise par écrit. Se posent alors la question de la spécificité de l'encodage des phénomènes oraux par rapport à ceux de l'écrit et la question de l'encodage des langues sans standard orthographique.

Déjà à la fin des années 80, les problèmes posés par la transcription de l'oral ont été abordés dans le cadre d'un corpus du français parlé en Belgique (centre de recherche VALIBEL) et en particulier celui de la compatibilité avec un traitement informatisé.

Bénéficiant du développement de technologies dédiées à l'analyse d'interactions orales, la linguistique des langues à tradition orale ou peu connues a pu trouver un essor nouveau dans sa tâche de description et de constitution de ressources.

C'est dans ce contexte que le projet ANR CorpAfroAs a pu voir le jour.

Les données linguistiques pour les « petites » langues sont souvent difficiles à localiser et on les trouve surtout dans les grammaires sous forme d'exemples sans possibilité d'accès aux enregistrements d'origine.

Ainsi le projet a réuni un groupe de linguistes de terrain, dans le but, non seulement de rassembler des enregistrements et des transcriptions de langues afroasiatiques mais aussi de fournir des méthodes de travail ainsi que des outils informatiques facilitant l'accès et la consultation des ressources et susceptibles d'être réutilisés par d'autres groupes de recherche.

Dans le cadre de ce mémoire, nous nous attacherons d'une part à la description générale du projet afin d'établir le cadre de notre travail (chapitres 1 à 4), puis d'autre part à ce qui relève uniquement du travail effectué dans le cadre du stage, c'est-à-dire le développement de notre interface de recherche et l'exploitation des données que l'on peut en tirer (chapitres 5 et 6).

Après une présentation des objectifs linguistiques, méthodologiques et techniques du projet CorpAfroAs, nous ferons une présentation plus détaillée du corpus en développant les principes qui ont régi son annotation. Puis nous procèderons à un état de l'art des outils d'annotation de corpus oraux qui nous permettra de justifier le choix qui a été fait quant aux logiciels de transcription et d'annotation et donnerons un aperçu du logiciel ELAN utilisé pour la tâche d'annotation dans la section 4.

Ce sont dans les parties suivantes que nous aborderons plus spécifiquement notre contribution à la création de l'outil de consultation et de recherche sur le corpus du projet. Ainsi, dans une cinquième partie, nous décrirons l'outil de recherche web que nous avons développé dans le cadre du stage et enfin nous fournirons, dans une sixième partie, des exemples d'utilisation des données qui peuvent être produites par cet outil, à des fins d'analyse quantitative ou de correction, avant de proposer des pistes de réflexion pour les futurs travaux.

Chapitre 1

ANR CorpAfroAs

Les langues étudiées dans ce projet recouvrent un large éventail aussi bien du point de vue de leurs caractérisiques linguistiques (langues avec ou sans ton, à morphologie plus ou moins riche, concaténative ou non concaténative...) que du point de vue de leur nombre de locuteurs (il peut s'agir de langues connues et bien décrites ou au contraire peu connues voire en danger de disparition).

Avec le développement de la typologie et le souci croissant face à la disparition rapide (faute de locuteurs) de la valeur patrimoniale de centaines de langues (parmi les 6000 parlées dans le monde), la description des langues est jugée de plus en plus nécessaire. Il ne s'agit pas d'enrayer ce phénomène mais plutôt de s'ériger en conservateur de la diversité. Les initiatives encourageant la récolte, la conservation, la description et la diffusion d'enregistrements sur des « langues en danger » parviennent ainsi à trouver des financements.

Ce projet s'inscrit aussi clairement dans un intérêt particulier pour la dimension discursive de l'analyse linguistique, pour l'oralité en ce qu'elle est susceptible d'être étudiée et conservée.

Nous allons voir, dans cette première partie, les objectifs linguistiques et technologiques fixés par le projet.

1.1 Objectifs du point de vue linguistique

Le but n'était pas tant une véritable analyse de la langue parlée (versus écrite avec par exemple l'analyse de dysfluences, des variations d'ordre linéaire, des reprises anaphoriques, etc.) qu'une utilisation de ce corpus pour étudier des phénomènes :

- > phonologiques : *assimilation* (modification phonétique subie par un son au contact d'un son voisin (contexte), qui tend à réduire les différences entres les deux), *allophonie* (variation des réalisations possibles d'un phonème), *élision*, *liaison*, *accentuation*...
- > morphosyntaxiques ainsi que leur rapport avec les structures intonatives
- > lexicaux : comme par exemple l'alternance de langues (ou *code switching*)
- > prosodiques et intonationnels : comme par exemple l'étude du lien entre prosodie et intonation du topic et du focus.

Il s'agissait pour les chercheurs non seulement de tester des hypothèses préétablies mais aussi de faire émerger des phénomènes linguistiques nouveaux afin d'affiner la grammaire de leur langue.

Les questions de structure informationnelle ne sont pas étudiées dans ce projet mais les données de ce corpus pourront servir de base dans le cadre d'une telle

recherche [1].

1.1.1 Découpage prosodique et glose morphosyntaxique

La segmentation du corpus en unités prosodiques (périodes intonatives, unités majeures/terminales et mineures/non-terminales) plutôt que morphosyntaxiques procède d'un choix de la part du projet de tenir compte de la dimension orale du corpus et d'explorer le lien qui peut exister entre prosodie et syntaxe, d'où l'ajout de la glose morphosyntaxique.

Bien qu'aucune autre annotation prosodique (tons, contours...) n'ait été ajoutée, la possibilité de se reporter à l'enregistrement doit éventuellement permettre une exploration plus poussée des phénomènes prosodiques si nécessaire.

En amont du travail d'annotation, la spécification des conventions pour ces annotations est une étape primordiale si l'on ne veut pas perdre du temps à devoir modifier les annotations déjà saisies.

Les difficultés rencontrées lors de l'élaboration de ces conventions sont discutées dans la section 2.3.3.

1.1.2 Quelques exemples de phénomènes étudiés

L'étude du corpus annoté devait permettre non seulement la vérification de la régularité et de la généralité de certains faits linguistiques mais aussi la découverte de nouveaux phénomènes (tels que les types de verbe utilisés au mode imperfectif, les contextes d'utilisation des particules de focus, les fréquences d'utilisation des cas génitif et locatif qui ont tendance à être abandonnés...).

L'annotation morphosyntaxique sur plusieurs niveaux (en anglais *tiers*), devait donc permettre une description suffisamment fine pour répondre aux questions linguistiques qui se sont posées.

L'alignement des annotations sur l'enregistrement permet également d'étudier plus facilement les phénomènes phonologiques.

A titre d'exemple, on peut citer quelques phénomènes découverts à l'aide de l'étude des corpus pour le Beja et le Zaar :

> Beja
- phénomènes d'allophonie localisés sur le niveau `tx`
- instabilité du système casuel avec des cas de décalage entre la fonction syntaxique et la marque casuelle (ex : des sujets portant la marque accusative ou bien le génitif utilisé à la place du locatif)
- formation du diminutif pour les adjectif (ajout du suffixe *-al*)
- position de l'adjectif variable suivant les contextes même si l'antéposition est plus fréquente
- différence de fonctionnement entre propositions relatives et complétives

> Zaar
- étude détaillée des phrases complexes (subordination complétive, circonstancielle et relative, coordination), souvent construites de façon paratactique
- désambiguïsation de certains adverbes phonétiquement similaires à des conjonctions
- étude de la ligne mélodique (F0) dans les tons grammaticaux et lexicaux (baisse de la F0 dans les questions ouvertes)
- étude des idéophones (souvent obtenus par répétition de mots)

1. Etude du problème prévu dans le 3ième axe du labex EFL (Typologie et dynamique des systèmes linguistiques)

Mais pour répondre à ces questions, il était bien sûr nécessaire que les informations soient codées de façon standardisée dans les niveaux de gloses et d'étiquetage morphosyntaxique.

1.2 Objectifs du point de vue mutualisation du corpus

L'objectif du projet est non seulement d'établir une méthodologie de partage et d'unification des données de terrain dans la famille des langues afro-asiatiques mais aussi de mettre à disposition des chercheurs, linguistes de terrain et typologues :

> un corpus de transcriptions traduites et annotées accompagnées de leur métadonnées ainsi que d'une esquisse grammaticale pour chaque langue

> des articles de typologie comparative (comme par exemple l'étude de l'intonation du topic et du focus dans différentes langues du projet)

> des outils à la fois pour la création de nouveaux corpus au format CorpAfroAs (avec une liste de plus de 300 gloses standardisées) et leur exploration à l'aide d'un outil de recherche permettant de formuler des requêtes complexes.

La gestion d'un corpus s'articule traditionnellement autour de 3 axes principaux et interdépendants :

> la sauvegarde. Les progrès technologiques et le développement du numérique ont permis aux linguistes de terrain d'avoir des outils plus performants pour leurs études tels que des systèmes d'enregistrement plus efficaces pouvant permettre une synchronisation temporelle entre le signal et une transcription. Les supports audio de meilleure qualité assurent également une sauvegarde plus durable.

De plus, la conservation des fichiers d'annotation et audio sur une plateforme hébergée par la très grande infrastructure TGE Adonis dont la mission principale est d'« assurer l'accès et la préservation des données numériques produites par les sciences humaines et sociales » [2] est une bonne garantie pour la pérennisation des données.

> la normalisation des données : il s'agit d'avoir des données décrites dans un métalangage partagé par tous, car dire que l'on se sert d'une norme particulière permet de ne pas avoir à redéfinir les concepts que l'on manipule et permet aussi une meilleure interopérabilité.

Ainsi pour le codage des métadonnées (l'ensemble des descripteurs des enregistrements et des annotations), c'est la norme OLAC (Open Language Archives Community) qui a été choisie (IMDI en est un autre exemple et c'est celui utilisé par le MPI). Ce codage est inspiré du DCMI (Dublin Core Metadata Initiative). Il est composé d'un jeu d'une quinzaine d'étiquettes (title, creator, subject, description, publisher...).

> la disponibilité des corpus : celle-ci est un problème récurrent. Les corpus sont souvent cités dans les articles mais rarement accessibles à tout un chacun. Les linguistes ont du mal à accéder aux données des autres linguistes qui peuvent parfois être réticents à diffuser librement un travail coûteux et laborieux.

Les objectifs une fois posés, nous pouvons maintenant regarder la façon dont a été constitué le corpus pour essayer de répondre à ces objectifs, en particulier linguistiques.

2. www.tge-adonis.fr/le-tge-en-bref, site consulté le 21/11/2012

Chapitre 2

Présentation du corpus CorpAfroAs

2.1 Langues étudiées

Les 12 langues étudiées se subdivisent en 5 groupes principaux :
- berbère (Kabyle, Tamashek),
- tchadique (Hausa, Zaar),
- cushitique (Beja, Gawwada, Ts'amakko),
- omotique (Wolaitta),
- sémitique (Arabe marocain, Arabe lybien, Arabe Juba, Hébreu).

En dehors du Hausa et du Zaar annotées par le même chercheur et de l'arabe marocain annoté par 4 chercheurs différents, chaque langue n'a été étudiée que par un chercheur à la fois.

Les données récoltées étaient soit des conversations (dialogues) soit des narrations (mais qui ne sont pas toujours des oraux purement monologaux, il peut quelquefois y avoir l'intervention d'un autre locuteur).

L'objectif initial était de récolter une heure d'enregistrement par langue (narrations et conversations confondues).

Il existe une différence dans les unités prosodiques suivant le type de discours, cependant, il n'a pas été envisagé de faire une étude translingue de ces phénomènes dans le projet.

Il a souvent été beaucoup plus facile au chercheur de faire des transcriptions de narrations que de conversations mais, en dehors des langues pour lesquelles les locuteurs ont refusé, pour diverses raisons, d'enregistrer leurs conversations, il a paru utile et intéressant d'avoir des échantillons de situation de communication spontanée qui reflètent probablement mieux les usages de la langue parlée.

Nous présentons dans le tableau 2.1, un récapitulatif de quelques données concernant l'ensemble du corpus.

2.2 Métadonnées

Il s'agit d'annotations qui portent non pas sur le signal proprement dit mais sur les informations concernant la situation d'enregistrement (les locuteurs, le lieu, la date) ainsi que le transcripteur. Elles font donc partie de la documentation du corpus, en le caractérisant et en lui fixant des bornes.

Leur consultation doit permettre au lecteur de savoir de quel type de langue il s'agit, le contexte de production des données, leur format...

Famille	**Berbère**		**Tchadique**		**Cushitique**			**Omotique**	**Sémitique**				**Total**
Langues	Kabyle	Tamashek	Hausa	Zaar	Beja	Gawwada	Tsamakko	Wolaytta	Arabe marocain	Arabe lybien	Juba	Hébreu	/
(code iso)	(kab)	(taq)	(hau)	(say)	(bej)	(gwd)	(tsb)	(wal)	(ary)	(ayl)	(pga)	(heb)	
Annotateur	AM	CL	BC	BC	MV	MT	GS	AA	AB AV DC	CP	SM	IM	/
Nbre fichiers (narr/conv)	3 (3/0)	3 (3/0)	7 (3/4)	6 (3/3)	18 (18/0)	8 (8/0)	13 (10/3)	5 (5/0)	17 (16/1)	7 (7/0)	4 (2/2)	7 (4/3)	98 (82/16)
Nbre mots	5929	474	11853	10690	5877	2528	2085	1407	11337	3335	9617	7734	71459
Nbre mots par famille	6403		22543		10490			1407	32023				/
Nbre gloses (ge+rx)	19002	2000	27303	24376	30063	12159	10798	6093	36405	10278	20541	27813	226831

TABLE 2.1 – Récapitulatif des données disponibles au 11/02/2013

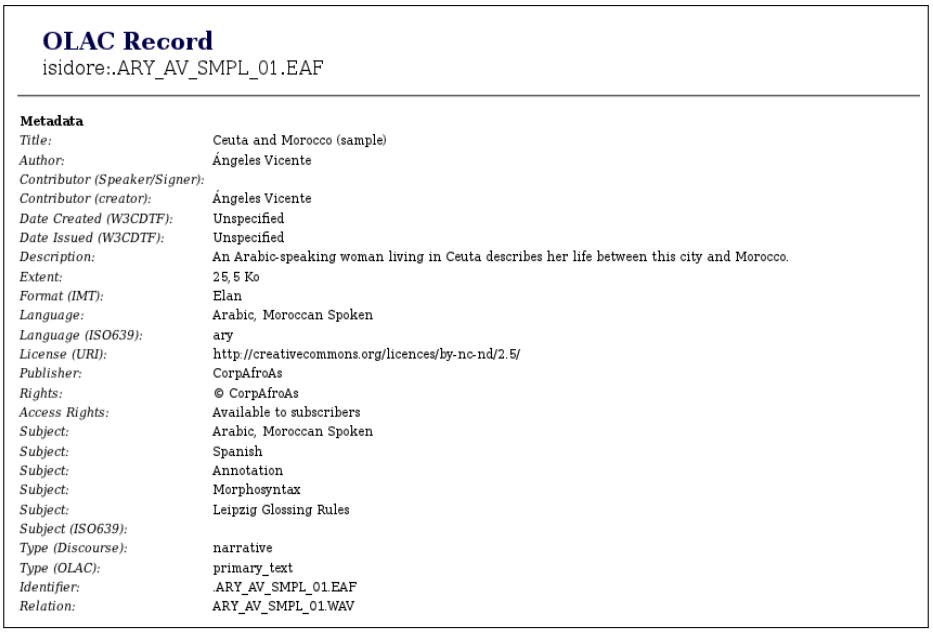
De plus elles rendent les ressources disponibles et comme elles sont utilisées par les moteurs de recherche à des fins d'indexation et de catalogage, elles sont primordiales pour la visibilité du corpus.

Dans le projet, les métadonnées sont externes, c'est-à-dire séparées de la ressource, et sont stockées sous 2 formats :

- le format IMDI (*ISLE Meta Data Initiative*), particulièrement adapté pour la description de ressources linguistiques multimedia et multimodales. C'est dans ce format que doivent être stockées les métadonnées des fichiers ELAN si l'on veut pouvoir les consulter depuis l'interface du logiciel ELAN.
- le standard OLAC (*Open Language Archives Community*) lui-même basé sur l'ensemble des 15 métadonnées du Dublin Core auxquelles ont été ajoutées un certain nombre de descripteurs particulièrement utiles aux linguistes comme le descripteur *spatial* qui désigne généralement le point d'enquête, ou encore *created* pour la date de création de la ressource.
  OLAC avait été développé à la base pour permettre l'échange des métadonnées dans le cadre de l'*Open Archives Initiative* (OAI).

Sur le site de l'outil de recherche [1], les métadonnées sont consultables dans le format OLAC. Ce sont des données accessibles sans identification de l'usager.

1. http://corpafroas.tge-adonis.fr/Archives/ListeFichiersELAN.php consulté le 10 janvier 2013



OLAC Record
isidore:.ARY_AV_SMPL_01.EAF

Metadata

Title:	Ceuta and Morocco (sample)
Author:	Ángeles Vicente
Contributor (Speaker/Signer):	
Contributor (creator):	Ángeles Vicente
Date Created (W3CDTF):	Unspecified
Date Issued (W3CDTF):	Unspecified
Description:	An Arabic-speaking woman living in Ceuta describes her life between this city and Morocco.
Extent:	25,5 Ko
Format (IMT):	Elan
Language:	Arabic, Moroccan Spoken
Language (ISO639):	ary
License (URI):	http://creativecommons.org/licences/by-nc-nd/2.5/
Publisher:	CorpAfroAs
Rights:	© CorpAfroAs
Access Rights:	Available to subscribers
Subject:	Arabic, Moroccan Spoken
Subject:	Spanish
Subject:	Annotation
Subject:	Morphosyntax
Subject:	Leipzig Glossing Rules
Subject (ISO639):	
Type (Discourse):	narrative
Type (OLAC):	primary_text
Identifier:	.ARY_AV_SMPL_01.EAF
Relation:	ARY_AV_SMPL_01.WAV

FIGURE 2.1 – Exemple de données Olac visualisables sur le site de CorpAfroAs

2.3 Transcription et annotations

Pour qu'un corpus soit exploitable, les règles de transcriptions préalablement établies doivent être appliquées par tous les transcripteurs et les annotations qu'il contient se doivent d'avoir une structure et une finesse permettant une interrogation des données pertinente pour les besoins du linguiste.

Les langues sur lesquelles les linguistes de terrain travaillent sont souvent des langues vernaculaires, « minoritaires », sans tradition d'écriture ou sans écriture normalisée. Elles sont donc utilisées parallèlement à des langues véhiculaires qui sont les langues de l'éducation, de l'administration et des medias. C'est la raison pour laquelle le corpus présente de nombreux phénomènes de contact de langues, en particulier de code switching vers des langues qui, elles, ont standards orthographiques. Pour ces quelques cas, la question de la transcription orthographique peut donc se poser.

Néanmoins, pour la grande majorité du corpus, nous nous situons dans le cadre d'une transcription phonétique.

Comme dans de nombreux autres projets, il a été choisi de faire une annotation sous forme de texte interlinéaire qui est la retranscription d'un phénomène linguistique aligné avec son analyse linguistique et que nous étudierons de façon plus détaillée dans la section 2.3.2.

Dans la mesure où les phénomènes phonétiques, tels que l'intonation, n'ont pas été annotés, il s'est avéré avantageux d'offrir à l'utilisateur l'enregistrement aligné à la transcription.

Nous allons voir dans cette partie les choix qui ont été faits quant à l'unité de découpage du corpus, la structure de l'annotation ainsi que les principes appliqués pour l'attribution des gloses.

2.3.1 La segmentation prosodique : choix de l'unité d'analyse

On entend par unité d'analyse des blocs de texte adjacents reliés chacun à une plage sonore.

Dans les fichiers d'annotation de CorpAfroAs, cette unité d'analyse correspond au découpage de la couche `ref`.

Le concept de phrase syntaxique étant une notion essentiellement liée à l'écrit, les marques de segmentation usuelles des textes écrits ne pouvaient être utilisées pour la segmentation de ce corpus.

Le choix d'une unité de base pour le découpage à la fois respectueuse des spécificités de l'oral, pertinente pour les langues étudiées et adaptée au degré d'analyse requis a depuis longtemps fait l'objet de nombreuses propositions (la phrase, le « tour », les « units of cognitive information » de [Gumperz and Berenz, 1990]...)

Dans la mesure où les unités tonales ou intonatives (notées UI à partir de maintenant) de [Chafe, 1993] sont reconnues comme étant des segments cohérents du point de vue soit sémantique, soit informationnel, soit même syntaxique [Izre'el, 2005], et qu'un des buts du projet était de rendre compte de la prosodie par le biais de la segmentation, c'est ce découpage qui a été adopté.

Une UI peut coïncider avec une phrase/clause mais aussi avec une unité syntaxique plus petite comme la phrase nominale ou adverbiale ou bien, au contraire, ne correspondre à aucune unité syntaxique classique.

Il n'y a pas d'annotation de la prosodie (mélodie, accentuation) dans ce corpus, cependant l'indexation du texte, et donc des annotations, sur le son doit permettre une étude du lien entre la prosodie et des phénomènes à d'autres niveaux linguistiques.

Lors de la tâche de découpage, ces UI sont tout d'abord repérées à l'oreille par le linguiste en collaboration avec son informateur puis ce découpage est vérifié physiquement par la détection des signaux visuels de frontière (pause, augmentation ou baisse de la F0, etc).

Les signaux permettant de détecter la présence d'une frontière d'unité prosodique sont :

- l'allongement final (en fin de période),
- l'anacrouse (inaccentuation en début de période),
- l'amplitude du saut, c'est-à-dire la différence de hauteur entre la dernière valeur de F0 précédant la pause et la première valeur de F0 suivant la pause (*pitch reset*),
- la durée de la pause.

Il peut quelquefois y avoir une confusion entre une frontière d'UI et une pause, mais les pauses sont tout de même majoritairement des frontières d'unités.

Les UI sont partagées en deux groupes : les unités non-terminales (ou mineures) et les unités terminales (ou majeures). La distinction se fait à nouveau à l'aide de Praat (cf 3.3).

Une unité majeure possède une frontière souvent caractérisée par une chute de l'intonation, mais elle peut quelques fois être délimitée, au contraire, par une augmentation de l'intonation. Ainsi la meilleure façon de la répérer est d'écouter cette unité seule : si l'on n'a pas une impression d'achèvement, on peut conclure que l'on a à faire à une unité mineure.

2.3.2 Structure et granularité de l'annotation

L'annotation se présente sous forme de texte interlinéaire, qui est une méthode fréquemment utilisée pour la visualisation de données en linguistique.
[Bird et al., 2002] définissent le texte interlinéaire comme suit :

> « a kind of text in which each word is annotated with some combination of phonological, morphological and syntactic information (displayed under the word) and each sentence is annotated with a free translation. »

Cependant cette définition ne permet pas de rendre compte du fait que ce type d'annotation peut aussi être utilisé pour marquer les relations temporelles entre un enregistrement audio (et/ou video) et sa transcription.

En revanche, la définition de [Bird et al., 2002] correspond bien à la « relation d'équivalence », définie par [Schmidt, 2003] comme étant la relation unissant deux éléments qui sont de même nature et non pas simplement synchronisés sur le même axe temporel.

Dans cette approche, chaque mot est noté dans un paradigme pouvant comporter plusieurs lignes et chaque ligne de ce paradigme représente une analyse du mot en question.

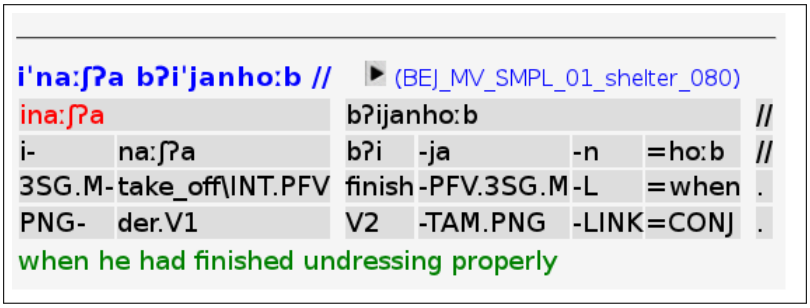


FIGURE 2.2 – Exemple de texte interlinéaire du corpus du Beja

Comme on peut le voir sur la figure précédente, les annotations se font sur 6 couches : 1 couche de transcription phonologique, 4 couches d'analyse morphosyntaxique et 1 couche de traduction.

Une couche donnée correspond donc à un niveau d'analyse et à un locuteur.

S'il y a interaction entre plusieurs locuteurs, chacun d'eux est associé à un ensemble de 6 couches, chaque ensemble étant représenté l'un en dessous de l'autre et parallèle à la ligne de temps, ce qui permet de rendre facilement compte de la simultanéité de certaines transcriptions et donc des éventuels chevauchements.

Voici un bref descriptif de chacune de ces couches :

- `ref` : référence de l'unité, permet d'identifier de façon unique l'unité en question. Elle est construite à partir du nom du fichier auquel est ajouté un nombre correspondant au rang de l'unité dans le fichier (ex : dans la figure précédente, il s'agit de la 80ème unité du fichier BEJ_MV_SMPL_01_shelter).
- `tx` : transcription phonétique large, proche de la réalisation phonétique, qui ne retient que les phénomènes phonologiques pertinents pour la segmentation (assimilation, dissimilation). Les frontières des unités terminales et non-terminales (// et /) sont indiquées à ce niveau, les pauses et les inspirations (notées `BI`) supérieures à 100 ou 200 ms sont indiquées dans une unité séparée. La transcription se fait à l'aide de l'API.
- `mot` : transcription phonologique. Il s'agit du niveau intermédiaire avant la segmentation suivante. Les voyelles et les consonnes sont transcrites selon leur valeur phonologique, plus d'assimilation phonétique mais les changements morphophonologiques restent. Cette ligne peut contenir des allomorphes.
- `mb` : découpage morphémique avec unification des éventuels allomorphes par la forme sous-jacente.
- `ge` : glose correspondant à `mb` et basée sur les « Leipzig glossing rules ».
- `rx` : catégorie grammaticale mais peut contenir plusieurs types d'informations : information morphologique complémentaire, informations syntaxiques, informations sémantiques.
- `ft` : traduction libre de l'unité prosodique `tx`.

– `mft` : uniquement utilisé dans certaines langues pour lesquelles le verbe est en position finale. Une unité `mft` regroupe plusieurs `ft` et permet au lecteur d'avoir une ligne de traduction plus compréhensible.

Ainsi, la couche `tx` présente une transcription phonétique large alors que la couche `mot` donne une représentation phonologique. En revanche, les unités de la couche `tx` sont des unités phonologiques alors que celles de la couche `mot` sont morphosyntaxiques. En comparant les couches `tx` et `mot`, on peut donc retrouver les phénomènes phonétiques et phonologiques caractéristiques de la production orale. Alors que la comparaison entre les couches `mot` et `mb`, permet de dégager les phénomènes morphophonologiques de la langue ainsi que la structure des mots (clitiques, affixes).

2.3.3 Définition des gloses

En partant de l'éventail de base proposé par les « Leipzig Glossing Rules », une liste officielle de gloses a été dressée pour essayer de tenir compte de la diversité des phénomènes linguistiques mais aussi des théories linguistiques en usage dans les différents groupes de langues.

Toute la difficulté a résidé dans l'obtention d'un degré optimal d'unification des annotations pour à la fois respecter les spécificités des langues et fournir une base suffisamment uniforme dans un but de comparabilité.

En d'autres termes, une fois les catégories fondamentales de chaque langue établies, il s'agissait de trouver des catégories syntaxiques à la fois suffisamment précises mais aussi assez larges pour qu'elles puissent être utilisées dans tous les corpus du projet, autrement dit, relier la description de ces langues « à la problématique générale des descriptions syntaxiques » [Creissels, 1991].

Il est très important que cette étape soit effectuée avec soin car de sa qualité et sa cohérence dépendent les résultats de recherche.

Le linguiste pourra ainsi récupérer des informations pertinentes pour découvrir des structures morphosyntaxiques et illustrer un phénomène langagier par des exemples dans une grammaire ou un article.

Les principes généraux d'annotation sur la couche `ge` sont indiqués sur le site de CorpAfroAs et nous en rappelons quelques-uns ici.

– Les gloses morphologiques sont en lettres capitales, les gloses lexicales en lettres minuscules. Les exceptions à cette règle sont les noms propres dont la première lettre est en capitale et le « n » minuscule accolé aux gloses en capitales pour désigner le contraire (ex : nASS = non-assertive).
– Lorsque plusieurs étiquettes de gloses sont nécessaires pour décrire un même élément, celles-ci sont séparées par des points (ex : SBJ.3SG.M pour sujet 3ème personne du singulier, masculin).
– Si une propriété grammaticale d'une unité lexicale se manifeste par une modification morphophonologique (alternance vocalique, tonale, mutation consonantique...), celle-ci est séparée de l'unité lexicale par un anti-slash (ex : write\PFV).
– Le tiret bas doit être utilisé pour marquer les espaces. Par exemple dans la partie correspondant à l'information sémantique du morphème, celle-ci peut nécessiter l'usage de plusieurs mots qui seront alors séparés par le caractère de soulignement (ex : « be_tall »).

Pour ce qui est de la couche `rx`, son but était de pouvoir fournir une étiquette plus générale ou bien de permettre d'ajouter des informations qui n'avaient pas pu être indiquées sur la ligne `ge`. Cette couche ne doit donc, en théorie, contenir aucune information lexicale (aucune minuscule).

Un certain nombre de questions sont apparues à l'élaboration de cette liste de gloses. Elles ont été discutées et les solutions apportées pour le cas des langues de la famille afroasiatique ont vocation à être réutilisées dans d'autres familles.

A titre d'illustration, nous citerons 3 exemples pris dans les corpus du kabyle et des dialectes arabes :

- Parmi les étiquettes des descriptions traditionnelles quelquefois vieilles de 100 ans de l'arabe et du berbère, on peut trouver, respectivement, les notions de *conjugaison suffixale* et d'*état libre*. Ces catégories ont été remplacées par les étiquettes *perfectif* et *absolu* qui sont plus largement utilisées dans d'autres traditions.
- Pour le berbère, la conservation de l'étiquette *état annexé* (aussi appelé *état construit* pour les arabisants) a été préféré à son interprétation casuelle par l'étiquette *nominatif marqué* jugée trop réductrice.
- Toujours pour le berbère, on trouve dans la grammaire traditionnelle l'étiquette *participe* qui désigne la forme particulière d'un verbe dans certaines propositions relatives. Comme cette étiquette caractérise une autre notion dans d'autres langues, en particulier indo-européennes, il a fallu créer une nouvelle étiquette pour rendre compte de ce phénomène.

Il est à noter que cette liste de gloses établie est extensible et qu'il est possible de faire une demande d'ajout par l'intermédiaire d'un formulaire disponible sur le site du projet, à condition que la glose en question soit en adéquation avec les « Leipzig glossing rules » et les besoins du projet.

2.3.4 Limites du système de transcription adopté

Malgré le temps consacré à cette importante étape de définition des gloses et les nombreuses concertations qui ont eu lieu, il reste des divergences d'annotations entre chercheurs qui peuvent être dues soit à des erreurs dans l'application des règles (par exemple pour les phénomènes de code switching, la règle d'utilisation des crochets séparés du mot emprunté n'a pas toujours été appliquée), soit à des désaccords théoriques sur l'interprétation de certains phénomènes linguistiques.

Ceci reflète bien la difficulté évoquée plus tôt quant à la découverte d'un terrain d'entente suffisamment large pour recouvrir tous les besoins mais aussi suffisamment borné pour permettre un travail de comparaison.

Il existe également d'autres limites à ce système comme par exemple le problème de l'annotation des phénomènes suprasegmentaux qui relève plus de la prosodie.

En effet, le type d'annotation adopté pour ce projet ne permet pas de rendre compte des variations intonatives ou les accents de syntagmes caractérisés par les montées et les descentes de F0.

Néanmoins, pour certaines langues et certains textes, le stress a été noté par l'apostrophe au niveau de la couche `mot` ou `mb`.

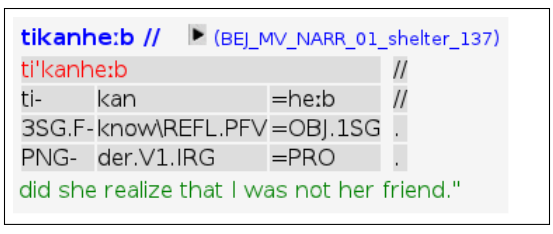


FIGURE 2.3 – Exemple de notation du stress sur la ligne `mot` dans le corpus du Beja

2.4 Conclusion

Nous venons de voir, dans ces deux premiers chapitres, les objectifs du projet CorpAfroAs ainsi que la composition et la structure de son corpus. La construction d'un tel corpus de ressources numériques, à la fois sonores et textuelles, nécessite l'utilisation d'outils informatiques accessibles, adaptés aux objectifs et capables de générer des formats adéquats pour une exploitation ultérieure ou une réutilisation dans d'autres contextes.

Nous allons donc maintenant établir un état des lieux des outils d'annotation de corpus oraux afin de justifier le choix qu'il a été fait quant aux logiciels de transcription et d'annotation pour le projet.

Chapitre 3

Etat de l'art des outils d'annotation de corpus oraux

Tout comme la linguistique textuelle, la linguistique de terrain travaille sur des corpus écrits, à la différence près que ceux-ci sont issus d'analyses d'enregistrements oraux.

Une fois l'étape de recueils des enregistrements sonores achevée, débute alors la phase de transcription et d'annotations du document sonore par le chercheur, souvent aidé d'un informateur.

Cette étape demande un investissement en temps très important pour le chercheur. A titre d'exemple, on peut citer le projet ANR Rhapsodie pour lequel le temps d'annotation des proéminences et des dysfluences était compris entre 20 et 60 fois le temps d'enregistrement [1].

Ainsi, il est donc primordial de fournir aux chercheurs un outil de transcription et d'annotation simple d'utilisation, intuitif, bien adapté à leurs besoins et qui puisse éventuellement automatiser certaines tâches.

De plus, comme les outils d'annotation nécessitent un temps de développement long, il est préférable de réutiliser les outils existants bien maintenus quitte à les adapter aux objectifs de la recherche.

Les outils d'annotation de corpus oraux sont des outils qui permettent la construction et l'analyse de corpus de langue parlée. Ils sont capables de gérer simultanément deux types de ressources : les enregistrements, audio et/ou vidéo (signaux temporels) et les annotations textuelles de ces enregistrements.

Avec l'intérêt grandissant pour l'étude des productions orales et le progrès de la technologie, les outils et les formalismes dédiés à l'annotation de corpus oraux se sont beaucoup développés, à l'origine pour la psycholinguistique, l'étude des différentes modalités de communication, l'analyse phonétique et phonologique, l'étude d'interaction entre plusieurs locuteurs ou encore l'étude de langues à tradition orale et/ou en voie de disparition, puis se sont étendus aux domaines de la dialectologie, de l'acquisition du langage et de l'annotation multilinéaire des textes écrits.

Ils doivent combiner au moins 3 fonctionnalités de base : un outil d'édition des ressources (permettant de les créer et de les modifier), un outil de consultation du support audio et/ou vidéo (un lecteur multimédia), un outil d'interrogation portant sur les annotations et/ou les enregistrements.

Pour répondre aux objectifs linguistiques du projet CorpAfroAs d'étudier des phénomènes faisant appel à plusieurs niveaux d'analyse, il était nécessaire d'avoir un outil qui permette l'annotation sur plusieurs niveaux entretenant entre eux des

1. Wiki du projet Rhapsodie sur les consignes de codage site consulté le 31 janvier 2013

relations hiérarchiques et/ou temporelles. La présence d'un module de recherche est également un critère de choix important pour l'exploitation postérieure des données.

De plus, le découpage du corpus en unités prosodiques requiert un outil qui permette également une visualisation assez fine du spectrogramme afin de pouvoir identifier avec précision les variations de fréquences et établir les frontières des unités.

Enfin, la capacité du logiciel à importer des fichiers d'annotations d'autres formats est également un critère de choix, en particulier le format Toolbox qui est un format largement et depuis longtemps utilisé par les linguistes de terrain du LLACAN. Les archives disponibles dans ce format sont donc nombreuses et il était primordial de pouvoir les exploiter.

Nous nous intéresserons donc plus particulièrement aux logiciels permettant de faire des annotations multilinéaires, capables d'importer d'autres formats et étant dotés d'un module de requêtes. Cependant, nous présenterons aussi quelques outils, plus orientés pour les études phonétiques et prosodiques, qui ne permettent que la gestion combinée du son et des transcriptions mais qui sont souvent plus performants pour la gestion du signal sonore.

Après un bref aperçu du cadre théorique dans lequel se situe la problématique des annotations linguistiques, qui nous sera utile par la suite pour comprendre la structure des fichiers d'annotation (cf 4.3), nous distinguerons dans les deux parties suivantes les groupes d'outils utilisant le formalisme XML et ceux utilisant un formalisme propriétaire, puis nous évoquerons quelques plates-formes regroupant plusieurs outils à la fois.

Enfin nous présenterons un récapitulatif sous forme de tableau qui permettra d'avoir une vision plus synthétique et mettre en relief les caractéristiques des différents outils afin de justifier les choix qui ont été faits.

3.1 Les formalismes sous-jacents

Dans le contexte des corpus annotés, il faut distinguer les conventions de transcription des structures des annotations.

Pour ce qui est des conventions de transcriptions, nous ne ferons ici qu'en citer quelques-unes dans la mesure où elles ne font pas partie de l'objet principal de ce travail. Cependant nous développerons celles qui ont été utilisées dans le cadre de ce projet dans la section 2.3.

Il existe deux types de conventions d'annotation : les conventions de transcription phonétiques et les conventions de transcription orthographique.

Les premières sont généralement basées sur l'API ou son extension le SAMPA.

Les secondes, qui nécessitent donc d'avoir une langue avec un système orthographique, sont quelques fois spécifiées dans le cadre d'un projet particulier pour satisfaire des besoins relatifs à un objet de recherche. Mais il en existe quelques unes largement utilisées dans le domaine de l'analyse conversationnelle parmi lesquelles on peut mentionner les conventions HIAT (*Halbinterpretative Arbeitstranskriptionen*, Jefferson (du nom de son développeur Gail Jefferson), GAT (*Gesprächsanalytisches Transkriptionssystem*, CHAT (*Codes for the Human Analysis of Transcripts*), CA (*Conversation Analysis*)...

En ce qui concerne les structures d'annotations, on distingue traditionnellement deux modèles qui sont :

- le format **OHCO** (*Ordered Hierarchy of Content Objects*). C'est un modèle hiérarchique, arborescent d'organisation des contenus qui, au début des années 90, était considéré comme la meilleure représentation possible d'un document.

Cette idée s'est cependant heurtée aux difficultés rencontrées pour traiter des phénomènes qui se chevauchent, telles que l'existence de différentes hiérarchies au sein d'un même document (par exemple la pagination d'un même texte dans différentes éditions ou encore, plus proche du problème qui nous intéresse, les relations syntaxiques dans un texte découpé prosodiquement).

– le format **STMT** (*Single Timeline, Multiple Tiers*) basé sur la théorie des GRAPHES D'ANNOTATION de [Bird and Liberman, 2001] qui est un formalisme linguistique permettant de représenter toute forme d'annotation en faisant abstraction de son format physique. Il offre ainsi un cadre unificateur pour les annotations linguistiques en séparant le niveau physique du niveau logique.
La représentation se fait à l'aide de graphes acycliques orientés dont les *nœuds* peuvent avoir ou ne pas avoir de positions temporelles (on parle de nœuds « ancrés » ou « non ancrés) et les *arcs*, qui relient deux nœuds, sont étiquetés, c'est-à-dire qu'ils portent des champs d'information de type attribut/valeur.
L'avantage de ce formalisme dans le cas d'annotations appliquées à des signaux est qu'il permet d'encoder des informations sans contrainte sur la segmentation pour l'alignement au signal.
Il a été développé dans le but de fournir un schéma général d'annotations pouvant servir de pivot à d'autres modèles et permettant à ceux-ci d'utiliser les mêmes outils de requête.
Il a d'ailleurs été utilisé lors des tentatives de conversion entre les différents formats d'annotations multimodales [Schmidt et al., 2009].

3.2 Outils utilisant le format XML

Ils constituent la majorité des outils disponibles de part le statut de standard qu'a acquis XML pour la structuration des documents.

Anvil *Annotation of Video and Language Data* de Michael Kipp, développé à l'université de Sarre en Allemagne, c'est un outil qui permet l'annotation multilinéaire (sous forme de lignes parallèles semblables à des portées musicales) de vidéos (au format Quicktime ou Avi) pour l'étude des gestes, postures et informations relatives au discours. Il est particulièrement utilisé dans les domaines de l'interaction homme-machine, la psychologie, l'ethnologie et même dans le domaine de la danse.

Il possède une interface ergonomique pour l'alignement temporel et est très flexible quant à la configuration des niveaux d'annotations et de leurs relations.

Il est également doté d'un module pour le calcul du coefficient Kappa de Cohen (indice qui quantifie l'accord inter-annotateur).

ELAN *EUDICO Linguistic Annotator* du projet DOBES (documentation sur les langues rares) développé au département de psycholinguistique du Max Planck Institute à Nigmegen aux Pays-Bas. Il a hérité du développement de la boîte à outils EUDICO (*European Distributed Corpora Project*).

Cet outil, particulièrement bien adapté à l'analyse des langues, est surtout utilisé dans l'étude du langage des signes et chez les linguistes de terrain.

Il permet l'annotation multilinéaire d'enregistrements audio et/ou vidéo et l'adaptation des niveaux et hiérarchies d'annotations aux besoins des utilisateurs.

Chaque projet ELAN est constitué d'au moins un fichier son ou vidéo accompagné de son fichier d'annotations.

Le modèle des graphes d'annotations se résume, ici, en une ligne temporelle à laquelle se réfère des annotations organisées en niveaux qui sont associés à un type d'information et à un unique participant (ou *speaker*). Les annotations sur un niveau sont adjacentes et ne peuvent se chevaucher.

EXMARaLDA Partitur-Editor *Extensible Markup Language for Discourse Annotation* de Thomas Schmidt, développé dans un centre de recherche collaboratif sur le multilinguisme, le SFB 538 Mehrsprachigkeit à l'université de Hamburg.

A l'origine, ce système avait été conçu pour permettre d'avoir un cadre commun utilisable par les différents projets du centre dans un souci de partage et d'échange des documents et archives.

Il est utilisé essentiellement dans les domaines de l'analyse conversationnelle, de l'analyse du discours, de l'apprentissage des langues ou encore de la dialectologie.

Il regroupe plusieurs outils et possède ainsi un outil d'édition d'annotations suivant le modèle des partitions musicales (les comportements non-verbaux sont considérés comme des annotations des énoncés verbaux), un outil de gestion des métadonnées et un outil de requêtes (KWIC searches). Il est écrit en Java ce qui en fait un outil multiplateforme.

Son modèle de données repose sur la théorie du graphe d'annotation de [Bird and Liberman, 2001] mais sa structure est tout de même moins complexe que ne l'est ce dernier.

Cet outil est utilisé dans le domaine d'étude des interactions verbales entre de nombreux locuteurs, de l'acquisition du langage et de la dialectologie.

C'est l'outil le plus proche d'ELAN au niveau de ses fonctionnalités. Il est interopérable avec ce dernier ainsi qu'avec Praat.

FOLKER Cet outil est beaucoup moins puissant qu'ELAN mais il est plus facile d'accès à des non-spécialistes. Il ne permet pas d'annotations multicouches. Il est surtout utilisé pour l'analyse des conversations et ses fonctionnalités se rapprochent de celles de Transcriber.

ITE (*Interlinear Text Editor*), génère un document XML avec la DTD du LACITO. Il permet la saisie de texte interlinéaire sur 4 niveaux (le texte, la phrase, le mot et le morphème) et entretient une relation entre les documents ouverts et un lexique enrichi au fur et à mesure de l'annotation ce qui facilite le travail de saisie. Cependant la mise à jour de ce logiciel n'est plus assurée.

MMAX Cet outil utilise comme entité abstraite le `markable` qui est l'élément de base de stockage de l'information des fichiers XML d'annotations. Un tel élément représente une liste d'ID de mots et/ou gestes (pointant vers ces entités) constituée selon les phénomènes que l'on désire étudier. Typiquement, il peut correspondre à l'ensemble des IDs de mots et gestes d'une phrase ou d'un tour de parole.

Chaque `markable` est constitué d'un ensemble de traits caractéristiques d'un niveau d'annotation. Ces traits sont soit des paires attribut-valeur (ex : pos=det) ou des relations entre `markables` (ex : relation de coréférence).

Il existe un langage de requête, MMAXQL, associé à cet outil et qui permet de formuler des interrogations sur plusieurs niveaux d'annotation. Les requêtes sont lancées depuis une console accessible par l'interface de l'outil et sont traduites en langage XQuery.

La structure d'annotation produite par cet outil est proche de celle des graphes d'annotation et la conversion d'un modèle à l'autre est donc possible.

SoundIndex (développé par Michel Jacobson tout comme ITE) est un outil qui allie un éditeur de texte XML et un éditeur de son. Il permet d'établir la correspondance entre la transcription d'un texte et son enregistrement sonore en ajoutant des balises <AUDIO> à n'importe quel niveau de l'arborescence d'un fichier XML. Il est cependant plus adapté à l'analyse acoustique du discours qu'à l'annotation de corpus à proprement parler.

TranscriberAG dont l'ancêtre est Transcriber, était conçu à l'origine pour la transcription des journaux télévisés. Les fichiers d'annotation ont l'extension .trs qui correspond à une structure XML dont l'élément racine s'appelle `Trans`. Ils utilisent le modèle des graphes d'annotation.

FOLKER et Transcriber ont été développés pour un type particulier de discours, les dialogues radio ou télédiffusés. FOLKER est cependant un peu plus performant pour la gestion des phénomènes de chevauchements entre plusieurs locuteurs.

3.3 Outils utilisant un formalisme propriétaire ou un format préexistant

CLAN (*Computerized Language Analysis*) est l'outil associé au projet CHILDES (*Child Language Data Exchange System*) pour l'analyse des situations de dialogues chez l'enfant. Il permet la manipulation des données contenues dans la base de données de ce projet.

Cet outil utilise les conventions de transcription CHAT et CA qui sont alignées sur des enregistrement audio ou vidéo.

Le logiciel PHON qui lui est associé permet l'exploration phonologique des transcriptions de ce format, il est plus orienté à l'usage des phonologues et est également doté d'un outil de recherche, Phonex.

La structure sous-jacente des données est OHCO ce qui les rend facilement transformables en XML.

MTrans Cet outil permet également la transcription de conversations multipartites avec une bonne gestion des chevauchements entre interlocuteurs. Les fichiers produits sont au format txt et la possibilité d'un rendu en XML est en cours de développement.

Praat développé par l'univerisé de Hambourg, permet le traitement et l'analyse du son et est très répandu dans la communauté des phonéticiens. Il possède également une fonctionnalité permettant l'insertion d'annotations sauvegardées au format Textgrid, qui est un format texte brut dans lequel les intervalles sont décrits par niveau d'annotation.

La structure sous-jacente des données est STMT.

Il a également un plugin, EasyAlign, qui permet de créer une annotation multilinéaire de façon semi-automatique à partir d'un document audio et de sa transcription orthographique.

Il a le grand avantage de permettre une visualisation précise du spectre des fréquences

Winpitch Tout comme Praat, il est adapté à l'analyse de la parole, de la prosodie et permet aussi l'annotation sur plusieurs niveaux indépendants les uns des autres.

Cependant, il n'est pas maintenu et la multitude des fonctionnalités offertes en fait un outil peu ergonomique.

Transana développé par l'université de Wisconsin-Madison, permet de taper du texte libre (comme dans un logiciel de traitement de texte) ce qui rend son approche très simple pour l'utilisateur mais son interopérabilité avec d'autres outils plus sophistiqués quasiment impossible dans la mesure où aucune information structurelle n'est fournie. Il utilise le standard d'annotation « Jeffersonian Transcription Annotation ».

Toolbox Développé par la SIL (*Summer Institure of Linguistics*), c'est un outil de gestion et d'analyse de données très populaire dans la communauté des linguistes de terrain. Il est utilisé en particulier pour saisir et stocker des données lexicales ainsi que pour la saisie de texte interlinéaire.

C'est donc une base de données textuelles offrant toutes les fonctionnalités nécessaires à la création de dictionnaires publiables mais permettant également à l'utilisateur de personnaliser les champs des tables suivant ses besoins spécifiques.

De plus, il possède un module d'interalignement qui permet d'annoter un texte sur la base du lexique préalablement entré et sur un segmenteur morphologique qui découpe les unités lexicales en morphèmes suivant les décompositions déjà fournies par l'utilisateur.

Les fichiers générés ont une extension .typ.

XTrans est un logiciel développé par le LDC (*Linguistic Data Consortium*). Il est surtout adpaté à la transcription de conversations téléphoniques, de réunions. Il permet soit d'associer une transcription existante à un enregistrement soit de faire directement la transcription aligné au son.

Il a l'avantage de permettre la mise en relief des portions de chevauchement des prises de parole sur le spectrogramme.

NWB3 *Nite Workbench for Windows*, développé dans le cadre du projet NITE (*Natural Interactivity Tools Engineering*) est également basé sur une structure de base de données relationnelles. Il a l'inconvénient de ne fonctionner que sur Windows.

The Observer , aussi développé dans le cadre du projet NITE, est un logiciel payant d'annotation vidéo développé pour l'étude des comportements. Il présente l'inconvénient de ne pas offrir de vue alignée au temps pour la saisie des annotations.

Les données sont sauvegardées dans des bases de données relationnelles.

Tout comme pour NWB3, il ne fonctionne que sur Windows.

On peut également citer l'outil **iLex** utilisé exclusivement pour l'annotation du langage des signes qui utilise une architecture serveur-client et dont l'installation un peu compliquée limite les usagers aux gros centres de recherche sur le langage des signes.

Telemata, développé par le Centre de recherche en ethnomusicologie. C'est une application web qui permet de consulter, éditer, télécharger et uploader des fonds d'archive sonore avec leurs métadonnées.

3.4 Les plateformes

Il existe également des plateformes regroupant plusieurs outils et permettant d'effectuer toutes les tâches citées ci-dessus.

La plateforme **IrcamCorpusTools** intègre quant à elle tout un éventail d'outils qui permettent la création, l'analyse et l'exploitation d'un corpus de parole. Elle utilise l'environnement de programmation Matlab/Octave.

La plateforme **Dolmen** (anciennement PFC) lit les fichiers d'annotation de Praat (TextGrid) et peut les convertir dans son format natif (DMF). Les fichiers sons reconnus sont les .wav, .flac et .aiff. Elle permet d'organiser des corpus, de spécifier des métadonnées et d'effectuer des requêtes, en particulier sur les métadonnées. Elle nécessite l'installation de l'interpréteur Python.

Le logiciel **AGTK** (*Annotation Graph Toolkit*) permet de construire des logiciels d'aide à l'annotation des données audio et vidéo basées sur le modèle des graphes d'annotation. Le logiciel comprend déjà des interfaces pour la création de texte interaligné au son, pour la création et la manipulation d'arbres syntaxiques et pour la transcription d'interactions entre plusieurs participants.

Tous les outils développés à partir d'AGTK ont une architecture à trois niveaux : un niveau physique (spécifiant les formats de sauvegarde et de codage des données), un niveau applicatif (renfermant l'outil) et un niveau logique intermédiaire entre les deux précédents (reposant sur le formalisme des graphes d'annotations capable de représenter toutes les informations contenues dans les annotations).

La collection d'outils regroupés sous le nom de **EMU Speech Database System** développé par l'Institute of Phonetics and Speech Processing de l'université de Munich permet la creation, la manipulation et l'analyse de corpus oraux. Elle est utilisée en particulier dans le domaine de la phonétique acoustique. Les outils peuvent importer des données au format Praat.

La plateforme **MATE** (*Multi-level Annotation Tool Engineering*) n'utilise que le format XML pour les données en entrée et sortie. Cependant, elle traite les problèmes de chevauchements de paroles ou bien de superpositions de hiérarchies d'annotations à utilisant des pointeurs qui renvoient à d'autres éléments, évitant ainsi les chevauchements de balises.

Elle est dotée d'un language de requête, Q4M, et permet à l'utilisateur de personnaliser l'affichage des données par l'intermédiaire de feuilles de style XSLT.

Nous citerons également **EOPAS** (*Ethnographic E-Research Online Presentation System*) qui est un outil ne permettant pas l'annotation mais la consultation de fichiers d'annotations ainsi que la recherche dans ces fichiers.

Largement inspiré de ITE, EOPAS a été développé dans le cadre du projet EthnoER (« Sharing access and analytical tools for ethnographic digital media using high speed networks ») de l'université de Melbourne. Il avait comme objectif principal de rendre les données linguistiques plus facilement accessibles à tout le monde, aussi bien aux chercheurs qu'aux non-spécialistes.

L'outil est utilisable en ligne ou bien téléchargeable.

Pour l'outil en ligne, l'utilisateur peut consulter les corpus accessibles et même téléverser ses propres corpus après s'être enregistré et à condition que ses fichiers d'annotation soient au format ELAN, Transcriber ou Toolbox, qu'ils aient des données temporelles et que ses fichiers multimedia soient dans l'un des formats suivants : wav, mp3, mov.

Il est ensuite possible d'exporter les fichiers d'annotations dans les formats précités ainsi qu'en PDF, RDF ou encore au format EOPAS qui est de type OHCO basé sur de l'XML.

On pourrait encore mentionner un bon nombre d'outils permettant la transcription et l'annotation de corpus oraux multimodaux (avec son et/ou video) mais nous ne nous y intéresserons pas dans cet état de l'art car ils ne permettent pas d'annotation sous-phrastique (tels que **DialogueTool** (Hardy et all., 2003) de l'université d'Albany ou encore **DAT** de l'université de Rochester). Ces outils sont plutôt dédiés à l'étude des phénomènes de communication non-verbale (langage des signes ou

annotations des gestes et expressions corporelles).

Il existe un wiki recensant tous les outils et les formats permettant la création et la gestion des annotations linguistiques : http://www.annotation.exmaralda.org/ qui a remplacé la page du LDC (*Linguistic Data Consortium*) de l'université de Pennsylvanie qui n'est cependant plus maintenue : http://www.ldc.upenn.edu/annotation/.

Le projet EMELD (*Electronic Metastructure for Endangered Languages Data*) de la Linguist list offre également un inventaire des logiciels disponibles et maintenu par les utilisateurs : http://www.emeld.org/school/index.html.

3.5 Tableaux récapitulatifs

Les critères à privilégier dans le choix d'un outil de transcription et d'annotation sont, au delà de l'ergonomie et de la facilité de manipulation, la possibilité d'importer et d'exporter d'autres formats, l'assurance d'un maintien du code du logiciel et d'un support technique ainsi que la possibilité d'accès à un large corpus.

Voici des tableaux récapitulatifs des outils cités qui permettent une comparaison rapide suivant une liste de paramètres relatifs aux critères cités ci-dessus ainsi qu'au type et à la structure des données [2].

2. Pour le paramètre « multilinéraire », on parle aussi de modèles « tier-based » « non-tier-based »

		OUTILS					
		Anvil(17)	CLAN(19)	Exmaralda(18)	Elan(17)	ITE(18)	SoundIndex(19)
DONNEES	Multilinéaire	oui	non	oui	oui	oui	oui
	Hiérarchie des annotations	oui	non	non	oui	oui	oui
	Format physique	XML	XML	XML	XML	XML	XML
	Gestion Métadonnées	limitée	limitée	oui	oui	non	non
	Fonction de recherche	oui	simple	oui	oui	oui	oui
	Annotation semi-automatique	non	non	non	oui	oui	oui
INTEROPERABILITE	Import de	praat, elan	.typ	praat, elan, tei	toolbox, chat, transcriber, elan	/	/
	Export vers	csv, spss, arff	.typ, .stm	praat, elan, .txt, chat, tei	csv, toolbox, chat...	/	/
LOGICIEL	Open source	non	oui	non	oui	oui	oui
	Maintenu	oui	oui	oui	oui	non	non

TABLE 3.1 – Récapitulatif des outils d'annotations

		OUTILS				
		Pratt(19)	Toolbox(20)	Transana(20)	Transcriber(19)	Folker(18)
DONNEES	Multilinéaire	oui	oui	non	non	non
	Hiérarchie des annotations	oui	oui	non	non	non
	Format Physique	textgrid	SFT (Standard Format Text)	RFT	XML	XML
	Gestion Métadonnées	non	oui	non	limitée	non
	Fonction de recherche	non	oui	oui	oui	non
	Annotation semi-automatique	oui (pour la prosodie et avec plug-in EasyAlign)	oui	oui	non	non
INTEROPERABILITE	Import de	/	txt	/	.typ	/
	Export vers	/	xml, RFT	xml	.stm, childes	elan, exmaralda
VISUALISATION	Open source	oui	non	non	oui	oui
	Maintenu	oui	dernière MàJ : 2011	oui	oui	oui

TABLE 3.2 – Récapitulatif des outils d'annotations

		OUTILS				
		MMAX(18)	WinPitch(19)	MTrans(19)	XTrans(20)	NWB3(20)
DONNEES	Multilinéaire	oui	non	oui	non	oui
	Hiérarchie des annotations	non	non	non	non	oui
	Format physique	XML	database	txt	tdf (tab delimited)	database
	Métadonnées	oui	non	non	oui	oui
	Fonction de recherche	oui	non	oui	non	oui
	Semi-automatique	non	oui	non	non	non
INTEROPERABILITE	Import de	xml	transcriber, xsl	praat	transcriber	/
	Export vers	xml	xml, xsl	xml, praat	transcriber, txt	xml
VISUALISATION	Open source	oui	non	oui	oui	oui
	Maintenu	non	non	oui	dernière MàJ : 2011	non

TABLE 3.3 – Récapitulatif des outils d'annotations

Au vu des tableaux ci-dessus, il apparaît que le logiciel qui répond le mieux aux contraintes imposées par le projet est ELAN.

Au delà des points évoqués dans les tableaux, les autres avantages que nous pouvons retenir pour le projet CorpAfroAs sont les suivants :

- les niveaux de description sont personnalisables suivant l'analyse que désire faire le chercheur,
- des tiers peuvent être ajoutés même une fois le processus d'annotation commencé,
- l'utilisation possible d'Unicode permet de transcrire l'alphabet phonétique,
- les relations hiérarchiques entre les tiers sont clairement représentées.

Cependant sa gestion du son n'est pas aussi précise qu'on le souhaiterait ce qui a orienté le choix final vers l'utilisation supplémentaire, en amont, du logiciel Praat pour la segmentation prosodique (cf 2.3.1).

Chapitre 4

Présentation de l'outil d'annotation choisi : ELAN

Comme l'indique l'état de l'art, le logiciel qui semble le plus adapté aux objectifs poursuivis par le projet au niveau de l'annotation est le logiciel ELAN. La tâche de transcription est, elle, assurée par le logiciel Praat, d'une part car c'est un outil mutualisé et très répandu dans la communauté des phonéticiens et d'autre part car il possède nativement de nombreux scripts pour automatiser des tâches et est facilement modularisable par l'ajout d'autres scripts. Cependant, il est à noter que le découpage peut également se faire directement dans ELAN, avec, toutefois, l'inconvénient d'une moins grande précision dans la segmentation.

Nous allons donner, dans cette partie, une brève présentation du logiciel ELAN.

4.1 Module d'interalignement

Si les tâches récurrentes de la linguistique textuelle que sont la segmentation et l'annotation morphologique et syntaxique (avec des outils de type parseur, lématiseur, tagger...) sont de plus en plus automatisées ou semi-automatisées pour les langues bien documentées, ceci est loin d'être le cas pour les langues rares et/ou peu dotées qui font l'objet d'étude de ce projet.

Au mieux, ces langues disposent d'un corpus de quelques heures d'enregistrement accompagné d'une description morpho-phonologique, quelquefois d'un lexique ou d'un dictionnaire de quelques milliers de mots et d'une ébauche de grammaire.

Nous avons vu précédemment dans l'état de l'art que le logiciel ELAN présentait plusieurs avantages pour le type d'annotations envisagées dans ce projet. Cependant, nativement, ELAN ne possède aucun outil d'aide à l'annotation.

Ainsi, un module supplémentaire permettant d'ajouter du texte interlinéaire de façon semi-automatique a été développé dans le cadre du projet CorpAfroAs afin de faciliter le travail de l'annotateur, ce qui a donné naissance au logiciel dérivé **ELAN-CorpA**.

Le module, appelé « Interlinearize », permet que le découpage en morphèmes se fasse de façon semi-automatique.

Le processus d'interalignement se base sur l'utilisation d'un lexique précédemment créé dans ELAN ou bien importé de Toolbox dans ELAN (fichier avec l'extension *eafl*). Ce lexique est au format XML et contient une liste de lemmes et d'affixes accompagnés de leurs gloses, de leurs éventuelles variantes et formes sous-jacentes. Le logiciel enrichit ce lexique au fur et à mesure des gloses saisies. L'avantage de ce processus, en plus du gain de temps, est de permettre au linguiste de rester cohérent dans le choix de ses gloses tout au long de ses annotations.

Il existe une autre sorte de lexique, « parse lexicon » (fichier avec l'extention *eafp*, également au format XML), que l'on peut générer lorsque tout un fichier a été annoté.

Ce lexique contient la liste de tous les mots du texte avec leurs segmentations et les gloses associées. Il permet d'accélérer le processus d'auto-interalignement sur les nouveaux fichiers. Il est possible de fusionner plusieurs « parse lexicon ».

Ainsi l'objectif principal n'est pas tant l'automatisation complète que l'aide à l'annotation pour le chercheur en particulier en vue d'une homogénéisation des étiquettes utilisées.

4.2 Types linguistiques des couches

Chaque interlocuteur est associé au même ensemble de couches prédéfini suivant les besoins d'analyse du corpus. La gestion du chevauchement de paroles ne pose donc pas de difficulté pour la transcription.

Chaque couche doit être définie avec un type linguistique qui informe le logiciel sur la nature de l'information qu'elle va contenir.

Il existe 5 types linguistique principaux (extensibles par l'utilisateur suivant les besoins) :

- `None` : l'annotation sur cette couche est alors directement liée au temps, comme par exemple la couche `ref`.
- `Time Subdivision` : les annotations de cette couche sont liées au temps par l'intermédiaire de la couche parent, qui doit être liée au temps et qui lui fait hériter des ses indications de temporelles. Chaque annotation de cette couche aura la même durée (soit la durée de la couche parent divisée par le nombre d'enfants) et tous les enfants d'un même parent seront contigus (ex : la couche `mot`). Ce type permet de caler les subdivisions de l'enfant sur la ligne du temps en respectant les bornes externes du parent.
- `Included In` : ce sont les mêmes caractéristiques que pour le type précédent mais les enfants ne sont pas nécessairement contigus, il peut y avoir des vides entre eux.
- `Symbolic Subdivision` : ce type est semblable à la `Time Subdivision` à ceci près que les annotations de ce type ne peuvent être liées au temps, en d'autres termes alignables avec la ligne temporelle (ex : les morphèmes de la couche `mb`).
- `Symbolic Association` : il s'agit d'une relation de correspondance exacte entre le parent et l'enfant (ex : la couche `tx` par rapport à la couche `ref` ou encore `rx` par rapport à `ge`). Pas de subdivision possible.

Tous ces types de couches permettent de rendre compte des relations hiérarchiques et/ou séquentielles qui peuvent exister entre les différents niveaux d'annotation. A partir d'un seul niveau synchronisé avec le signal sonore, les relations hiérarchiques permettent de propager les marques temporelles entre niveaux d'annotation.

Bien que la représentation visuelle de l'annotation soit de type texte interlinéaire, la représentation XML interne des données ne reflète pas ce modèle.

Dans la partie suivante, nous allons étudier plus en détail la structure d'un fichier ELAN.

4.3 Structure XML d'un fichier ELAN

La structure d'un document XML peut s'exprimer sous forme de DTD ou de XML-Schema. La syntaxe formelle qui y est décrite devrait refléter l'analyse que l'on souhaite faire des données.

Ainsi pour les fichiers ELAN, on pourrait imaginer une structure hiérarchique à plusieurs niveaux dans laquelle chaque niveau contiendrait un ou plusieurs items du niveau inférieur (c'est le cas du format développé au LACITO par Michel Jacobson, mettre la ref de talJacobson2.0). On pourrait donc penser que les éléments de la couche `mot` seraient enfants de ceux de la couche `tx` et parents de ceux de la couche `mb` comme le montre le schéma suivant :

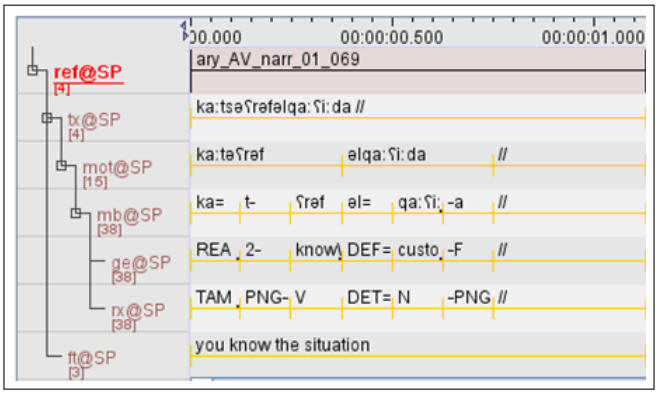


FIGURE 4.1 – Hiérarchie des couches dans un fichier ELAN - Exemple du corpus de l'arabe Marocain

Cependant, ce n'est pas ainsi qu'est organisé un fichier ELAN.

Au contraire nous avons plutôt à faire à une structure « plate » où chaque annotation d'une couche est enfant d'un élément `TIER` (par exemple tous les tx appartiennent à l'élément `TIER` dont l'attribut `LINGUISTIC_TYPE` est `tx`) et possède un attribut `ID` qui permet d'y faire référence au niveau de l'attribut `ANNOTATION_REF` (par exemple, une annotation `mb` aura comme valeur d'attribut `ANNOTATION_REF` la valeur de l'attribut `ID` de l'annotation `mot` dont elle est un morphème).

Voici pour illustration, un exemple de fichier ELAN (BEJ_MV_NARR_01_shelter.eaf) dans lequel nous n'avons gardé que les pointeurs de temps et annotations relatives à la première unité de découpage (soit la ref dont l'identifiant est « BEJ_MV_NARR_01_shelter_071 ») :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ANNOTATION_DOCUMENT AUTHOR="unspecified" DATE="2011-07-29T14:05:28+01:00" FORMAT="2.6"
   VERSION="2.6" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="http://www.mpi.nl/tools/elan/EAFv2.6.xsd">
3      <HEADER MEDIA_FILE="" TIME_UNITS="milliseconds">
4          <MEDIA_DESCRIPTOR MEDIA_URL="file://.../BEJ_MV_NARR_01_shelter.wav"
           MIME_TYPE="audio/x-wav"
           RELATIVE_MEDIA_URL="file:/.../BEJ_MV_NARR_01_shelter.wav"/>
5          <PROPERTY NAME="lastUsedAnnotationId">2541</PROPERTY>
6      </HEADER>
7      <!--========= Definition des pointeurs temporels ======== -->
8      <TIME_ORDER>
9      .
10     .
11       <TIME_SLOT TIME_SLOT_ID="ts190" TIME_VALUE="65384"/>
12       <TIME_SLOT TIME_SLOT_ID="ts191" TIME_VALUE="65384"/>
13       <TIME_SLOT TIME_SLOT_ID="ts192" TIME_VALUE="65384"/>
14       <TIME_SLOT TIME_SLOT_ID="ts193" TIME_VALUE="65969"/>
15     .
16     .
17     </TIME_ORDER>
```

```
18    <!--============== Tier "ref" ============= -->
19    <TIER DEFAULT_LOCALE="us" LINGUISTIC_TYPE_REF="ref" PARTICIPANT="SP" TIER_ID="ref@SP">
20      .
21      .
22        <ANNOTATION>
23            <ALIGNABLE_ANNOTATION ANNOTATION_ID="ann1108" TIME_SLOT_REF1="ts190"
              TIME_SLOT_REF2="ts193">
24                <ANNOTATION_VALUE>BEJ_MV_NARR_01_shelter_071</ANNOTATION_VALUE>
25            </ALIGNABLE_ANNOTATION>
26        </ANNOTATION>
27      .
28      .
29      .
30    </TIER>
31    <!--============== Tier "tx" ============= -->
32    <TIER DEFAULT_LOCALE="fr" LINGUISTIC_TYPE_REF="tx" PARENT_REF="ref@SP"
      PARTICIPANT="SP" TIER_ID="tx@SP">
33      .
34      .
35        <ANNOTATION>
36            <REF_ANNOTATION ANNOTATION_ID="ann1113" ANNOTATION_REF="ann1108">
37                <ANNOTATION_VALUE>’u:nu:ta /</ANNOTATION_VALUE>
38            </REF_ANNOTATION>
39        </ANNOTATION>
40      .
41      .
42    </TIER>
43    <!--============== Tier "mot" ============= -->
44    <!-- contient 3 enfants "u:n", "u:tak" et "/" -->
45    <TIER DEFAULT_LOCALE="fr" LINGUISTIC_TYPE_REF="mot" PARENT_REF="tx@SP"
      PARTICIPANT="SP" TIER_ID="mot@SP">
46      .
47      .
48        <ANNOTATION>
49            <REF_ANNOTATION ANNOTATION_ID="a294" ANNOTATION_REF="ann1113">
50                <ANNOTATION_VALUE>u:n</ANNOTATION_VALUE>
51            </REF_ANNOTATION>
52        </ANNOTATION>
53        <ANNOTATION>
54            <REF_ANNOTATION ANNOTATION_ID="a295" ANNOTATION_REF="ann1113"
              PREVIOUS_ANNOTATION="a294">
55                <ANNOTATION_VALUE>u:tak</ANNOTATION_VALUE>
56            </REF_ANNOTATION>
57        </ANNOTATION>
58        <ANNOTATION>
59            <REF_ANNOTATION ANNOTATION_ID="a296" ANNOTATION_REF="ann1113"
              PREVIOUS_ANNOTATION="a295">
60                <ANNOTATION_VALUE>/</ANNOTATION_VALUE>
61            </REF_ANNOTATION>
62        </ANNOTATION>
63      .
64      .
65    </TIER>
66    <!--============== Tier "mb" ============= -->
67    <!-- le 2nd mot (dont l’id est a295) est constitue de 2 morphemes : "u:=" et "tak" -->
68    <TIER DEFAULT_LOCALE="fr" LINGUISTIC_TYPE_REF="mb" PARENT_REF="mot@SP"
      PARTICIPANT="SP" TIER_ID="mb@SP">
69      .
70      .
71        <ANNOTATION>
72            <REF_ANNOTATION ANNOTATION_ID="a807" ANNOTATION_REF="a294">
73                <ANNOTATION_VALUE>u:n</ANNOTATION_VALUE>
74            </REF_ANNOTATION>
75        </ANNOTATION>
76        <ANNOTATION>
77            <REF_ANNOTATION ANNOTATION_ID="a808" ANNOTATION_REF="a295">
78                <ANNOTATION_VALUE>u:=</ANNOTATION_VALUE>
79            </REF_ANNOTATION>
80        </ANNOTATION>
81        <ANNOTATION>
82            <REF_ANNOTATION ANNOTATION_ID="a809" ANNOTATION_REF="a295"
              PREVIOUS_ANNOTATION="a808">
83                <ANNOTATION_VALUE>tak</ANNOTATION_VALUE>
```

```
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a810" ANNOTATION_REF="a296">
                <ANNOTATION_VALUE>/</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
      .
      .
    </TIER>
    <!--============== Tier "ge" ============= -->
    <TIER DEFAULT_LOCALE="fr" LINGUISTIC_TYPE_REF="ge" PARENT_REF="mb@SP"
    PARTICIPANT="SP" TIER_ID="ge@SP">
      .
      .
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a1469" ANNOTATION_REF="a807">
                <ANNOTATION_VALUE>PROX.SG.M.NOM</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a1470" ANNOTATION_REF="a808">
                <ANNOTATION_VALUE>DEF.SG.M.NOM=</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a1471" ANNOTATION_REF="a809">
                <ANNOTATION_VALUE>man</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a1472" ANNOTATION_REF="a810">
                <ANNOTATION_VALUE>.</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
      .
      .
    </TIER>
    <!--============== Tier "rx" ============= -->
    <TIER DEFAULT_LOCALE="fr" LINGUISTIC_TYPE_REF="rx" PARENT_REF="mb@SP"
    PARTICIPANT="SP" TIER_ID="rx@SP">
      .
      .
        <ANNOTATION>
          <!-- on voit par la valeur de l'attribut "annotation_ref" que rx est fils de mb
          et non de ge -->
            <REF_ANNOTATION ANNOTATION_ID="a2131" ANNOTATION_REF="a807">
                <ANNOTATION_VALUE>DEM</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a2132" ANNOTATION_REF="a808">
                <ANNOTATION_VALUE>DET=</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a2133" ANNOTATION_REF="a809">
                <ANNOTATION_VALUE>SBJ.N.M</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="a2134" ANNOTATION_REF="a810">
                <ANNOTATION_VALUE>.</ANNOTATION_VALUE>
            </REF_ANNOTATION>
        </ANNOTATION>
      .
      .
    </TIER>
    <!--============== Tier "ft" ============= -->
    <TIER DEFAULT_LOCALE="us" LINGUISTIC_TYPE_REF="ft" PARENT_REF="ref@SP"
    PARTICIPANT="SP" TIER_ID="ft@SP">
        <ANNOTATION>
            <REF_ANNOTATION ANNOTATION_ID="ann1114" ANNOTATION_REF="ann1108">
```

```
153                 <ANNOTATION_VALUE>that man</ANNOTATION_VALUE>
154               </REF_ANNOTATION>
155             </ANNOTATION>
156           .
157           .
158           .
159         </TIER>
160 </ANNOTATION_DOCUMENT>
```

Il s'agit là d'un modèle qui s'inspire du modèle abstrait des graphes d'annotations (cf 3.1).

Il distingue les annotations ancrées dans le temps des annotations flottantes. La structure temporelle est décrite dans un premier bloc (entre les balises `TIME_ORDER`) puis les annotations linguistiques correspondant aux différents nœuds de cette structure temporelle sont décrites avec un système d'indentifiants et de pointeurs sur ces identifiants.

4.4 Stockage des données

Comme nous l'avons vu dans la première partie de notre état de l'art (cf 3.1), il existe deux types principaux et bien connus de représentation physique des données : un qui privilégie la structure temporelle (« time-based model ») et l'autre qui privilégie la structure hiérarchique (« hierarchical model »).

La complémentarité qui existe entre ces deux modèles rend le choix du stockage non trivial.

Tout comme pour le choix de l'unité de découpage et de la finesse de l'annotation, le choix du formalisme pour le stockage des données se devait d'être adapté aux types de recherches envisagées afin de faciliter l'expression des requêtes et réduire le temps de réponse. Du fait de la dimension orale du corpus, il était aussi nécessaire de conserver un ancrage temporel qui permette de se reporter aux données d'origine en réécoutant un segment bien précis.

Le logiciel ELAN produit des fichiers d'annotations au format XML, un simple stockage des données dans ce format aurait donc été possible. Cependant l'interrogation des données par un langage tel que XPath ou Xquery peut s'avérer très chronophage et également complexe à exprimer, en particulier dans notre cas où la structure non hiérarchique du fichier (cf 4.3) rend complexes les requêtes sur plusieurs niveaux d'annotation imbriqués. Les fichiers eaf ne se prêtent donc pas réellement à ce genre d'interrogation.

ELAN, pour son module de recherche interne, génère des bases de données relationnelles SQL qui permettent de stocker temporairement les données relatives aux fichiers sur lesquels portent les requêtes.

Le système utilisé pour ce module est HSQLDB (*HyperSQL Data Base*) qui est un système de gestion de base de données relationnelles écrit en Java.

Toutes les données relatives à un ou plusieurs fichiers ELAN sont stockées dans deux tables principales appartenant à un schéma nommé `search` d'une base de données appelée *Corpafas* :

- la table `annotations` : elle contient les informations relatives à toutes les annotations, quelle que soit leur couche. Le champ clé est appelé `ann_id` et attribue à chaque annotation un identifiant numérique unique.
- la table `tiers` : elle contient les informations relatives à chaque couche. Le champ clé est appelé `tier_id` et permet d'identifier de façon unique une couche donnée.

Il existe également une troisième table, `prog_data`, qui est une table outil permettant de stocker les indices des derniers `ann_id` et `tier_id`.

	ann_id integer	annotation text	ann_position integer	begin_time integer	end_time integer	ref_ann_id integer	aligned boolean	ann_tier_id integer
1	0	BEJ_MV_NARR_10_rabbit_01	0	0	1766	-1	TRUE	0
2	1	BEJ_MV_NARR_10_rabbit_02	1	1766	2105	-1	TRUE	0
3	2	BEJ_MV_NARR_10_rabbit_03	2	2105	3614	-1	TRUE	0
4	3	BEJ_MV_NARR_10_rabbit_04	3	3614	4456	-1	TRUE	0
5	4	BEJ_MV_NARR_10_rabbit_05	4	4456	4877	-1	TRUE	0
6	5	BEJ_MV_NARR_10_rabbit_06	5	4877	5637	-1	TRUE	0

FIGURE 4.2 – Extrait de la table `annotations` pour le fichier BEJ_MV_NARR_10_Rabbit.eaf

	tier_id integer	tier_name text	tier_type text	default_lo text	participan text	n_annotati integer	ref_tier_id integer	transcription_type integer	node_id text
1	0	ref@SP	ref	us	SP	75	-1	0	11
2	1	tx@SP	tx	us	SP	75	0	0	11
3	2	mot@SP	mot	fr	SP	195	1	0	11
4	3	mb@SP	mb	fr	SP	278	2	0	11
5	4	ge@SP	ge	fr	SP	278	3	0	11
6	5	rx@SP	rx	fr	SP	278	3	0	11
7	6	ft@SP	ft	us	SP	42	0	0	11
8	7	Mft	ref	fr	unknown	30	-1	0	11
9	8	ref@SP	ref	us	SP	70	-1	0	12
10	9	tx@SP	tx	us	SP	70	8	0	12
11	10	mot@SP	mot	fr	SP	181	9	0	12
12	11	mb@SP	mb	fr	SP	263	10	0	12
13	12	ge@SP	ge	fr	SP	263	11	0	12
14	13	rx@SP	rx	fr	SP	263	11	0	12
15	14	ft@SP	ft	us	SP	39	8	0	12
16	15	Mft	ref	fr	unknown	31	-1	0	12

FIGURE 4.3 – Extrait de la table `tiers` pour les fichiers BEJ_MV_NARR_10_Rabbit.eaf et BEJ_MV_NARR_09_Jewel.eaf

Cette table n'est pas utilisée lors des requêtes mais seulement lorsque de nouveaux fichiers sont ajoutés au domaine de recherche et que les informations qu'ils contiennent doivent être ajoutées aux deux tables précédentes. C'est ce que l'on appelle l'étape d'*ingestion* (cf 5.2).

4.5 Module de recherche dans ELAN

Afin de rendre un corpus exploitable, il est indispensable de doter celui-ci d'un outil de recherche à la fois simple et puissant c'est-à-dire capable de formuler des requêtes précises et obtenir des réponses dans un temps raisonnable.

Cet outil de recherche peut se baser soit sur un langage de requête préexistant (XQuery ou SQL) soit sur un langage dérivé et spécifique au format d'annotation du logiciel dont il dépend (ex : NXT Search ou LQL).

Le logiciel ELAN contient nativement un module de recherche. Ce module permet de faire différents types de recherches dans un ensemble de fichiers défini par l'utilisateur mais qui doivent être accessibles localement sur son ordinateur.

Le formulaire de recherche permet d'exprimer des contraintes aussi bien séquentielles que hiérarchiques (que l'on appelle horizontales et verticales) sur des couches spécifiques (contraintes de couches).

Suivant la finesse de l'annotation, le chercheur peut s'offrir de nouvelles perspectives de recherche. De plus toute consultation conservant le lien avec les ressources, il est donc toujours possible d'écouter le segment correspondant à une annotation.

4.6 Conclusion

Nous venons, dans ces 4 premières parties, de faire un bilan du travail qui avait été fait en amont de notre contribution qui, elle, va essentiellement adresser la question de la mutualisation des ressources et de leur analyse d'un point de vue plus quantitatif.

Comme exposé dans la section 1.2, ce corpus a pour vocation de mettre à disposition des linguistes de terrain et des typologues des ressources sonores transcrites et annotées de langues afroasiatiques peu décrites pour lesquelles il est difficile d'obtenir des données audio. Il faut donc leur permettre un accès aux métadonnées (cf 2.2), aux fichiers d'annotations mais aussi aux enregistrements et doter ce corpus d'un outil d'interrogation qui permette son exploration.

Il a également l'ambition de fournir des esquisses grammaticales enrichies par des références relevées directement dans le corpus ainsi que des études de phénomènes à travers plusieurs langues.

Il faudrait donc pouvoir proposer à l'utilisateur de ce corpus une interface lui permettant d'avoir une vue d'ensemble du corpus puis d'accéder à toutes les informations citées ci-dessus.

C'est ce que nous allons aborder dans les parties suivantes.

Chapitre 5

Outil de recherche web

5.1 Présentation de l'interface

Afin de permettre aux chercheurs d'accéder à distance à leurs données ainsi qu'à celles d'autres langues et aux linguistes et typologues d'accéder à l'ensemble du corpus, nous avons développé une interface web simple d'utilisation, offrant plusieurs fonctionnalités de recherche, consultation, concordance et ne nécessitant pas l'installation de logiciel mais simplement l'utilisation d'un navigateur.

5.1.1 Page d'accueil et gestion des métadonnées

La page d'accueil du site présente le corpus sous forme d'arborescence dont la figure 5.1 est un échantillon.

FIGURE 5.1 – Echantillon de l'arborescence de la liste des fichiers ELAN du corpus CorpAfroAs avant connexion

Chacune des langues constitue une branche elle-même découpée en un nombre

de sous branches variant de 1 à 4 : « Conversation », « Narration », « Samples » et « PDF ».

Une fois connecté (cf figure 5.2) et aux conditions de restriction d'accès près (si certains documents ne peuvent être diffusés librement), l'utilisateur a le droit d'accéder et télécharger les fichiers d'annotation (icone ELAN) et les fichiers sons (icone « wav ») ainsi que de lancer une recherche sur ceux-ci (cf 5.1.2).

La sous-branche « Samples » contient des échantillons de fichiers accessibles à tout utilisateur non enregistré.

La sous-branche « PDF » contient les éventuels sketchs grammaticaux ou articles concernant la langue en question.

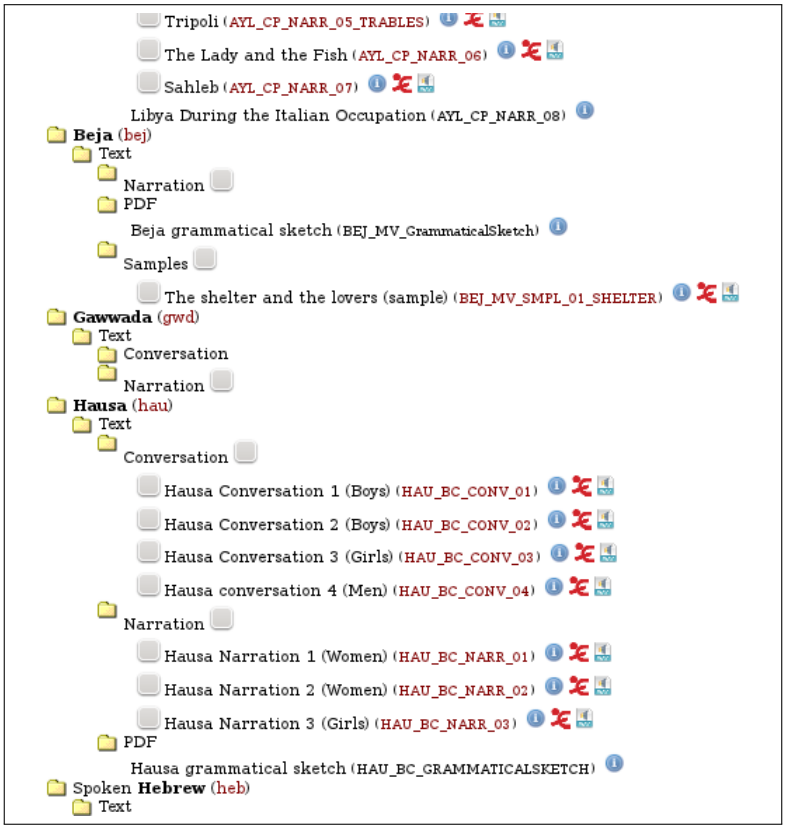


FIGURE 5.2 – Echantillon de l'arborescence de la liste des fichiers ELAN du corpus CorpAfroAs après connexion

Cette première page du site répond donc au besoin de consultation des données brutes requis par le projet mais aussi à celui d'accès aux métadonnées.

Comme nous l'avons vu dans la section 2.2, les métadonnées sont des données descriptives concernant la source, la nature et le contenu des fichiers d'annotations (dans notre cas des fichiers ELAN, .eaf), et des fichiers sons, .wav, associés. Ces informations sont très importantes pour tout projet ayant pour vocation la gestion de ressources documentaires. Il était donc nécessaire d'en concevoir un stockage et un accès simples et pérennes.

La consultation des métadonnées OLAC de tous les fichiers du corpus est libre depuis cette page d'accueil [1]. Cette page a été écrite en php et les métadonnées sont visualisables en cliquant sur l'icône bleue figurant un « i ».

1. Il est à noter que sur les 197 fichiers dont les métadonnées sont consultables, seulement 98 sont à ce jour (janvier 2013) interrogeables par l'interface de recherche car tous les fichiers n'ont pas encore fini d'être annotés.

De plus, il existe une table *olac* de la base de données *Corpafas* (dont nous avons parlé dans la section 4.4) dans laquelle une partie de ces métadonnées est enregistrée (par exemple les champs `creator`, `title`, `contributor`, `date_created`...) et dont la clé primaire est un identifiant numérique permettant de désigner sans ambiguïté un fichier ELAN ou .wav. Cet identifiant est utilisé dans les tables de stockage des annotations (cf 5.2) ainsi que dans la table `droits` qui gère les droits des différents utilisateurs quant à l'accès aux différents fichiers. Ces droits sont accordés à l'utilisateur après une demande faite auprès du propriétaire concerné.

5.1.2 Page de recherche

Nous nous sommes inspirés de l'outil de recherche existant dans le logiciel ELAN en essayant de conserver le maximum de fonctionnalités de celui-ci (expressions des contraintes horizontales, verticales et de couches, lien au son).

Nous avons repris la même organisation des données et l'avons légèrement modifiée pour l'outil de recherche en ligne. En particulier nous l'avons adapté au fait que les données (fichiers d'annotations, fichiers son, métadonnées) sont stockées en dur et de façon pérenne au sein du TGE Adonis (très grande infrastructure de recherche pilotée par l'*Institut des sciences humaines et sociales*) dont une des missions de service est de mettre à disposition des laboratoires de recherche des infrastructures et des systèmes informatiques « pour mutualiser, diffuser et stabiliser dans le temps les données et documents issus de la recherche ainsi que les méthodes de traitement afférentes » [2].

Contrairement à l'outil de recherche local qui s'appuie sur HSQLDB, notre outil web, basé sur la paquetage *mfSearch* du MPI [3], utilise PostgreSQL qui est un moteur de bases de données relationnelle et objet, open source, respectant les normes SQL, robuste et supportant des volumes importants de données. Il n'est pas limité au niveau de la taille de ses tables alors que HSQLDB se limite à des tables de 8 GB.

Une capture d'écran de l'interface de recherche est présentée figure 5.3.

Elle se compose de 3 parties : le domaine de recherche qui permet de voir le nombre de fichiers sur lequel vont se faire les recherches (le passage de la souris sur le mot « file » affiche la liste des fichiers sélectionnés dans la page d'accueil du site), la partie « Listes et concordances » et la partie formulaire de recherche. Ces diverses fonctionnalités sont décrites ci-dessous.

Les fonctionnalités de l'outil en ligne permettent de :

- lister les fréquences d'apparition des mots, morphèmes et étiquettes d'une sélection de fichiers (pour le repérage des formes rares, déviantes, voire même les erreurs de transcription).
  La figure 5.4 fournit un exemple de liste de `rx` triés par nombre d'occurrences décroissant sur 4 fichiers dont la liste est obtenue par le passage de la souris sur le « search domain ».
  Le tableau de droite donne la liste des étiquettes simples par oppostion aux annotations dans `rx` qui peuvent être la concaténation de plusieurs étiquettes ou gloses séparées par un point.
  Les annotations et étiquettes sont cliquables et redirigent vers la liste des refs qui les contiennent. Les étiquettes qui apparaissent en violet sont celles pour lesquelles la liste a été tronquée car trop longue.
- faire une concordance pour une approche qualitative d'occurrences en contexte.
  Les concordances permettent de se faire une meilleure idée des conditionne-

2. http://wwww.tge-adonis.fr:service/grille-adonis consultée le 8 janvier 2013
3. Site du Max Planck Institute : http://www.eva.mpg.de/lingua/

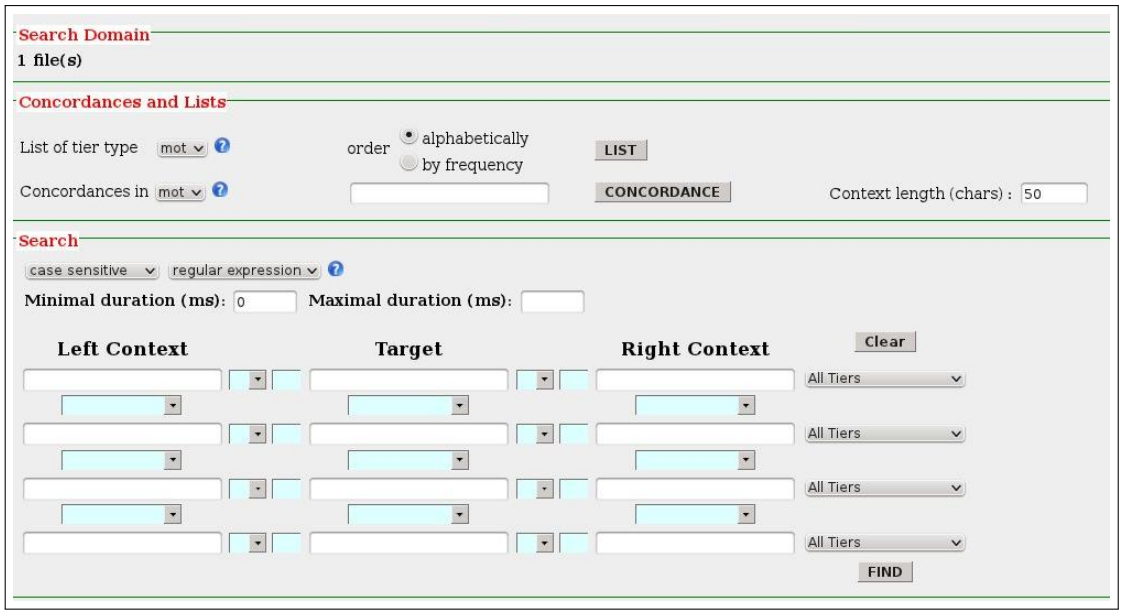
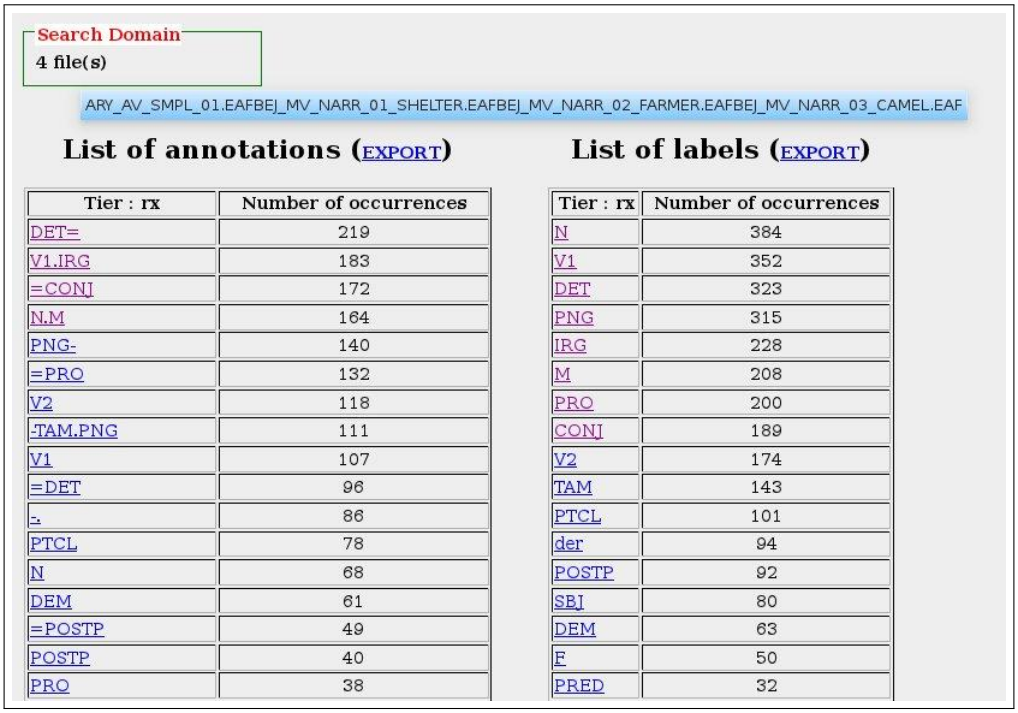


FIGURE 5.3 – Interface de recherche de l'outil web



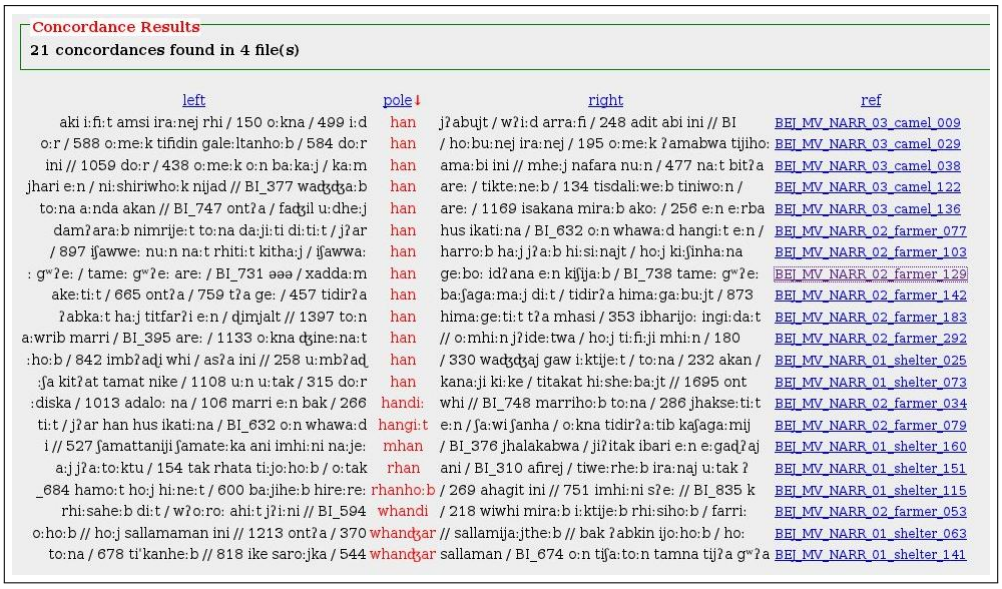
Tier : rx	Number of occurrences
DET=	219
V1.IRG	183
=CONJ	172
N.M	164
PNG-	140
=PRO	132
V2	118
-TAM.PNG	111
V1	107
=DET	96
-.	86
PTCL	78
N	68
DEM	61
=POSTP	49
POSTP	40
PRO	38

Tier : rx	Number of occurrences
N	384
V1	352
DET	323
PNG	315
IRG	228
M	208
PRO	200
CONJ	189
V2	174
TAM	143
PTCL	101
der	94
POSTP	92
SBJ	80
DEM	63
F	50
PRED	32

FIGURE 5.4 – Exemple de listes de fréquences des annotations `rx` sur 4 fichiers du Beja

ments qui peuvent exister entre les segments au niveau phonologique, distinguer les différents usages d'une forme lexicale afin d'en extraire les différents sens par exemple pour la constitution d'un dictionnaire.
Dans l'exemple de la figure 5.5, on recherche les concordances pour le motif « han » dans les annotations de la couche `mot`.
Des clics sur les en-têtes de colonnes (« left », « pole » et « right ») permettent de changer l'ordre de tri.

– localiser un motif, une forme particulière, pouvant être spécifiée par une expression régulière, sur une couche particulière (c'est à-dire non seulement



Concordance Results

21 concordances found in 4 file(s)

left	pole	right	ref
aki i:fi:t amsi ira:nej rhi / 150 o:kna / 499 i:d	han	jʔabujt / wʔi:d arra:fi / 248 adit abi ini // BI	BEJ_MV_NARR_03_camel_009
o:r / 588 o:me:k tifidin gale:ltanho:b / 584 do:r	han	/ ho:bu:nej ira:nej / 195 o:me:k ʔamabwa tijiho:	BEJ_MV_NARR_03_camel_029
ini // 1059 do:r / 438 o:me:k o:n ba:ka:j / ka:m	han	ama:bi ini // mhe:j nafara nu:n / 477 na:t bitʔa	BEJ_MV_NARR_03_camel_038
jhari e:n / ni:shiriwho:k nijad // BI_377 wadʒdʒa:b	han	are: / tikte:ne:b / 134 tisdali:we:b tiniwo:n /	BEJ_MV_NARR_03_camel_122
to:na a:nda akan // BI_747 ontʔa / fadʒil u:dhe:j	han	are: / 1169 isakana mira:b ako: / 256 e:n e:rba	BEJ_MV_NARR_03_camel_136
damʔara:b nimrije:t to:na da:ji:ti di:ti:t / jʔar	han	hus ikati:na / BI_632 o:n whawa:d hangi:t e:n /	BEJ_MV_NARR_02_farmer_077
/ 897 iʃawwe: nu:n na:t rhiti:t kitha:j / iʃawwa:	han	harro:b ha:j jʔa:b hi:si:najt / ho:j ki:ʃinha:na	BEJ_MV_NARR_02_farmer_103
: gʷʔe: / tame: gʷʔe: are: / BI_731 əəə / xadda:m	han	ge:bo: idʔana e:n kiʃija:b / BI_738 tame: gʷʔe:	BEJ_MV_NARR_02_farmer_129
ake:ti:t / 665 ontʔa / 759 tʔa ge: / 457 tidirʔa	han	ba:ʃaga:ma:j di:t / tidirʔa hima:ga:bu:jt / 873	BEJ_MV_NARR_02_farmer_142
ʔabka:t ha:j titfarʔi e:n / djmjalt // 1397 to:n	han	hima:ge:ti:t tʔa mhasi / 353 ibharijo: ingi:da:t	BEJ_MV_NARR_02_farmer_183
a:wrib marri / BI_395 are: / 1133 o:kna dʒine:na:t	han	// o:mhi:n jʔide:twa / ho:j ti:fi:ji mhi:n / 180	BEJ_MV_NARR_02_farmer_292
:ho:b / 842 imbʔadi whi / asʔa ini // 258 u:mbʔad	han	/ 330 wadʒdʒaj gaw i:ktije:t / to:na / 232 akan /	BEJ_MV_NARR_01_shelter_025
:ʃa kitʔat tamat nike / 1108 u:n u:tak / 315 do:r	han	kana:ji ki:ke / titakat hi:she:ba:jt // 1695 ont	BEJ_MV_NARR_01_shelter_073
:diska / 1013 adalo: na / 106 marri e:n bak / 266	handi:	whi // BI_748 marriho:b to:na / 286 jhakse:ti:t	BEJ_MV_NARR_02_farmer_034
ti:t / jʔar han hus ikati:na / BI_632 o:n whawa:d	hangi:t	e:n / ʃa:wi ʃanha / o:kna tidirʔa:tib kaʃaga:mij	BEJ_MV_NARR_02_farmer_079
i // 527 ʃamattaniji ʃamate:ka ani imhi:ni na:je:	mhan	/ BI_376 jhalakabwa / jiʔitak ibari e:n e:gadʔaj	BEJ_MV_NARR_01_shelter_160
a:j jʔa:to:ktu / 154 tak rhata ti:jo:ho:b / o:tak	rhan	ani / BI_310 afirej / tiwe:rhe:b ira:naj u:tak ʔ	BEJ_MV_NARR_01_shelter_151
_684 hamo:t ho:j hi:ne:t / 600 ba:jihe:b hire:re:	rhanho:b	/ 269 ahagit ini // 751 imhi:ni sʔe: // BI_835 k	BEJ_MV_NARR_01_shelter_115
rhi:sahe:b di:t / wʔo:ro: ahi:t jʔi:ni // BI_594	whandi	/ 218 wiwhi mira:b i:ktije:b rhi:siho:b / farri:	BEJ_MV_NARR_02_farmer_053
o:ho:b // ho:j sallamaman ini // 1213 ontʔa / 370	whandʒar	// sallamija:jthe:b // bak ʔabkin ijo:ho:b / ho:	BEJ_MV_NARR_01_shelter_063
to:na / 678 ti'kanhe:b // 818 ike saro:jka / 544	whandʒar	sallaman / BI_674 o:n tiʃa:to:n tamna tijʔa gʷʔa	BEJ_MV_NARR_01_shelter_141

FIGURE 5.5 – Exemple de concordances sur le motif « han » dans 4 fichiers du Beja

dans le texte mais aussi dans des lemmes ou des catégories grammaticales) avec la possibilité d'indiquer des contraintes séquentielles (l'axe syntaxique ou contexte) et/ou hiérarchiques (l'axe paradigmatique). On les appelle également contraintes horizontales et verticales.

Dans l'exemple de la figure 5.6, on recherche les suites de mots (« . » dans `mot`) constitués d'un adjectif (« ADJ » dans `rx`) suivi d'un nom (« N » dans `rx`) séparé d'un verbe (« V » dans `rx`) par au plus 3 annotations.

Il est à noter que dans la cellule « Right context », il faudrait en fait entourer le « V » de « \b » (frontières de mot) pour ne pas récupérer d'éventuels adverbes, glosés « ADV ».

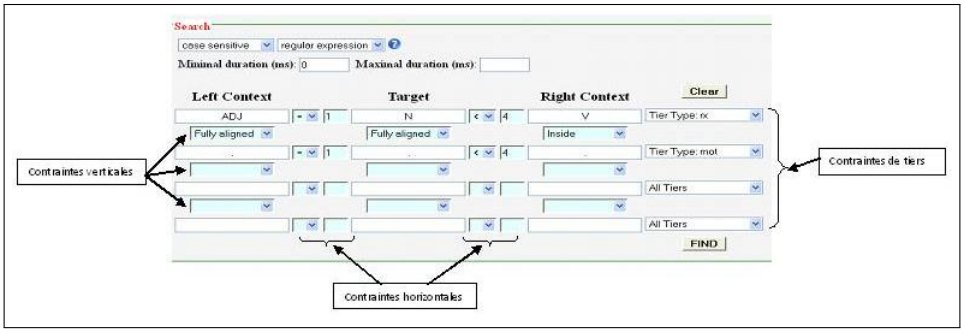


FIGURE 5.6 – Exemple de recherche sur les fichiers BEJ_MV_NARR_01_Shelter.eaf et BEJ_MV_NARR_02_Farmer.eaf

Les résultats obtenus pour cette recherche sont présentés sur la figure 5.7.

Les refs contenant le motif de recherche (correspondant au champ du milieu dans le formulaire de recherche) sont indiquées. L'utilisateur peut soit cliquer sur une ref individuelle soit en sélectionner plusieurs et cliquer sur le bouton « Show selected items ».

En cliquant sur la deuxième ref de la figure 5.7 et en demandant l'affichage de 2 refs supplémentaires à droite de celle-ci à l'aide du bouton « Extend Display », on obtient l'aperçu de la figure 5.8 :

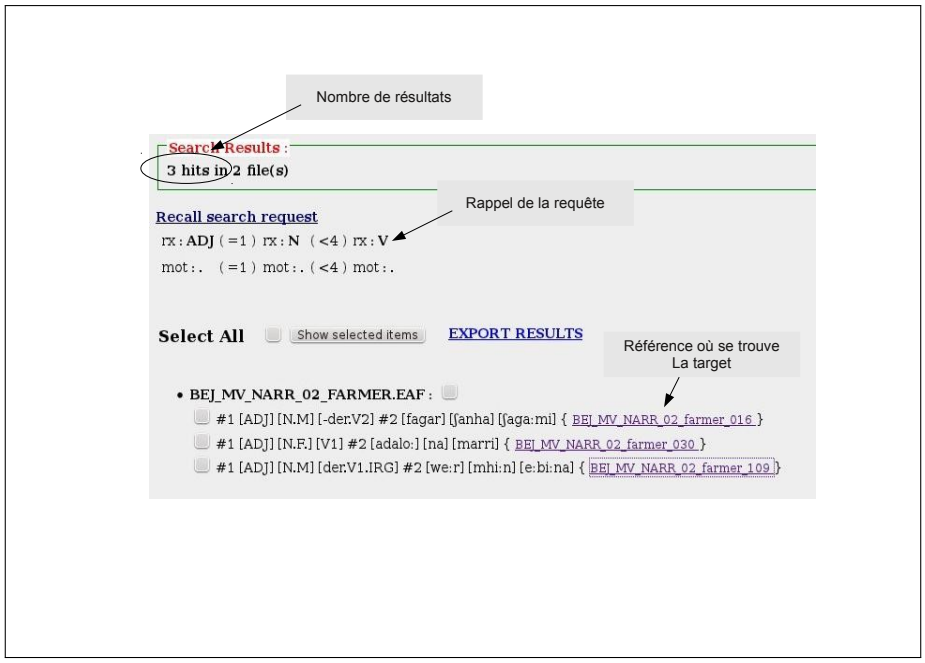


FIGURE 5.7 – Résultats de la recherche formulée dans la figure 5.6

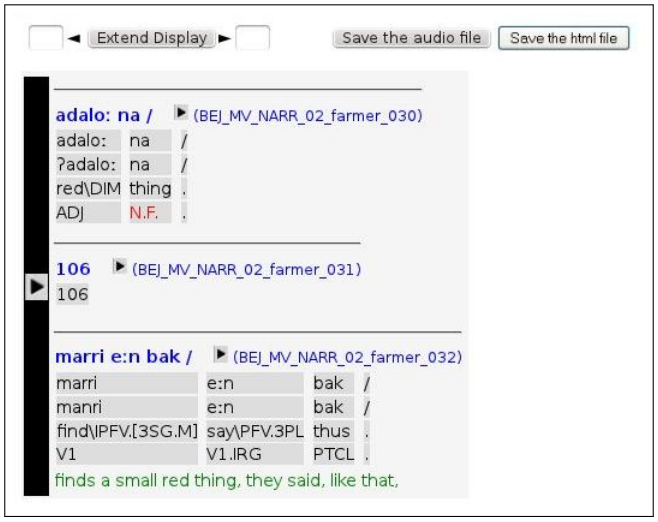


FIGURE 5.8 – Affichage des refs BEJ_MV_NARR_02_farmer_030 à 032

5.2 Ingestion des fichiers

Comme nous l'avons vu dans la partie 4.4, les données dans lesquelles le chercheur désire faire des recherches sont stockées dans deux tables principales : `annotations` et `tiers`, contenant chacune 9 champs. C'est cette étape de récupération des données dans les fichiers XML pour peupler une base de données relationnnelle qui est appelée *ingestion*.

Lors de cette étape d'ingestion, chaque fichier XML est parcouru par un parseur SAX (*Simple API XML*) qui récupère toutes les informations nécessaires au remplissage des tables `annotations` et `tiers`. Une fois les tables remplies, des index sur ces tables sont créés afin d'optimiser la vitesse des requêtes.

Les classes en jeu lors de cette étape de lecture des flux XML sont :

– `AnnexAnnotation`. C'est un objet sérialisé contenant toutes les informations relatives à une annotation d'un fichier EAF donné, c'est-à-dire les informations

que l'on trouve à l'intérieur des balises ANNOTATION (sa valeur, ses temps de début et de fin, son identifiant...).
Il est simplement défini par des attributs de classe et un constructeur prenant en arguments les paramètres sus-cités.

- AnnexTier. C'est un objet sérialisé contenant toutes les informations relatives à une couche d'un fichier EAF donné, c'est-à-dire les informations que l'on trouve à l'intérieur des balises TIER ainsi que la liste des annotations qu'elle contient (soit une ArrayList d'objets de type AnnexAnnotation).
  Il est défini par des attributs de classe correspondant aux attributs de la balise TIER et par un constructeur prenant en arguments le nom et le type de la couche (soit les attributs TIER_ID et LINGUISTIC_TYPE_REF). Il possède également une méthode permettant d'ajouter une annotation à la liste des annotations, qui n'est pas utilisée dans le cas de fichiers EAF mais dans celui des fichiers Toolbox.
- AnnexTranscription. C'est un objet sérialisé contenant les informations relatives à un fichier de transcription telles que le type du fichier (EAF, Toolbox...), l'id du fichier correspondant au node_id de la table olac, la liste des couches qu'il contient (soit une ArrayList d'objets de type AnnexTier) et des informations sur la validité (au sens XML) du document ainsi que sur les éventuelles erreurs rencontrées lors du parsing du fichier.
  Son constructeur prend en argument le node_id du fichier, l'identifiant de son type (ex : « 0 » pour les fichiers EAF) et le nom ou l'url du fichier.
- AnnexEAFHandler. C'est une classe qui hérite de la classe DefaultHandler et qui définit les méthodes permettant de lire le fichier XML et récupérer les informations utiles au traitement que l'on cherche à faire, dans notre cas peupler les tables de stockage.
  Son constructeur prend en argument un objet AnnexTranscription.
- AnnexParser. Cette classe contient seulement les différentes méthodes de parsing suivant les types de fichier définies dans les différents handlers. Dans notre cas, nous ne nous sommes intéressés qu'au handler des fichiers EAF.

Aux champs existants dans la table annotations, nous avons ajouté le champ ref_ref pour accéder plus facilement à l'unité prosodique contenant l'information recherchée.

Après avoir essayé d'accéder à cette information par des requêtes SQL requérant une jointure entre les deux tables et la création d'une table intermédiaire, ce qui est apparu comme une méthode trop lourde et peu efficace, nous avons pris le parti de la récupérer directement au moment de la lecture des fichiers XML par l'API SAX.

Pour cela, nous avons procédé à deux modifications :

- ajout d'un attribut refName dans l'objet AnnexAnnotation,
- ajout de la méthode getRefName dans la classe AnnexEAFHandler qui prend en paramètre l'id de l'annotation de référence (ex : ann125) relative à l'annotation en train d'être parsée et obtenue par la méthode getAlignedAnnotation. Cette méthode permet donc de récupérer la valeur de cette référence (ex : BEJ_MV_NARR_05_shelter_210). Celle-ci est ensuite mémorisée comme champ refName de l'objet AnnexAnnotation.

Aux champs existants dans la table tiers, nous avons ajouté le champ node_id qui permet de faire directement le lien avec la table olac et donc avec les informations du fichier contenant la couche en question. Ce champ est récupéré au moment de l'ingestion car l'on fournit à la méthode myFillDB une liste de string de la forme *node_id%0%url_du_fichier* avec « 0 » correspondant au code des fichiers EAF.

Cette liste est générée par la méthode createDescTab de la classe OlacIngester.

Voici un schéma récapitulatif du processus d'ingestion :

- **mscdb** est une instance de la classe `MySearchCorpusDB` qui possède la méthode `myFillDB` déclenchant tout le processus d'ingestion
- **oi** est une instance de la classe `OlacIngester`
- **desc** est l'ArrayList de string de la forme *node_id%0%url_du_fichier* suscitée
- **olac** est le nom de la table contenant les informations sur les fichiers nécessaires au remplissage de **desc**

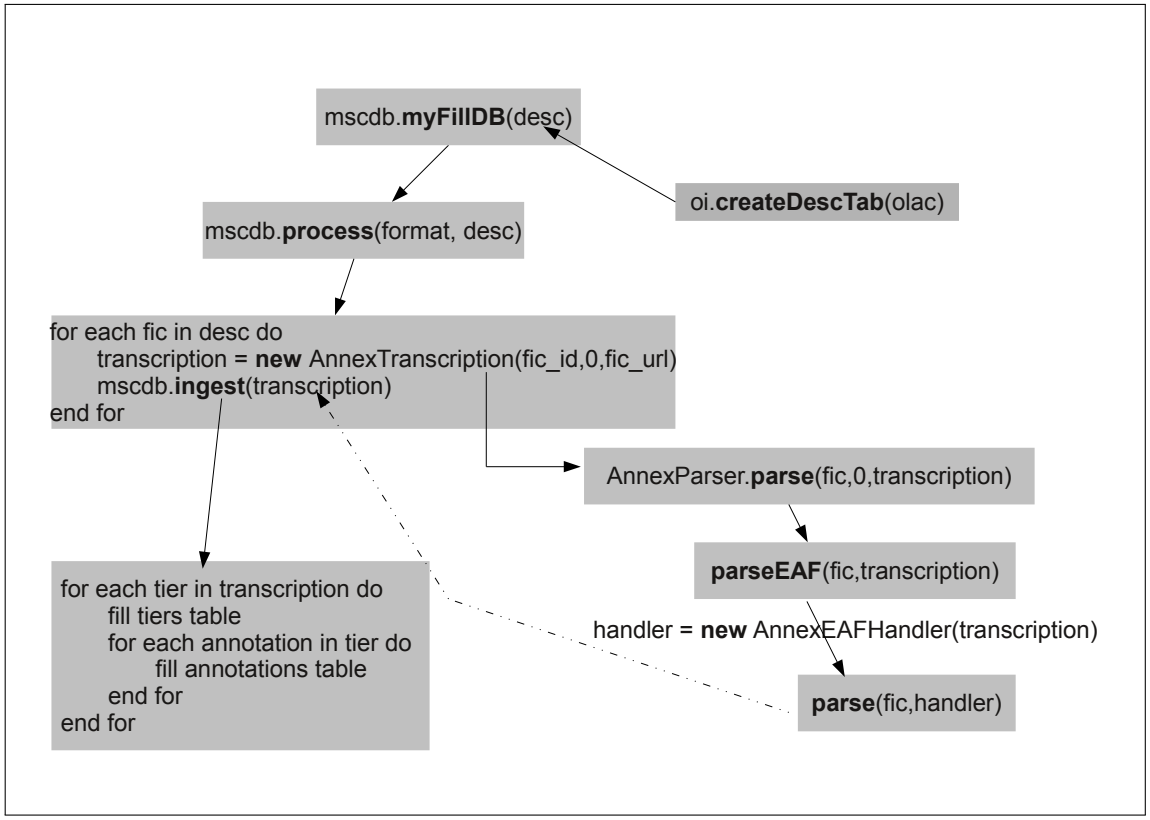


FIGURE 5.9 – Schéma récapitulatif de l'ingestion

Il est aussi à noter que nous avons développé deux modes d'ingestion :

- ingestion totale : si des tables existent déjà, elles sont sauvegardées à l'aide d'une commande système `dump` dans un dossier d'archive sur le serveur avant d'être supprimées et remplacées par de nouvelles tables construites à partir des fichiers à ingérer,
- ingestion partielle : les données des nouveaux fichiers sont ajoutées à la suite de celles qui existent déjà.

5.3 Génération de la requête SQL

Nous allons voir dans cette partie comment les paramètres de la requête formulée par l'utilisateur au travers du formulaire de recherche de l'interface web (cf figure 5.6) sont transmis au programme java et comment les résultats sont générés et retournés à l'utilisateur.

Comme vu précédemment dans la partie 4.5, tout le mécanisme de traitement des requêtes existe déjà dans le logiciel ELAN à travers son module de recherche. Le travail que nous avions donc à faire était tout d'abord d'isoler les classes chargées de

ce traitement et de fournir au système les paramètres dans un format interprétable par celui-ci.

Le formulaire de recherche est écrit en JSP, *Java Server Pages* (`mfsearch.jsp`) et transmet ses paramètres à une servlet (`FormulaireAction`), qui est une classe java permettant de gérer des requêtes HTTP.

La transmission de la JSP à la servlet se fait par l'intermédiaire d'une chaîne de caractères encodée de la manière suivante :

- les paramètres de la requête sont transmis sous la forme « nom_du_paramètre=valeur »,
- le domaine de recherche est spécifié par 2 paramètres : le *domainSize* qui donne le nombre Nf de fichiers sélectionnés en amont et les *indexSelk*, avec k compris entre 0 et Nf-1 et dont les valeurs sont de la forme « node_id%nom_du_fichier » (ex : 61%BEJ_MV_NARR_01_SHELTER),
- les modes de recherche sont transmis par les paramètres *mode1* (pour la casse) et *mode2* (pour la recherche avec ou sans prise en compte des expressions régulières). Il est à noter qu'il existe un paramètre *mode0* dont la valeur est « Annotation » mais qui est transmis en champ masqué et qui ne sert que pour le décodage de la requête assuré par une méthode préexistante dans ELAN et que nous avons réutilisée,
- les N champs à remplir sont appelés *motifk* avec k un entier compris 0 et N-1 (dans l'exemple 5.6, N=12, 3 motifs par ligne sur 4 lignes),
- les contraintes horizontales sont composées de deux champs : un champ « signe » à sélectionner dans une liste déroulante et un champ « valeur » renseignée par l'utilisateur. Le nom de ces paramètres sont respectivement *hConsSi_j* et *hConsNi_j* avec i l'indice de la ligne et j l'indice de la colonne,
- les contraintes verticales ne sont composées que d'un champ à sélectionner dans une liste déroulante. Le nom de ces paramètres sont *vConsi_j*, i et j étant les indices de ligne et colonne,
- les spécifications sur les couches qui sont aussi sélectionnables dans une liste déroulante et sont trasmises avec le paramètre *tierk* avec k compris entre 1 et le nombre de lignes du formulaire (soit 4 dans l'exemple 5.6).

Dans l'exemple de la figure 5.6, on aurait les paramètres suivants :

- domainSize = « 2 »,
- indexSel0 = « 61%BEJ_MV_NARR_01_Shelter.eaf », indexSel1 = « 59%BEJ_MV_NARR_02_Farmer.eaf »,
- motif0 = « ADJ », motif1 = « N », motif2 = « V », motif3 = « . », motif4 = « . », motif5 = « . »,
- hConsS0_0 = « = », hConsS0_1 = « < », hConsS1_0 = « = », hConsS1_1 = « < »,
- hConsN0_0 = « 1 », hConsN0_1 = « 4 », hConsN1_0 = « 1 », hConsN1_1 = « 4 »,
- vCons0_0 = « Fully aligned », vCons0_1 = « Fully aligned », vCons0_2 = « Inside »,
- tier0 = « Tier Type : rx », tier1 = « Tier Type : mot ».

Une fois ces paramètres récupérés par la servlet, ceux-ci sont encodés à l'aide de la méthode `encodeMultipleLayerQuery` de la classe `ClassTest` afin d'être transmis à la méthode `myDoComplexQuery` de la classe `PostgresQuery` qui assure le traitement de la requête.

Celui-ci est basé sur une méthode récursive (`investigateComplexPattern` de la classe `SearchQuery`) qui envoie des requêtes SQL successives à la base de données de stockage afin d'obtenir la liste des ids des annotations satisfaisant toutes les

contraintes de la recherche.

La matrice du formulaire de recherche est parcouru de gauche à droite et de haut en bas à partir du premier champ motif renseigné.

5.4 Autres fonctionnalités de l'outil

5.4.1 Listes de fréquences

La classe `AnnotationQuery` permet de générer les listes de fréquences d'annotations de type `mot`, `mb`, `ge` ou `rx`, ordonnées par ordre alphabétique des annotations ou bien par ordre décroissant des fréquences.

Pour la couche `mot`, on obtient également la liste des pauses avec leur nombre d'occurrences.

Pour les couches `ge` et `rx`, on obtient parallèlement les listes d'étiquettes et leurs fréquences. On entend par « étiquettes » :

- pour `ge`, la partie correspondant aux gloses simples (ex : dans « go_to_the_well \IPFV.[3SG.M] », on veut récupérer `IPFV`, `3SG` et `M`).
  L'expression régulière utilisée est : `[0-9]?n?[A-Z]+[0-9]\*?`
- pour `rx`, il s'agit aussi des gloses mais sur cette couche, on n'a pas de traduction du morphème et théoriquement il n'y a pas de minuscule à part pour la glose « der » correspondant à « derivation » (ex : dans « der.V1.IRG », on veut récupérer `der`, `V1` et `IRG`).
  L'expression régulière utilisée est : `[0-9]?[A-Za-z]+[0-9]\*?`

Pour chacune des fréquences associées, on peut en cliquant dessus avoir la liste des refs contenant l'annotation ou l'étiquette en question.

Pour le tri par fréquences décroissantes, il a fallu définir une relation d'ordre permettant de comparer les nombres d'occurrences puis les annotations dans le cas de nombres d'occurrences égaux. Cette tâche est accomplie à travers la méthode `sortByValue(Map<String, Integer> map)` qui renvoie une `Map<String, Integer>`, c'est à dire une table de hash dont les clés sont des chaînes de caractères (les étiquettes) et les valeurs des entiers (les nombres d'occurrences). Cette méthode fait appel à une méthode statique `ValueThenKeyComparator` qui hérite de l'interface `Comparable`.

```
/**
 * Methode pour le tri par frequence decroissante
 */
public Map<String, Integer> sortByValue(Map<String, Integer> map) {
  List<Map.Entry<String, Integer>> list = new ArrayList<Map.Entry<String,
      Integer>>(map.entrySet());
  Collections.sort(list, new ValueThenKeyComparator<String, Integer>());

  Map<String, Integer> result = new LinkedHashMap<String, Integer>();
  for (Iterator it = list.iterator(); it.hasNext();) {
      Map.Entry entry = (Map.Entry)it.next();
      result.put((String) entry.getKey(), (Integer) entry.getValue());
  }
  return result;
}
/**
 * Definition de la relation d'ordre pour le tri
 */
public static class ValueThenKeyComparator<K extends Comparable<? super K>,V
    extends Comparable<? super V>>
implements Comparator<Map.Entry<K, V>> {

    public int compare(Map.Entry<K, V> a, Map.Entry<K, V> b) {
      // on commence par comparer les frequences
      int cmp1 = - a.getValue().compareTo(b.getValue());
      if (cmp1 != 0) {
```

```
            return cmp1;
        } else {
        // dans le cas de frequences egales, on compare alphabetiquement les
            etiquettes
            return a.getKey().compareTo(b.getKey());
        }
    }
}
```

5.4.2 Concordances

C'est l'objet `Concordancier` héritant de la classe abstraite `SearchQuery` et muni de la méthode `grepRegexp` qui permet de lister les contextes d'occurrence pour un terme de requête dans un domaine (ensemble de fichiers) défini par l'utilisateur.

Il est à noter que la requête est sensible à la casse et qu'elle peut se formuler à l'aide des expressions régulières.

La méthode `grepRegexp` prend en argument quatre paramètres :

- la liste des identifiants des fichiers formant le domaine de recherche (de type ArrayList de String),
- le terme de requête (de type String),
- la couche de requête (de type String) à choisir parmi les couches `mot`, `mb`, `ge` et `rx`,
- la connexion à la base de données (de type Connection) qui se fait par l'intermédiaire d'un pilote JDBC et qui permet au programme java d'interroger le système de gestion de bases de données.

Cette méthode renvoie une ArrayList de String contenant toutes les occurrences en contexte (dont la longueur peut-être spécifiée en nombre de caractères au moment de la requête) du terme recherché.

A nouveau, chaque résultat de cette liste est associée à la ref contenant l'occurrence et permet donc la consultation de l'unité avec ses annotations et l'accès à l'enregistrement du segment.

```
public ArrayList<String> grepRegexp (ArrayList<String> nodeIds, String motif,
    String selectedTier, Connection con) {
    ArrayList<String> listConcordances = new ArrayList<String>();
    ResultSet rsAnnot = null;
    ResultSet rsContextG = null;
    ResultSet rsContextD = null;

    PreparedStatement stmtAnnot = null;
    PreparedStatement stmtContext = null;
    String sep = "\u2022";

    String findAnnotation = "SELECT annotation, ann_id, ann_tier_id, ref_ref
        FROM search.annotations WHERE " + "annotation ~ '" + escapeRegExp(
        motif) + "'" +
            " AND ann_tier_id IN " +
            "(SELECT tier_id FROM search.tiers WHERE node_id = ? AND tier_type
                ='"+selectedTier+"' )";
    String findContext = "SELECT annotation, ann_id FROM search.annotations
        WHERE ann_id < ? AND ann_id > ? AND ann_tier_id=? ORDER BY ann_id";

    // On pourrait aussi recuperer les queryModes depuis le formulaire... ici
        ils sont imposes
    String[] queryModes=new String[3];
    queryModes[0]=Constants.ANNOTATIONS;
    queryModes[1]=Constants.CASE_SENSITIVE;
    queryModes[2]=Constants.REGEXP_MATCH;
    // on prepare le pattern qui va servir a verifier que le hit obtenu par la
         requete matche bien la regexp
    Pattern regExpPattern = Pattern.compile(motif);
    try {
        // on convertit la requete avant de l'executer
```

```
25          stmtAnnot = con.prepareStatement(convertRegexpPatternsForPostgres(
                findAnnotation),
                    ResultSet.TYPE_SCROLL_INSENSITIVE,
27                  ResultSet.CONCUR_READ_ONLY);
            stmtContext = con.prepareStatement(findContext,
29                  ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
31
            for (int i=0; i<nodeIds.size(); i++) {
33              stmtAnnot.setString(1, nodeIds.get(i));
                rsAnnot = stmtAnnot.executeQuery();
35              while (rsAnnot.next()) {
                    String s = rsAnnot.getString(1); // s = annotation
37                  // le dernier parametre (boolean false) indique que le pattern
                         n'est pas un NOT pattern
                    int realHit = countHitsInAnnotation(s, queryModes, motif,
                        regExpPattern, false);
39                  if (realHit>=1){ // pour verifier qu'on matche bien la regexp
                        au moins une fois
                        int annId = rsAnnot.getInt(2);
41                      int annTierId = rsAnnot.getInt(3); // necessaire pour la
                            seconde requete
                        String ref = rsAnnot.getString(4); // on recupere la ref
                            contenant le motif
43                      // on lance la requete pour trouver le contexte gauche (20
                             annotations pour le moment)
                        stmtContext.setInt(1, annId);
45                      stmtContext.setInt(2, annId-nbAnnot);
                        stmtContext.setInt(3, annTierId);
47                      rsContextG = stmtContext.executeQuery();
                        // on prepare un StringBuffer pour concatener les
                            contextes au motif
49                      StringBuffer concordance = new StringBuffer();
                        // on recupere les mots du contexte gauche
51                      while (rsContextG.next()) {
                            String annotG = rsContextG.getString(1);
53                          int annIdG = rsContextG.getInt(2);
                            concordance.append(annotG + " ");
55                      }
                        int contextGLength = concordance.length();
57                      String splitContextG = "";
                        if (contextGLength > lg) {
59                          splitContextG = concordance.substring(contextGLength-
                                lg, contextGLength);
                        } else {
61                          splitContextG = concordance.toString();
                        }
63
                        // on lance la requete pour trouver le contexte droit
65                      stmtContext.setInt(1, annId+nbAnnot);
                        stmtContext.setInt(2, annId);
67                      stmtContext.setInt(3, annTierId);
                        rsContextD = stmtContext.executeQuery();
69                      if (concordance.length()>0)
                            concordance.delete(0,contextGLength-1); // on efface
                                ce qu'il y a dans concordance avant d'y mettre le
                                contexte droit
71                      while (rsContextD.next()) {
                            String annotD = rsContextD.getString(1);
73                          concordance.append(annotD + " ");
                        }
75                      int contextDLength = concordance.length();
                        String splitContextD = "";
77                      if (contextDLength > lg) {
                            splitContextD = concordance.substring(0, lg);
79                      } else {
                            splitContextD = concordance.toString();
81                      }
                        // on concatene le motif (entoure du separateur pour
                            preparer le futur split) au contexte gauche tronque a
                            "lg" caracteres et au contexte droit
83                      // et on ajoute le resultat a la liste des concordances
                        listConcordances.add(splitContextG.concat(sep+s+sep + " "
```

```
                        + splitContextD + sep+ ref));
85                  } // if
                } // while
87          } // for

89      }catch (SQLException e) {
            logger.debug("grepRegexp query executor", e);
91      }finally{
            try {
93              con.close();
            } catch (SQLException ex) {
95              logger.debug("closing connection : ", ex);
            }
97      }
        return listConcordances;
99  }
```

5.5 Conclusion

Dans cette cinquième partie, nous avons décrit et expliqué le fonctionnement de l'interface de recherche que nous avons développé au cours du stage. Cette interface répondait essentiellement aux besoins pour les chercheurs du projet de faire des requêtes sur l'ensemble du corpus CorpAfroAs, plutôt que simplement le sous-corpus sur lequel ils avaient travaillé, en particulier dans un souci de comparatisme.

Celui-ci s'inspire largement du module de recherche du logiciel ELAN et nous y avons ajouté un concordancier et un outil pour établir des listes de fréquences.

Si le concordancier permet une étude plus qualitative, le second outil permet en revanche une étude quantitative.

Nous allons voir dans la suite comment ces données quantitatives peuvent contribuer à améliorer l'annotation ou à faire une étude comparative portant sur les familles de langues, les langues ou encore les annotateurs.

Chapitre 6

Exploitation des listes de fréquences d'étiquettes

Si l'interface de recheche (locale sur ELAN ou web) a beaucoup été utilisée par les chercheurs pour détecter des phénomènes linguistiques (en particulier lorsqu'il s'agit de mettre à jour des constructions syntaxiques par la succession de certaines étiquettes morphosyntaxiques), l'idée de l'utilisation d'une telle interface à des fins d'exploration quantitative est une démarche beaucoup moins naturelle pour ces populations de chercheurs travaillant sur des langues peu documentées.

Dans cette partie de notre travail, nous allons donc tenter une approche plus quantitative en essayant d'exploiter les listes de fréquences des étiquettes obtenues à l'aide de l'outil web « Listes ».

6.1 Nombre d'occurrences des mots et des gloses

Il est à noter que les listes des nombres d'occurrences de mots et de gloses ont déjà été exploitées pour remplir le tableau récapitulatif 2.1. Un petit script perl permet de faire la somme de tous les nombres d'occurrences depuis les fichiers exportés au format csv. En lisant le fichier ligne à ligne, on récupère les nombres d'occurrences de la façon suivante :

```
1 if ($ligne =~ m/\"(\p{IsAlpha}+|\p{IsAlnum}+)\"\t(\d+)\t/) {
2   $freq = $2;
3   $total += $freq;
```

6.2 Vérification des erreurs d'annotations

Une application directe de l'analyse des listes de fréquences est la vérification de la conformité des annotations à la norme choisie à travers le système de gloses constitué collectivement et préalablement au travail d'annotations.

Dans ce travail, l'accès à la liste des refs est un atout dans la mesure où il permet à l'annotateur de retrouver rapidement l'emplacement de l'erreur pour la corriger.

Les listes de gloses non-conformes ont été établies à l'aide d'un script perl qu'on exécute avec les 3 arguments suivants :

- le fichier glosses2.txt contenant la liste des gloses officielles (une glose par ligne et obtenu en téléchargeant la liste depuis le site de CorpAfroAs (List of Glosses).
- le dossier des fichiers exportés depuis l'interface web,
- le dossier des fichiers résultats.

Ce script génère dans le dossier des fichiers résultats passé en dernier argument, les fichiers contenant la liste des mauvaises étiquettes avec les refs correspondantes).

La première étape du script consiste à obtenir, à l'aide d'une table de hash, la liste des gloses officielles « simples », c'est-à-dire que l'on découpe toutes les gloses composées au niveau du point (ex : PRO.CL donne 2 étiquettes PRO et CL) car dans la liste des étiquettes générées sur l'interface web, les étiquettes sont tronquées au niveau du point. Cependant, cette manipulation peut poser certains problèmes comme par exemple pour l'étiquette « V.A » qui signifie « verbe adjectival » mais pour laquelle la séparation des deux étiquettes n'a pas de sens car la glose « A » seule n'existe pas. Néanmoins, ce cas reste rare et en général, les compositions se font à partir d'étiquettes préexistantes.

Dans une seconde étape, on compare les étiquettes des fichiers exportés à celles de la liste que l'on vient d'établir pour récupérer la liste des gloses erronées ou du moins n'appartenant pas à la liste préexistante.

```
my $glosses = $ARGV[0];
my $dirRefs = $ARGV[1];
my $dirres = $ARGV[2];
open(GLOS, "<$glosses") || die "$!";

opendir(DIRREFS, $dirRefs) || return(0);
my @fichiersRef=readdir(DIRREFS); # le repertoire est lu, on peut le fermer
closedir(DIREFS);
opendir(DIRRES, $dirres) || return(0);

# On separe chaque glose de la liste officielle au niveau du point
my %hsplittedGlos;
my $sg;
my $ligne;
while ($ligne=<GLOS>) {
  $ligne =~ s/[\n\r]//g;
  my @splittedGlosses = split(/\./,$ligne);
  foreach $sg (@splittedGlosses) {
    $hsplittedGlos{$sg}++;
  }
}
my @tabSplittedGlos = keys (%hsplittedGlos);
# On parcourt tous les fichiers du repertoire contenant les refs
foreach my $fic (@fichiersRef) {
  if (($fic ne ".") && ($fic ne "..")) {
    open(FILEREF, "<$dirRefs/$fic") || die "$!";
    # on renomme le fichier resultat
    $fic =~ s/\.etiq/\.falsetiq/;
    open(FILERES, ">$dirres/$fic") || die "$!";
    while ($ligne = <FILEREF>) {
      if ($ligne =~ m/\"(.*)\"\t(\d+)\t\"(.*)\"/) {
        my $etiq = $1;
        # On nettoie les etiquettes des eventuels crochets residuels
        $etiq =~ s/\[|\]//g;
        my $refs = $3;
        # On ne prend pas les etiquettes qui commencent ou finissent par un nombre et celles qui appartiennent a la liste officielle
```

```
37          if (!($etiq =~ m/^\d+|\d+$/) && !grep(/^\Q$etiq\E$/,
                @tabSplittedGlos)) {
38            print FILERES "$etiq\t$refs\n";
39          }
40        }
41      }
42      close(FILERES);
43      close(FILEREF);
44    }
45  }
46  closedir(DIRRES);
```

Le tableau 6.1 présente une synthèse quantitative des non-conformités trouvées dans tout le corpus pour les couches `ge` et `rx` . La ligne « type » correspond au nombre d'étiquettes-type non-conformes alors que la ligne « occ. » correspond au nombre total d'étiquettes non conformes (autrement dit à la somme des nombres d'occurrences de chaque étiquette non conforme). La dernière ligne indique le pourcentage que représente la somme des lignes « ge (occ.) » et « rx (occ.) » par rapport au nombre total de gloses par langue :

Langues	kab	taq	hau	say	bej	gwd	tsb	wal	ary	ayl	pga	heb
ge (type)	12	2	11	11	3	11	4	22	19	0	5	23
ge (occ.)	105	5	1609	282	18	409	78	371	66	0	5	337
rx (type)	11	3	6	3	2	10	13	9	30	1	14	11
rx (occ.)	309	13	286	466	45	94	173	195	821	27	892	292
%	2,2	0,9	7	3	0,2	5	2,3	9	2,4	0,2	4,3	2,2

TABLE 6.1 – Synthèse des non-conformités du corpus à la date du 13/02/2013

Le pourcentage d'écart à la norme était donc à cette date compris entre 0.2 (pour le beja et l'arabe lybien) et 9% (pour le wolaytta).

Malgré l'ajout du module développé pour l'automatisation partielle de l'inter-alignement, on peut constater que le nombre d'erreurs reste important. En effet, la composition des étiquettes en rend le contrôle plus difficile car on ne peut pas implémenter simplement (liste déroulante) un vocabulaire contrôlé.

Du point de vue de la typologie des erreurs, on peut regrouper :

- les habitudes de gloses des chercheurs qui peuvent différer de celles préconisées par la liste officielle de gloses (ex : les noms propres sont glosés « N.PR » par certains chercheurs alors que la glose officielle est « NP »),
- les substitutions (ex : « PREQ » au lieu de « FREQ » pour fréquentatif),
- les transpositions (ex : « CWS » au lieu de « CSW » pour le code switching),
- les suppressions (ex : « IFV » au lieu de « IPFV » pour imperfectif),
- les ajouts (ex : « COLL » au lieu de « COL » pour collectif),
- les omissions de points pour séparer les étiquettes (ex : « FOBL » au lieu de « F.OBL »),
- les étiquettes de langues non harmonisées (ex : « FR », « FRE » et « FREN » utilisées pour le français)
- l'utilisation de majuscules à la place de minuscules, en particulier dans le cas de l'étiquette « DER » au lieu de « der » pour dérivation,
- les ajouts d'étiquettes spécifiques à une langue dont l'existence n'avait pas été identifiée lors du travail de réflexion sur la liste de gloses mais est apparue au cours de l'étape d'annotations. Il était, dans ce cas, possible à l'annotateur de faire ajouter cette glose à la liste officielle en renvoyant un formulaire à l'équipe coordinatrice du projet.

6.3 Analyse quantitative

6.3.1 Hypothèses de travail et outil d'analyse

Ce travail s'inscrit dans la perspective d'exploration typologique du projet mais aussi dans l'étude de l'impact des divergences de théories linguistiques sur la tâche d'annotation.

Les langues étudiées sont regroupées par famille génétique, donc sur des critères plus lexicaux, mais nous allons tenter de voir si des critères grammaticaux tels que les fréquences des étiquettes grammaticales peuvent nous permettre d'entrevoir un autre découpage dans ce corpus (cf 6.3.4).

De plus, dans le cas de l'arabe, le corpus présente deux variantes, le marocain et le lybien, dont la première a été annotée par 3 chercheurs. Bien que ces 3 chercheurs n'aient pas annoté les mêmes textes et que l'on ne peut donc pas véritablement parler d'accord inter-annotateur, il serait intéressant de voir si un système d'apprentissage est capable de les différentier.

En considérant les deux dialectes pour lesquels les sensibilités théoriques des annotateurs divergeaient, on peut également envisager de voir si ces analyses quantitatives permettent de rendre compte de ces divergences (cf 6.3.3).

Pour cette étude, nous nous servirons de l'outil open source de fouilles de données Weka [1] soutenu par l'université néo-zélandaise de Waikato.

Cet outil permet de réaliser assez simplement des analyses de fouilles de données à condition que ces données soient fournies dans un format adéquat.

Nous allons surtout nous intéresser à deux modules :

- « Classifiers » qui sont des outils de **catégorisation**. Cette tâche consiste à apprendre à ranger des éléments dans des catégories prédéfinies en fonction de leurs caractéristiques. Il s'agit là d'une analyse dite « supervisée ».
  Dans le cas de notre corpus, les caractéristiques sont les fréquences d'apparition des gloses. Le principe est donc d'essayer de trouver le lien entre ces caractéristiques et la catégorie à laquelle ils appartiennent (par exemple la langue ou annotateur).
- « Clustering » qui propose des modèles de **classification** (aussi appelés **segmentation**). Cette tâche consiste à regrouper les données automatiquement en classes, dont seul le nombre est fourni préalablement et arbitrairement par l'utilisateur.
  On parle dans ce cas d'analyse « non-supervisée ». Il est à noter que les résultats obtenus par cette technique sont très dépendants des algorithmes choisis pour la modélisation ainsi que du nombre de classes choisi.

6.3.2 Préparation des données

Afin de pouvoir procéder aux expérimentations, il est nécessaire de fournir au logiciel des fichiers au format ARFF (*Attribute-Relation File Format*).

Un fichier de ce format se compose de deux sections distinctes : l'**en-tête** et les **données**.

L'**en-tête** contient le nom de la relation et la liste des attributs suivis de leur type (ex : numérique, nominal, string...)

Dans l'exemple de la figure 6.1, la relation est appelée « alltags » et les attributs sont les abbréviations des gloses (ex : l'attribut « CONJ » correspond à « conjonc-

1. http://www.cs.waikato.ac.nz/ml/weka/ site consulté le 30/01/2013

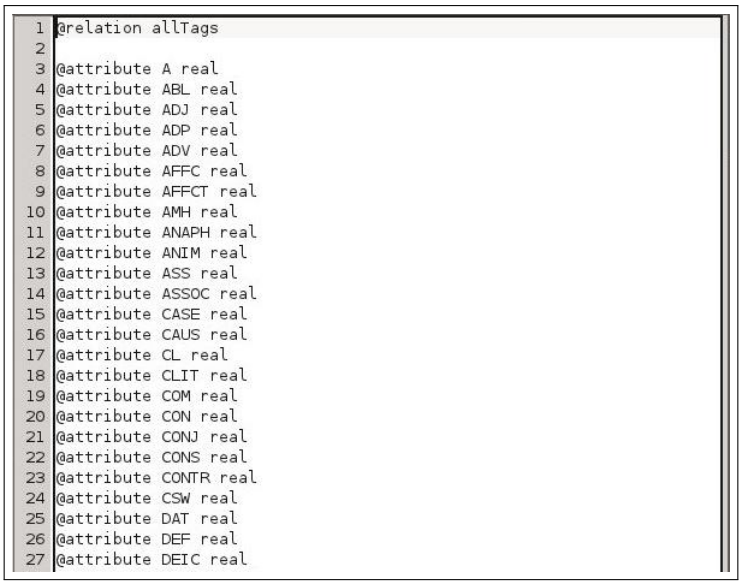


FIGURE 6.1 – Exemple d'en-tête de fichier WEKA

tion ») et leurs valeurs, qui sont des nombres d'occurrences ou des fréquences, sont de type « real », soit nombre réel.

La partie **données** est composée d'une simple ligne de déclaration contenant la chaîne de caractères « @data » suivie de la liste des instances.

Chaque instance correspond à une ligne de valeurs d'attributs séparées par des virgules. Il y a autant de valeurs que d'attributs déclarés dans l'en-tête et si des valeurs sont manquantes, elles sont remplacées par un point d'interrogation.

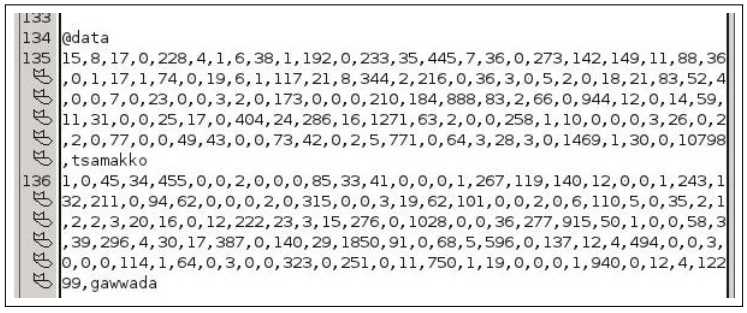


FIGURE 6.2 – Exemple de données de fichier WEKA

Ainsi dans l'exemple de la figure 6.2 qui est la suite de l'exemple de la figure 6.1, on peut lire qu'il y a 17 occurrences de l'étiquette « ADJ » ($3^{\text{ème}}$ attribut de l'en-tête) dans la première instance (qui correpond au tsamakko) et 45 occurrences de cette même étiquette dans la seconde instance (qui correspond au gawwada).

Les listes de fréquences des étiquettes sont générées par l'outil de recherche web et sont exportables au format csv. Elles sont ensuite transformées au format ARFF à l'aide d'une chaîne de traitement en perl dont voici les étapes :

- nettoyage des fichiers csv exportés pour supprimer les listes de ref, les guillemets autour des chaînes de caractères et les éventuels crochets résiduels,
- création de deux fichiers contenant les listes de toutes les étiquettes `ge` pour l'un et de toutes les étiquettes `rx` pour l'autre, utilisées dans l'ensemble des fichiers traités
- fusion sans doublon des deux listes précédentes. Cet ensemble permet d'avoir la liste des attributs à déclarer dans la partie en-tête du fichier ARFF

– création d'un fichier texte intermédiaire au format csv. La première ligne contient toutes les étiquettes séparées par des tabulations, puis un champ intitulé « total » pour la somme de toutes les fréquences et permettant de calculer les fréquences relatives. Dans le cas d'un apprentissage supervisé, on ajoute comme dernier champ, la variable cible c'est-à-dire le nom du paramètre qui va servir à la classification (ex : « annotateur » si l'on veut catégoriser par annotateur).
Sur chacune des lignes suivantes, on a les valeurs correspondantes de fréquences pour une instance (les variables prédictives), une instance étant une liste de fréquences établies sur un fichier ou un ensemble de fichiers d'annotations.
– création du fichier ARFF. Dans notre cas, les attributs sont tous de type `real` dans la mesure où il s'agit de nombres d'occurrences ou de fréquences. Pour ce qui est de la catégorie, il s'agit d'un type `nominal`, c'est-à-dire d'une valeur à choisir dans un ensemble donné. (ex : si l'on veut classer par famille de langue, on aura en face de l'attribut « famille », l'ensemble ainsi noté : « { cushitique,omotique,semitique,tchadique,berbere } »).

Dans la partie suivante, nous allons mener deux expérimentations simples qui vont pouvoir illustrer les deux techniques d'apprentissage.

6.3.3 Catégorisation : corpus des langues arabes

Pour l'apprentissage supervisé, nous allons étudier le groupe des langues arabes (marocain et lybien) qui a été annoté par 4 chercheurs (3 pour l'arabe marocain et un pour l'arabe lybien). Il y a 24 fichiers au total répartis en 17 fichiers d'arabe marocain et 7 fichiers d'arabe lybien.

Les catégories prédéfinies sont les initiales des annotateurs (AB, AV, DC et CP). Une instance, ou élément, est un fichier d'annotations et les caractéristiques sont les fréquences des étiquettes.

Plusieurs modèles sont disponibles pour cette tâche dans Weka, parmi lesquels on peut citer les modèles statistiques (classifieurs de Bayes, réseaux de neurones, machines à vecteurs de support...), les arbres de décision, le modèle des K-plus proches voisins...

Nous utiliserons les algorithmes d'arbres de décision (appelé `J48` dans Weka) avec les jeux d'apprentissage et de test ainsi qu'avec la validation croisée.

Dans le premier cas, un modèle est élaboré qui résume les relations entre les variables. Ce modèle est ensuite appliqué à la catégorisation de nouvelles données. Ici, nous créons tout d'abord un ensemble de 19 instances, pour l'apprentissage, constitué de fichiers pris dans le corpus de chaque annotateur et les 5 fichiers restant sont regroupés pour constituer l'ensemble de test.

Dans le second cas, l'ensemble des données est divisé en K parties, une des parties est utilisée pour le test et les (K-1) autres sont utilisées pour l'apprentissage. Ce processus est répété K fois afin que chaque partie soit utilisée une fois comme jeu de test.

Avec la validation croisée (K=5), nous obtenons les résultats suivants :

```
J48 pruned tree
------------------
FS <= 0
|   IMPV <= 0
|   |   REAL <= 0: CP (7.0)
|   |   REAL > 0: AB (8.0)
|   IMPV > 0: DC (5.0)
FS > 0: AV (4.0)
```

```
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances          21               87.5    %
Incorrectly Classified Instances         3               12.5    %
Kappa statistic                          0.8278
Mean absolute error                      0.0625
Total Number of Instances               24

=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.125      0.8       1         0.889      0.938    AB
                 0.8       0          1         0.8       0.889      0.9      DC
                 0.75      0.05       0.75      0.75      0.75       0.85     AV
                 0.857     0          1         0.857     0.923      0.929    CP
Weighted Avg.    0.875     0.05       0.892     0.875     0.876      0.913

=== Confusion Matrix ===
 a b c d   <-- classified as
 8 0 0 0 | a = AB
 0 4 1 0 | b = DC
 1 0 3 0 | c = AV
 1 0 0 6 | d = CP
```

Avec l'utilisation du modèle construit sur 18 instances, toujours en utilisant l'arbre de décision, on obtient les résultats suivants sur l'ensemble de test :

```
J48 pruned tree
------------------
DER9 <= 0
|   IMPV <= 0
|   |   REAL <= 0: CP (5.0)
|   |   REAL > 0: AB (6.0)
|   IMPV > 0: DC (4.0)
DER9 > 0: AV (3.0)

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances           5               83.3333 %
Incorrectly Classified Instances         1               16.6667 %
Kappa statistic                          0.76
Mean absolute error                      0.0833
Total Number of Instances                6
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.25       0.667     1         0.8        0.875    AB
                 1         0          1         1         1          1        DC
                 0         0          0         0         0          0.5      AV
                 1         0          1         1         1          1        CP
Weighted Avg.    0.833     0.083      0.722     0.833     0.767      0.875

=== Confusion Matrix ===
 a b c d   <-- classified as
 2 0 0 0 | a = AB
 0 1 0 0 | b = DC
 1 0 0 0 | c = AV
 0 0 0 2 | d = CP
```

On observe que le résultat avec la validation croisée est meilleur qu'avec les jeux d'apprentissage et de test et que dans les 2 cas, l'arbre de décision ne comporte que 3 tests sur la présence ou l'absence d'une certaine étiquette.

L'annotateur CP semble être identifiable avec la meilleure précision et le meilleur rappel (6 fichiers sur 7 bien classés dans la validation croisée et les 2 fichiers du jeu de test bien classés) ce qui rejoint le fait que le cadre linguistique théorique choisi pour l'annotation de l'arabe lybien différait quelque peu de celui choisi par les annotateurs de l'arabe marocain.

Ainsi on peut visualiser dans les deux arbres de décision que l'annotateur CP n'a utilisé ni l'étiquette `IMPV` (propre à l'annotateur DC et qui est une étiquette n'appartenant pas à la liste offcielle des gloses), ni l'étiquette `REAL`, correspondant à l'aspect « realis ». En revanche la racine de l'arbre diffère suivant le type de validation : dans le cas de la validation croisée, c'est l'étiquette `FS` (« false start ») qui a été retenue alors dans la validation par le jeu de test, c'est l'étiquette `DER9` (dérivation de forme IX) également absente du corpus d'arabe lybien.

On voit donc que l'algorithme s'appuie sur très peu de critères pour sa catégorisation mais il est toutefois difficile de conclure sur sa véritable performance au vu du petit échantillon dont nous disposons.

De plus on peut noter que le choix de la classe correspondant à l'annotateur DC est basé sur la présence d'une étiquette non officielle et qui pourrait simplement être une erreur d'annotation.

6.3.4 Segmentation : corpus entier

Pour l'apprentissage non-supervisé, nous allons étudier l'ensemble du corpus, toutes langues confondues, afin de trouver les groupes de fichiers présentant le plus de similarité au niveau de leurs caractéristiques, c'est-à-dire, dans notre contexte, les fréquences d'étiquettes.

Il existe là aussi plusieurs algorithmes de regroupement. On peut citer les algorithmes `k-means` (basés sur un calcul de distance euclidienne pour mesurer la similarité entre les différentes instances) et `Expectation-Maximisation` (basé sur une description probabiliste des clusters en terme de moyenne et d'écart-type pour les clusters).

Nous avons procédé à deux expériences : l'une (Expérience 1) réunissant tous les fichiers d'une langue pour n'en faire qu'une instance (en d'autres termes, une langue représente un individu et il y a donc 12 individus à classer), l'autre (Expérience 2) prenant tous les fichiers du corpus et en faisant autant d'instances à classer.

Expérience 1 : En utilisant l'algorithme « k-means », appelé `SimpleKMeans` dans Weka, sur les 12 instances et en lui imposant de segmenter selon 5 classes, on obtient la répartition suivante :

```
kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 286.5770363455217
Missing values globally replaced with mean/mode

Cluster centroids:
                                Cluster#
Attribute           Full Data          0          1          2          3          4
```

```
                          (12)        (1)        (5)        (1)        (4)        (1)
=============================================================================================
A                       0.0004     0.0002     0.0003          0     0.0001     0.0025
ABL                     0.0004     0.0017     0.0001          0          0      0.003
ABS                     0.0048          0     0.0115          0          0     0.0002
ABST                         0          0          0          0     0.0001          0
ABSV3                        0          0          0          0          0          0
AC                           0     0.0004          0          0          0          0
ACC                     0.0072     0.0282     0.0022          0          0     0.0473
.
.
.
X                       0.0001          0          0     0.0002     0.0001     0.0002
XX                      0.0011          0     0.0003     0.0021     0.0023          0
XXX                     0.0011          0     0.0013     0.0001     0.0016          0
YTH                     0.0003          0          0          0      0.001          0
total                18905.9167      30063      13667      27303   23769.25       6093
annotateur                 BC         MV         AM         BC         DC         AA
langue                  lybian       beja   tsamakko      hausa     lybian   wolaitta

Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       1 (  8%) <- beja (cushitique)
1       5 ( 42%) <- tsamakko (cushitique)
2       1 (  8%) <- hausa (tchadique)
3       4 ( 33%) <- lybian (sémitique)
4       1 (  8%) <- wolaitta (omotique)
```

La première observation que l'on peut faire est qu'une des 5 familles n'est pas représentée, il s'agit du groupe berbère avec le kabyle et le tamashek.

Il existe une option d'évaluation, `Classes to Clusters Evaluation`, qui, pendant l'étape de segmentation, ignore l'attribut classe (dans ce cas il s'agit de l'attribut « famille ») puis, dans la phase de test, assigne une classe au cluster en se basant sur la valeur d'attribut classe la plus représentée dans le cluster et enfin génère la matrice de confusion.

En imposant cette option dans notre expérience, on obtient la matrice de confusion suivante :

```
=== Model and evaluation on training set ===

Clustered Instances

0       1 (  8%)
1       5 ( 42%)
2       1 (  8%)
3       4 ( 33%)
4       1 (  8%)

Class attribute: famille
Classes to Clusters:
 cushitique
    berbere
   tchadique
   semitique
    omotique
```

```
 -  - -  - -
 0  1 2  3 4  <-- assigned to cluster
 1  2 0  0 0 | cushitique
 0  0 0  0 1 | omotique
 0  0 0  4 0 | semitique
 0  1 1  0 0 | tchadique
 0  2 0  0 0 | berbere

Incorrectly clustered instances : 3.0  25      %
```

Si l'on prétraite les données en appliquant une sélection d'attributs, on obtient une liste de seulement 11 attributs et les résultats sont meilleurs :

```
kMeans
======

Number of iterations: 4
Within cluster sum of squared errors: 2.498592113007888
Missing values globally replaced with mean/mode

Cluster centroids:
                          Cluster#
Attribute    Full Data          0          1          2          3          4
                  (12)        (4)        (2)        (1)        (4)        (1)
=============================================================================
ABS             0.0048          0     0.0287          0          0     0.0002
ADJ             0.0073     0.0029     0.0011     0.0044     0.0153     0.0082
IDEOPH          0.0005     0.0007          0     0.0034          0          0
LINK            0.0038     0.0106          0          0          0      0.003
M               0.0662     0.0804     0.1034     0.0001     0.0471     0.0771
OBJ             0.0052      0.004          0     0.0182     0.0038     0.0133
PFV             0.0265     0.0336     0.0531     0.0093     0.0153     0.0069
POS             0.0016          0     0.0033     0.0124          0          0
PREP            0.0228     0.0043      0.025     0.0223     0.0461          0
QNT             0.0011          0     0.0008          0     0.0028     0.0003

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       4 ( 33%)
1       2 ( 17%)
2       1 (  8%)
3       4 ( 33%)
4       1 (  8%)

Class attribute: famille
Classes to Clusters:
 cushitique
 berbere
 tchadique
 semitique
 omotique

 -  - -  - -
 0  1 2  3 4  <-- assigned to cluster
 3  0 0  0 0 | cushitique
 0  0 0  0 1 | omotique
 0  0 0  4 0 | semitique
```

```
 1   0 1   0 0 | tchadique
 0   2 0   0 0 | berbere

Incorrectly clustered instances : 1.0    8.3333 %
```

Ainsi on peut voir que la présence de trop nombreux attributs dégrade sensiblement la performance de l'algorithme de clustering et que finalement une liste restreinte d'étiquettes associées à leur fréquence semble permettre une segmentation satisfaisante.

Cependant, ces observations doivent être tempérées du fait de la faible quantité de données en jeu.

Expérience 2 : Nous avons réitéré cette expérience avec un jeu de données plus important. Nous avons créé 98 instances correspondant aux 98 fichiers actuellement annotés dans l'ensemble du corpus, toujours en fusionnant les gloses `ge` et `rx`.

Le nombre total de gloses utilisées sur l'ensemble du corpus est de 527. Fort de notre expérience précédente, nous procédons à une sélection d'attributs dans l'étape de preprocessing (Filter Choose/supervised/Attribute/AttributeSelection) ce qui réduit le nombre d'attributs à 55.

Voici la matrice de confusion obtenue pour le clustering de l'algorithme « k-means » avec un nombre imposé de 12 classes et l'option d'évaluation `Classes to Clusters Evaluation` :

```
kMeans
======

=== Model and evaluation on training set ===

Clustered Instances

 0       6 (  7%) <-- SAY
 1       1 (  1%) <-- No class
 2       4 (  4%) <-- HEB
 3       1 (  1%) <-- No class
 4      24 ( 26%) <-- ARY
 5       8 (  9%) <-- GWD
 6       4 (  4%) <-- PGA
 7      15 ( 16%) <-- TSB
 8       7 (  8%) <-- HAU
 9       2 (  2%) <-- KAB
10       2 (  2%) <-- No class
11      18 ( 20%) <-- BEJ


Class attribute: langue
Classes to Clusters:

  SAY  No class  HEB  SAY  ARY  GWD  PGA  TSB  HAU  KAB  No class  BEJ
  -    -        -    -    -    -    -    -    -    -    --        --
  0    1        2    3    4    5    6    7    8    9    10        11   <-- assigned to cluster
  0    0        0    0   17    0    0    0    0    0     0         0 | ARY
  0    0        0    0    7    0    0    0    0    0     0         0 | AYL
  0    0        0    0    0    0    0    0    0    0     0        18 | BEJ
  0    0        0    0    0    8    0    0    0    0     0         0 | GWD
```

```
  0   0   0   0   0   0   0   0   7   0    0    0 | HAU
  0   1   4   0   0   0   0   0   0   0    2    0 | HEB
  0   0   0   1   0   0   0   0   0   2    0    0 | KAB
  0   0   0   0   0   0   4   0   0   0    0    0 | PGA
  6   0   0   0   0   0   0   0   0   0    0    0 | SAY
  0   0   0   0   0   0   0   3   0   0    0    0 | TAQ
  0   0   0   0   0   0   0   7   0   0    0    0 | TSB
  0   0   0   0   0   0   0   5   0   0    0    0 | WAL
```

Incorrectly clustered instances : 19.0 20.6522 %

En utilisant l'algorithme EM sur le même jeu de données et la même sélection d'attributs, on obtient de meilleurs résultats :

```
=== Model and evaluation on training set ===

Clustered Instances

 0       5 (  5%) <-- WAL
 1      15 ( 16%) <-- ARY
 2       7 (  8%) <-- HEB
 3       1 (  1%) <-- No class
 4       8 (  9%) <-- AYL
 5       6 (  7%) <-- KAB
 6       7 (  8%) <-- HAU
 7       7 (  8%) <-- TSB
 8       4 (  4%) <-- PGA
 9       6 (  7%) <-- SAY
10       8 (  9%) <-- GWD
11      18 ( 20%) <-- BEJ

Log likelihood: 218.08616

Class attribute: langue
Classes to Clusters:

  WAL  ARY  HEB  No class  AYL  KAB  HAU  TSB  PGA  SAY  GWD  BEJ
  -    -    -    -        -    -    -    -    -    -    --   --
  0    1    2    3        4    5    6    7    8    9    10   11  <-- assigned to cluster
  0   15    0    1        1    0    0    0    0    0    0    0 | ARY
  0    0    0    0        7    0    0    0    0    0    0    0 | AYL
  0    0    0    0        0    0    0    0    0    0    0   18 | BEJ
  0    0    0    0        0    0    0    0    0    0    8    0 | GWD
  0    0    0    0        0    0    7    0    0    0    0    0 | HAU
  0    0    7    0        0    0    0    0    0    0    0    0 | HEB
  0    0    0    0        0    3    0    0    0    0    0    0 | KAB
  0    0    0    0        0    0    0    0    4    0    0    0 | PGA
  0    0    0    0        0    0    0    0    0    6    0    0 | SAY
  0    0    0    0        0    3    0    0    0    0    0    0 | TAQ
  0    0    0    0        0    0    0    7    0    0    0    0 | TSB
  5    0    0    0        0    0    0    0    0    0    0    0 | WAL
```

Incorrectly clustered instances : 5.0 5.4348 %

L'écart d'erreur entre les deux algorithmes est d'environ 15 points.

De plus, on remarque que dans le second cas, les 3 fichiers du Tamashek ont été classés avec ceux du Kabyle (les deux langues font partie de la famille berbère), un

fichier de l'arabe marocain a été classé avec ceux de l'arabe lybien (deux langues sémitiques) et il n'y a qu'un fichier de l'arabe marocain qui a été mis dans la classe sans nom, c'est-à-dire la classe pour laquelle il n'a pas été possible de décider d'une valeur pour l'attribut classe car la seule qui était possible (dans ce cas ARY) était déjà prise.

Dans la mesure où ces langues sont réparties en 5 familles, nous proposons maintenant au modèle de segmenter les données en 5 « clusters », en utilisant le même algorithme mais avec la classe « famille » au lieu de la classe « langue ». On obtient les résultats suivants :

```
=== Model and evaluation on training set ===

Clustered Instances

0      15 ( 16%)
1      23 ( 25%)
2       6 (  7%)
3      13 ( 14%)
4      35 ( 38%)

Log likelihood: 150.5048

Class attribute: famille
Classes to Clusters:
  cushitique  omotique  berbere  tchadique  semitique
  -   -   -   -   -
  0   1   2   3   4  <-- assigned to cluster
 15  18   0   0   0 | cushitique
  0   5   0   0   0 | omotique
  0   0   0   0  35 | semitique
  0   0   0  13   0 | tchadique
  0   0   6   0   0 | berbere


Incorrectly clustered instances : 18.0  19.5652 %
```

Ces résultats montrent que si les langues sémitiques, tchadiques et berbères ont pu être classées correctement, le programme semble avoir du mal à distinguer les langues cushitiques et omotiques (dans le corpus, seul le wolaitta fait partie de cette dernière). Nous pouvons signaler à ce stade que pour certains linguistes, les langues omotiques sont considérées comme la branche occidentale des langues cushitiques. Cependant ces quelques résultats obtenus sur un petit jeu de données ne permettent en aucun cas de conclure sur une plus grande parenté et des expériences complémentaires, sur une plus grande quantité de données, ou après découpage en unités plus petites (par exemple en découpant chaque fichier en sous-fichiers d'une centaine d'unités prosodiques) ou encore sur un jeu de données ne contenant des langues que de ces deux familles pourraient permettre d'explorer ces hypothèses.

Chapitre 7

Conclusion et perspectives

Partant de l'objectif de créer un environnement technologique qui simplifie et réduise le temps d'accès à un corpus et qui permette son questionnement et son exploitation analytique, nous avons développé un site dynamique en JSP hébergé au TGE Adonis et permettant à l'utilisateur de formuler des requêtes sur les données des fichiers ELAN créés par les chercheurs du projet, de faire des concordances ou encore d'obtenir des listes de fréquences.

Les résultats obtenus sont tous associés à l'unité de découpage qui les contient ainsi qu'au segment sonore correspondant ce qui permet de pallier le manque d'annotations concernant les aspects prosodiques, intonatifs ou tonals.

La particularité des données liées au mode de production de l'oral ainsi que le manque de ressources des langues du corpus rendent l'automatisation de leur traitement difficile. C'est la raison pour laquelle il n'est probablement pas pertinent de parler de linguistique de corpus dans le cadre de ce projet, cette approche supposant l'utilisation de requêtes qui portent sur des calculs de fréquence qui ne sont pas pertinents si les corpus sont de trop petite taille.

Comme le corpus obtenu au terme du projet n'est que de 71500 mots, répartis sur 12 langues, il ne permet donc pas l'approche quantitative propre à la linguistique de corpus.

Cependant, il a tout de même été possible d'amorcer un travail d'analyse quantitative en mettant en place une chaîne de traitement allant de la récupération de listes de fréquences à leur exploitation par un outil de fouilles de données et de procéder à quelques expériences qui suggèrent des différences d'annotations entre deux variantes de l'arabe (arabe marocain et arabe lybien) ainsi qu'une certaine « proximité » entre les langues des familles cushitiques et omotiques. Mais une analyse plus poussée faisant appel à des informations syntaxiques (tel que l'ordre des étiquettes et leurs cooccurrences) et un plus grand nombre de données ou un découpage différent des données actuelles seraient nécessaires pour explorer ces hypothèses.

Dans les développements que l'on peut suggérer concernant l'exploitation quantitative des résultats obtenus avec l'outil de recherche web, il serait intéressant d'explorer les questions suivantes :

- Comparaison des longueurs entre unités narratives et conversationnelles. La récupération n'est pour l'instant pas disponible mais serait simple à implémenter dans la mesure où la table `annotations` contient les indices temps absolus de début et de fin de chaque unité.
- Etude des durées des pauses. Celles-ci sont déjà récupérables avec la liste des annotations de type `mot`. Ces données pourraient être corrélées à celles du point précédent.
- Degré d'interaction d'une conversation. La difficulté de cette question réside

dans le choix du ou des indicateurs quantitatifs susceptibles de décrire au mieux cette interaction. Une piste pourrait être le nombre ou la longueur des chevauchements entre les prises de parole des différents interlocuteurs.

- Vérification de l'homogénéité de l'annotation. Si, dans le contexte de ce projet, un texte n'a pas donné lieu à l'annotation par plusieurs annotateurs et que par conséquent la question de l'accord inter-annotateur ne se pose pas, en revanche, on pourrait essayer d'évaluer la cohérence de l'annotation d'un annoteur, c'est-à-dire à l'intérieur d'une langue. En raison du faible nombre de fichiers par langue, on pourrait envisager de découper les fichiers en morceaux de taille comparable afin de créer artificiellement un plus grand nombre d'instances, puis de faire un apprentissage non supervisé sur l'ensemble des morceaux pour voir si les clusters obtenus correspondent ou non aux langues.
- En ce qui concerne la fouille de données à l'aide du logiciel Weka, la chaîne de traitement actuelle pourrait être simplifiée en réduisant le nombre de scripts perl à lancer pour obtenir les fichiers ARFF (5 pour le moment).

Pour ce qui est de l'outil web de recherche, les fonctionnalités suivantes seraient utiles à développer :

- Possibilité d'étendre le formulaire en rajoutant des lignes et des colonnes.
- Historique des requêtes. Pour l'instant, on ne peut qu'utiliser la touche « arrière » de la barre de navigation du navigateur sur la page des résultats de recherche, mais l'historique est perdu d'une session à l'autre.
- Ajout de la possibilité pour les propriétaires des fichiers d'annotations de faire des modifications directement sur le site. Pour l'instant, les modifications sont faites par le chercheur sur son fichier local qui est ensuite envoyé à l'administrateur du site qui procède à la suppression de l'ancien fichier et à l'ingestion du nouveau.
- Conversion et sauvegarde sur le serveur des fichiers ELAN au format TEI dans un souci de normalisation (difficile dans la mesure où l'on se rapproche des théories utilisées et des besoins spécifiques des travaux dans les formats des données) et afin de pouvoir bénéficier des avancements technologiques afférents à ce consortium.

Enfin, concernant l'outil d'annotation ELAN, les améliorations suivantes pourraient être envisagées :

- Annotation des relations syntaxiques. Actuellement, le découpage prosodique ne gêne pas le travail d'annotation dans la mesure où il n'y a pas de référence ou de renvois recouvrant plusieurs unités prosodiques. En revanche, cela serait le cas pour l'exploration d'un phénomène tel que la coréférence et l'organisation des couches adoptée pour le projet CorpAfroAs ne permet pas ce type d'annotation.
- Comparaison avec d'autres outils de requête tels que CQP (*Corpus Query Processor*) pour situer l'extensibilité de notre outil.
- Pour les prochains projets qui utiliseront ELAN-CorpA, on pourrait envisager pour la glose d'avoir le choix entre un champ de saisie manuelle et un champ avec menu déroulant. A l'usage, cela permettrait de voir si offrir cette possibilité aux chercheurs ne l'encouragerait pas à aller vers l'utilisation de la liste et donc d'assurer une glose plus homogène du corpus.

Bibliographie

[Bird and Liberman, 2001] Bird, S. and Liberman, M. (2001). A formal framework for linguistic annotation. Speech communication, 33 :23–60.

[Bird et al., 2002] Bird, S., Maeda, K., Ma, X., Lee, H., Randall, B., and Zayat, S. (2002). Tabletrans, multitrans, intertrans and treetrans : Diverse tools built on the annotation graph toolkit. Proceedings of the Third International Conference on Language Resources and Evaluation, Paris : European Language Resources Association.

[Chafe, 1993] Chafe, W. (1993). Prosodic and functional units of language. In Talking data : Transcription and coding in discourse research, pages 33–43. Jane A. Edwards and Martin D. Lampert (eds.).

[Creissels, 1991] Creissels, D. (1991). Description des langues négro-africaines et théorie syntaxique. Ellug.

[Dister and Simon, 2008] Dister, A. and Simon, A. (2008). La transcription synchronisée des corpus oraux. un aller-retour entre théorie, méthodologie et traitement informatisé. Arena Romanistica, 1/1 :54–79.

[Gumperz and Berenz, 1990] Gumperz, J. and Berenz, N. (1990). Transcribing conversational exchanges. Technical report, University of California at Berkeley.

[Izre'el, 2005] Izre'el, S. (2005). Intonation units and the structure of spontaneous spoken language : A view from hebrew. In Proceedings of the IDP05 International Symposium on Discourse-Prosody Interfaces.

[Schmidt, 2003] Schmidt, T. (2003). Visualising linguistic annotation as interlinear text. Arbeiten zur Mehrsprachigkeit, Folge B, 46 :1 ff.

[Schmidt et al., 2009] Schmidt, T., Duncan, S., Ehmer, O., Hoyt, J., Kipp, M., Loehr, D., Magnusson, M., Rose, T., and Sloetjes, H. (2009). An exchange format for multimodal annotations. In Kipp, M., Martin, J.-C., Paggio, P., and Heylen, D., editors, Multimodal corpora, pages 207–221. Springer-Verlag, Berlin, Heidelberg.