Alignement temporel entre transcriptions et audio de données de langue japhug

Cécile Macaire

Alignement temporel entre transcriptions et audio de données de langue japhug

Cécile Macaire[1 2]
(1) Langues et Civilisations à Tradition Orale (LACITO), CNRS-Sorbonne Nouvelle, France
(2) Laboratoire d'Informatique de Grenoble (LIG), CNRS-Université Grenoble Alpes, France
cecile.macaire@live.com

RÉSUMÉ

La collection Pangloss héberge un corpus de langue japhug : plus de 30 heures d'enregistrements audio (et dans une moindre mesure vidéo) accompagnés de transcriptions. La grande majorité des transcriptions, réalisées par le linguiste Guillaume Jacques, spécialiste de la langue, ne comportaient pas d'informations concernant l'alignement texte-son : les seuls points de référence étaient le début et la fin des enregistrements, dont la durée va de 22 secondes à 33 minutes. Le présent exposé présente la façon dont des chronocodes (balises indiquant l'alignement texte-son) ont été ajoutés au niveau de la phrase. La chaîne de traitement consiste en une application de l'outil d'alignement forcé MAUS, doublé d'une étape de vérification manuelle. Ce travail, réalisé sans intervention du linguiste, permet désormais la consultation phrase par phrase de la transcription, la citation d'une phrase spécifique par le biais d'une référence DOI, ainsi que l'utilisation des documents pour entraîner un modèle acoustique en vue de la transcription phonémique automatique de cette langue *à faibles ressources*. Cette tâche illustre l'utilité de collaborations entre linguistes et ingénieur·e·s informatiques pour la documentation linguistique.

ABSTRACT

Time alignment between transcriptions and audio of Japhug language data

The Japhug language corpus in the Pangloss Collection consists of a set of over thirty hours of audio recordings (and some video recordings) transcribed in the International Phonetic Alphabet (IPA). The vast majority of the transcriptions, made by linguist Guillaume Jacques, a specialist in the language, did not have detailed information about text-sound alignment : the only points of reference were the beginning and end of each of the recordings, which range from 22 seconds to 33 minutes in length. This presentation provides an account of the way in which detailed temporal information has been added (time codes indicating text-to-sound alignment at the level of the sentence) using the MAUS segmentation tool, and how the temporal information has been added to the transcripts. This task was entirely conducted without hands-on implication by the linguist. The result is that it is now possible (i) to consult texts sentence-by-sentence, (ii) to quote a specific sentence via a DOI, and last but not least, (iii) to use the documents for training an acoustic model for automatic phonemic transcription of untranscribed audio. The alignment task reported here illustrates the usefulness of collaborations between linguists and computer engineers for linguistic documentation.

MOTS-CLÉS : Alignement temporel, japhug, documentation linguistique.

KEYWORDS: Temporal alignment, Japhug, linguistic documentation.

2èmes journées scientifiques du Groupement de Recherche Linguistique Informatique Formelle et de Terrain (LIFT), Montrouge, France (Virtuel), 10-11 décembre 2020, pages 9–22,

1 Introduction

1.1 La collection Pangloss

La collection Pangloss (Michailovsky et al., 2014) [1] est une archive multimédia de langues en danger. Elle accueille des ressources linguistiques dans plus de 150 langues : enregistrements audio et vidéos accompagnés, pour certains, de transcriptions et annotations. Certains ont exprimé la crainte de voir les archives orales devenir des «cimetières de données» (“data graveyards” : Gippert et al., 2006, 4, 12-13) ; la collection Pangloss est bien plutôt un «jardin de données», dans lequel les corpus sont enrichis, au fil des ans, d'annotations de plus en plus complètes. Le dépôt de fichiers dans divers formats est accepté, y compris de simples fichiers texte brut ou des manuscrits scannés, mais l'objectif est de parvenir au stade de textes dotés d'un appareil critique complet (traductions, notes...), encodé en XML. Ainsi, le présent exposé relate l'ajout d'un alignement temporel entre transcriptions et audio pour le corpus de langue japhug, et en expose brièvement les enjeux.

1.2 La langue japhug et le corpus japhug

Le japhug est une langue sino-tibétaine parlée par environ 10 000 personnes dans la province du Sichuan (Chine). Le travail de Guillaume Jacques au sujet de cette langue depuis une vingtaine d'années a abouti à de nombreuses publications, dont une thèse (Jacques, 2004), un dictionnaire japhug-chinois-français (Jacques, 2016), et une étude phonético-phonologique (Jacques, 2019). Dans la collection Pangloss, il y a environ 400 enregistrements sonores en langue japhug, dont la durée va de 22 secondes pour le plus court à 33 minutes pour le plus long. La majorité des enregistrements font moins de 10 minutes. Trois locuteurs sont présents. Les enregistrements contiennent des contes, aussi bien que des témoignages au sujet de la culture locale : descriptions de la faune et de la flore, informations sur les techniques agricoles, la construction de maisons, le tissage... (Jacques, 2015). Avant les tâches décrites ici, les transcriptions réalisées par G. Jacques ne possédaient pas d'informations concernant l'alignement texte-son, hormis quelques documents qui possèdent un alignement phrase par phrase (par exemple le récit *Les trois sœurs* [2]). Il était de ce fait difficile d'accéder à l'enregistrement d'une phrase spécifique de la transcription. Un travail a été réalisé afin d'aligner les transcriptions avec les enregistrements audio, et d'ajouter ces informations temporelles (chronocodes) au niveau de la phrase.

2 Méthode

2.1 Outil d'alignement automatique : MAUS

Il existe divers outils pour effectuer un alignement forcé entre un fichier audio et sa transcription : SailAlign (Katsamanis et al., 2011), EasyAlign (Golman, 2011), etc. Le choix s'est porté sur MAUS (*Munich Automatic Segmentation*) (Schiel, 1999; Kisler et al., 2017), qui a été utilisé avec succès dans le contexte de la documentation linguistique (Strunk et al., 2014). MAUS calcule une segmentation

1. https://pangloss.cnrs.fr
2. https://doi.org/10.24397/pangloss-0003357

phonétique et un étiquetage, sur la base du signal audio fourni, et d'une transcription phonologique encodée selon des conventions spécifiques : en Alphabet Phonétique International, ou dans son équivalent ASCII selon les conventions SAMPA. La transcription phonologique doit être fournie dans un fichier BPF au format BAS Partitur (extension .par). MAUS permet de traiter un discours aussi bien lu que spontané d'une langue non répertoriée.

2.2 Chaîne de traitement

Dans le présent travail, il n'a pas été fait usage de fonctionnalités avancées de MAUS, telles que la prise en charge des variantes de prononciation. L'outil a été utilisé de façon relativement élémentaire. La figure 1 explique les étapes nécessaires à l'ajout des balises temporelles au niveau de la phrase dans les fichiers XML.

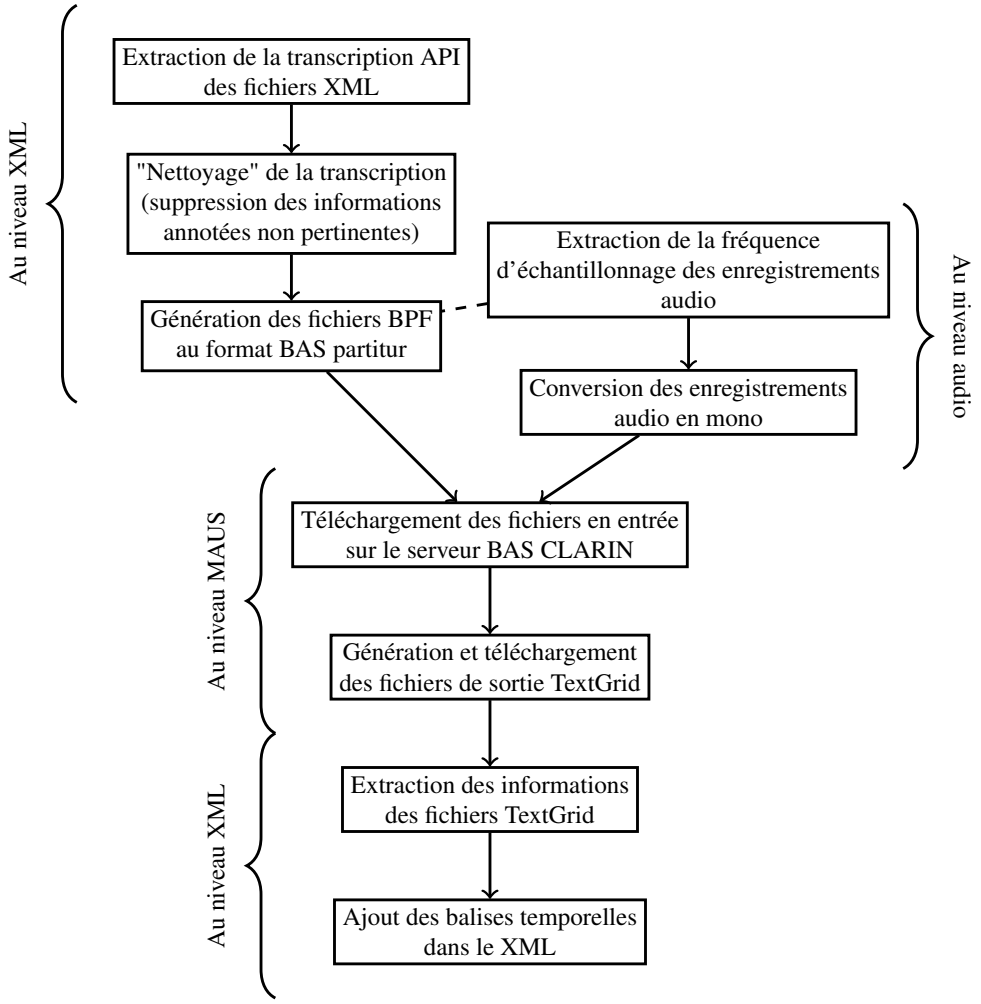

FIGURE 1 – Les différentes étapes réalisées par le script python.

En partant du corpus japhug, la première étape a consisté en l'extraction des données, ici, des transcriptions au format XML (cf. figure 2), et leur prétraitement : un toilettage pour supprimer certaines informations non prises en compte par l'outil et vérifier l'encodage de certains caractères.

Dans le corpus japhug, les mots empruntés au chinois mandarin sont notés en écriture chinoise, ce qui a le double avantage de les identifier avec grande précision et de les faire ressortir visuellement (ainsi que dans l'encodage). On voit ainsi dans la figure 3 le mot 机耕道 'sente pour tracteurs'. Cette pratique est claire et intuitive pour les linguistes spécialistes de langues de Chine, mais comme elle

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TEXT SYSTEM "https://cocoon.huma-num.fr/schemas/Archive.dtd">
<TEXT id="crdo-JYA_HIST-06-TSHAT-QAZO-KALAG" xml:lang="jya">
    <HEADER>
</HEADER>
    <FORM>
tshɤt qaʑo kɤ-lɤɣ ɲɯ-qha-nɯ ma
ɲɯ-rkaŋ-nɯ qhe aʁɤndɯndɤt ʑo sɯŋgɯ tu-ɕe-nɯ qhe
rdɤstaʁ ra pjɯ-tʂaβ-nɯ qhe
tɯrme tu-xtsɯɣ ɲɯ-ŋu
tʂu ra pjɯ-sɯ-Nqhi ɲɯ-ŋu qhe
jigendao 机耕道 nɯnɯ
mkhɯrlu tʂu nɯra pjɯ-sɯNqhi tɕe ɲɯ-qha-nɯ qhe
tɕe ɯ-kɯ-ftɕa ɲɯ-rkɯn
</FORM>
</TEXT>
```

FIGURE 2 – Transcription de l'histoire 6 *tshAt* [3] dans un fichier XML.

n'est pas explicitée dans la courte documentation disponible au sujet du corpus, elle constitue un obstacle pour les utilisateurs non spécialistes du domaine.

Les transcriptions pré-traitées ont ensuite été converties au format XML de la collection Pangloss : le découpage en phrases, indiqué à l'origine par des retours chariot, se trouve encodé (dans la logique du *texte structuré logiquement*) sous forme d'éléments <S> (*Sentence* : niveau de la phrase), puis alignées avec l'enregistrement audio. Enfin, ces informations temporelles récupérées ont été ajoutées dans les documents XML, sous forme de chronocodes au niveau de la phrase. La figure 3 montre le début du récit *tshAt* [3] obtenu après ce traitement. On y reconnaît les trois premières phrases du texte, cette fois-ci explicitement codées comme telles (éléments <S>) et chacune dotée d'une balise <AUDIO> contenant le début et la fin de l'intervalle audio correspondant. On relèvera également l'ajout d'une balise <TITLE> (on s'est contenté, pour l'heure, de recopier l'identifiant *a minima* que constituait le nom du fichier fourni par le linguiste) et d'une balise <SOUNDFILE> indiquant le nom du fichier audio correspondant.

```
<?xml version="1.0" ?>
<!DOCTYPE TEXT SYSTEM "https://cocoon.huma-num.fr/schemas/Archive.dtd">
<TEXT id="crdo-JYA_HIST-06-TSHAT-QAZO-KALAG" xml:lang="jya">
        <HEADER>
                <TITLE>crdo-JYA_HIST-06-TSHAT-QAZO-KALAG</TITLE>
                <SOUNDFILE href="hist-06-tshAt-qaZo-kAlAG.wav"/>
        </HEADER>
        <S id="S001">
                <AUDIO start="0.21" end="2.48"/>
                <FORM kindOf="phono">tshɤt qaʑo kɤ-lɤɣ ɲɯ-qha-nɯ ma</FORM>
        </S>
        <S id="S002">
                <AUDIO start="2.82" end="5.47"/>
                <FORM kindOf="phono">ɲɯ-rkaŋ-nɯ qhe aʁɤndɯndɤt ʑo sɯŋgɯ tu-ɕe-nɯ qhe </FORM>
        </S>
        <S id="S003">
                <AUDIO start="5.8" end="7.52"/>
                <FORM kindOf="phono">rdɤstaʁ ra pjɯ-tʂaβ-nɯ qhe</FORM>
```

FIGURE 3 – Début de la transcription de l'histoire 6 *tshAt* [3] annotée avec les balises temporelles phrase par phrase.

L'ensemble de ces étapes a été automatisé par un script python : **xml_info_japhug.py** (Macaire,

3. https://doi.org/10.24397/pangloss-0003420

2020) [4]. Il faut tout d'abord extraire les transcriptions. La fonction **extract_information(xml_file)** extrait les informations de la balise $< FORM >$. La transcription est une chaîne de caractères bruts, les phrases étant séparées par un saut de ligne. La chaîne de caractères est découpée par ligne, et nettoyée.

La langue Japhug n'est pas répertoriée dans MAUS, il a donc fallu utiliser le paramètre "Language independent" et son inventaire de caractères acceptés. Spécifiquement au corpus Japhug, de nombreux caractères spéciaux posaient problème car ils ne sont pas pris en charge dans l'inventaire utilisé. La solution a été de remplacer les caractères inconnus par MAUS par ceux répertoriés.

La notation de la langue japhug comporte des marques accentuelles, qui font référence à des événements prosodiques. Les formes accentuées, ó, ú, á, í, etc. sont remplacées par o, u, a, i. De plus, certains mots possèdent un préfixe et/ou suffixe séparé de la base par un '-'. Pour l'alignement phonémique, ces particules ont été "raccrochées" à la base pour ne former qu'un ensemble : [jɤ-ari-a] devient [jɤaria].

2.3 Informations écartées lors du traitement

Certaines informations ajoutées par le linguiste ne font pas partie du discours mais sont interprétées comme telles par MAUS. Celles-ci posent problèmes lors de la génération des fichiers BPF au format BAS partitur :

- Indications temporelles posées dans le fil du document : par exemple *(201)* correspondant à 2 minutes 01.
- Division de la transcription en parties (paragraphes) au moyen de suites de '=', 'x', '-', '*'.
- Caractères chinois : utilisation d'une librairie python hanzidentifier pour les identifier.
- Signes de ponctuations (' ?', ' !', '.', etc.)
- Autres informations : '(calque)', '(faute)', '(ideoph)', ' ???', 'X', etc. regroupés dans une liste *irrelevant_annotations* (cf. figure 9).

Lorsque le fichier est débarrassé d'informations autres que la transcription brute du discours, on peut créer les fichiers BPF au format BAS partitur.

2.4 Génération des fichiers BPF au format BAS partitur

La fonction **create_par(text, file, frame_rate)** (cf. figure 13) génère les fichiers BPF, fichiers en entrée pour MAUS. Le texte est découpé en mots, et chaque mot se voit attribuer un identifiant unique. La fréquence d'échantillonnage du fichier audio correspondant à la transcription est récupérée en parallèle grâce à la fonction **get_sampling_rate(file)** (cf. figure 11), et les fichiers son stéréo sont convertis en mono au moyen de la librairie python pydub, appelée par la fonction **convert_mono(path, path_export, file)** (cf. figure 12). Les fichiers sont ensuite téléchargés en entrée dans MAUS. Une requête HTTP télécharge les fichiers vers le serveur, effectue la segmentation et retourne un fichier TextGrid en sortie. Des paramètres sont à spécifier en entrée, comme montré en figure 4.

Dans le script python, cela correspond à la ligne de commande :

4. https://gitlab.com/macairec/alignement_texte_son_japhug

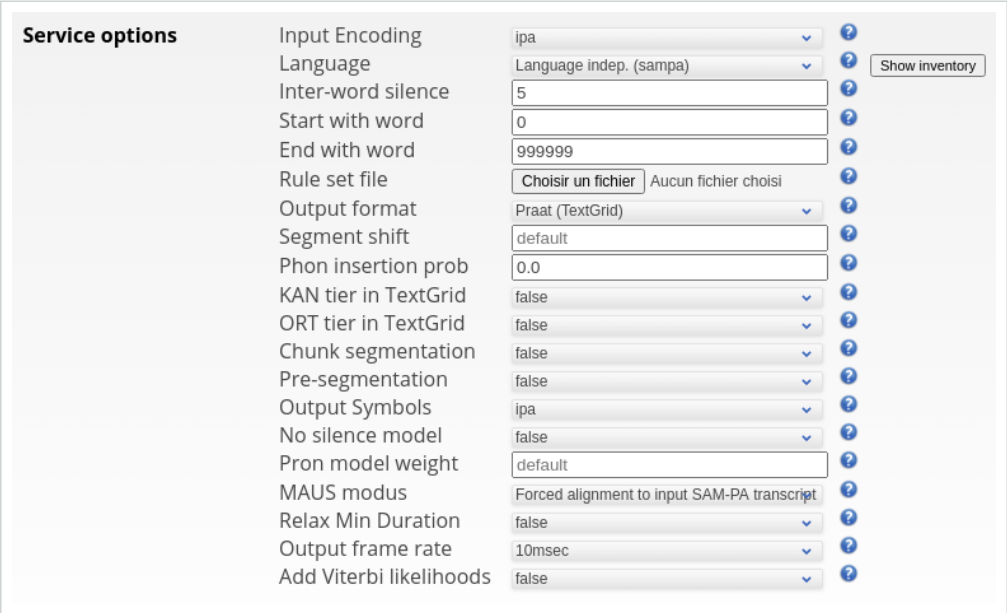

FIGURE 4 – Options en entrée de MAUS.

```
'curl -v -X POST -H \'content-type: multipart/form-data\' -F SIGNAL=@' \
+ path_wav + wav_file + ' -F LANGUAGE=sampa -F INSKANTEXTGRID=false -F ' \
'MODUS=align -F RELAXMINDUR=false -F OUTFORMAT=TextGrid -F ' \
'TARGETRATE=100000 -F ENDWORD=999999 -F STARTWORD=0 -F INSYMBOL=ipa -F ' \
'PRESEG=false -F USETRN=false -F BPF=@' + path_par + par_file + \
' -F MAUSSHIFT=10 -F INSPROB=0.0 -F INSORTTEXTGRID=false -F MINPAUSLEN=5 -F ' \
'OUTSYMBOL=ipa -F WEIGHT=default -F NOINITIALFINALSILENCE=false -F ' \
'ADDSEGPROB=false ' \
'\'https://clarin.phonetik.uni-muenchen.de/BASWebServices/services/runMAUS\''
```

FIGURE 5 – Ligne de commande dans le script python.

2.5 Extraction des chronocodes des TextGrid et ajout dans le fichier XML

Le but de ce script est d'ajouter des informations temporelles pour chaque ligne de transcription. Chaque ligne est identifiée grâce à un saut de ligne matérialisé par **\n**. La fonction **extract_info_textGrid(textgrid)** (cf. figure 14) crée un dictionnaire qui comprend deux éléments : un identifiant pour chaque phrase de transcription, et les balises temporelles de début et de fin d'apparition dans l'audio (exprimées en secondes). En effet, la fonction récupère le paramètre **xmin** du premier mot de la phrase, et le paramètre **xmax** du dernier mot de la phrase, reconnaissable par **\n**.

La figure 6 ci-dessous est un exemple d'un fichier TextGrid : ici la phrase commence par le mot [stu] à 0.44 seconde de l'audio, et se termine par le mot [babɯ] à 1.73 secondes.

La dernière étape consiste à ajouter les informations temporelles de début et de fin de phrase dans le fichier XML comprenant la transcription. La fonction **add_xml_info(timecode, wav_file, xml_file)** (cf. figure 15) récupère le dictionnaire précédemment créé, puis la transcription du fichier XML. A chaque ligne dans le fichier XML, on associe un identifiant, les balises temporelles, et la transcription. Certains caractères tels que les séparateurs de parties de transcriptions ('-', '=', '*') sont répertoriés dans des balises **NOTE**. La figure 7 illustre le fichier en sortie créé. Le script permet

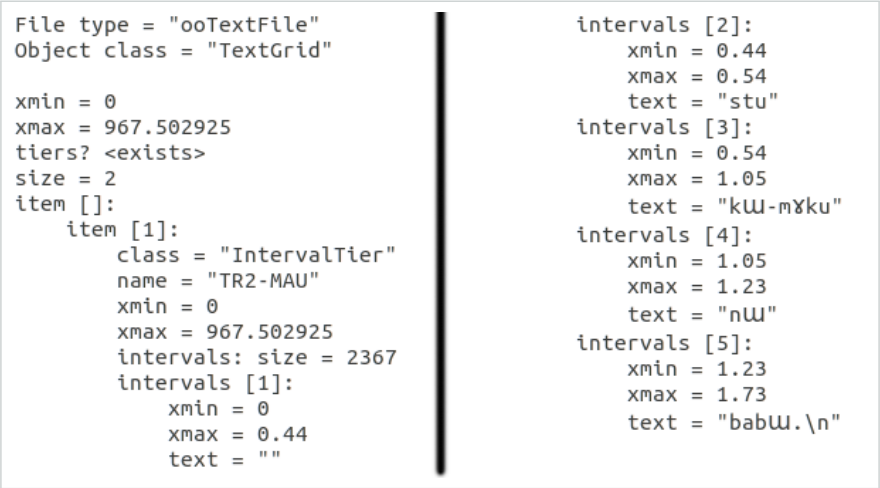

FIGURE 6 – Exemple d'un fichier TextGrid généré par MAUS (Schiel, 1999), (Kisler et al., 2017).

```
<?xml version="1.0" ?>
<!DOCTYPE TEXT SYSTEM "https://cocoon.huma-num.fr/schemas/Archive.dtd">
<TEXT id="crdo-JYA_HIST140425_KWFCI" xml:lang="jya">
        <HEADER>
                <TITLE>crdo-JYA_HIST140425_KWFCI</TITLE>
                <SOUNDFILE href="hist140425_kWfCi.wav"/>
        </HEADER>
        <S id="S001">
                <AUDIO start="0.53" end="4.23"/>
                <FORM kindOf="phono">kɯɕɯŋgɯ tɕe, kɯfɕi nɯnɯra wuma ʑo, nɤkinɯ,</FORM>
        </S>
        <S id="S002">
                <AUDIO start="4.23" end="7.0"/>
                <FORM kindOf="phono">nɯ kɯ-fse mɤ-kɯ-taŋ tu-nɤma-nɯ pjɤ-ŋgrɤl tɕe,</FORM>
        </S>
```

FIGURE 7 – Exemple d'un début de fichier XML avec les balises temporelles.

également d'ajouter deux informations dans l'en-tête : le titre du fichier et la référence du fichier audio correspondant. La seconde figure 8 ci-dessous montre l'utilisation d'une balise **NOTE** qui encapsule des informations portées par le linguiste mais qui n'appartiennent pas à proprement parler à la forme d'une phrase.

```
<NOTE message="================" xml  :lang="fr"/>
<S id="S022">
        <AUDIO start="75.3" end="76.61"/>
        <FORM kindOf="phono">tɕe mɤʑɯ phuɲi tu.</FORM>
</S>
<S id="S023">
        <AUDIO start="77.38" end="79.92"/>
        <FORM kindOf="phono">phuɲi nɯ li si ŋu tɕeri, mɤ-mbro.</FORM>
</S>
<S id="S024">
        <AUDIO start="80.26" end="80.99"/>
        <FORM kindOf="phono">tɕhi tɤ-mbro,  </FORM>
</S>
<S id="S025">
        <AUDIO start="80.99" end="83.07"/>
        <FORM kindOf="phono">tɯrme tɯ-fsu ɕaŋtaʁ tu-mbro mɤ-cha.</FORM>
</S>
```

FIGURE 8 – Partie d'un fichier XML annoté comprenant une balise NOTE.

3 Problèmes rencontrés

Quelques difficultés ont été rencontrées lors de la création du script et de son application. Tout d'abord, afin d'exclure les caractères non pertinents lors de la création des fichiers BPF, il a fallu regarder un à un chaque fichier qui posait problème lors du lancement de leur segmentation. Le principal défi a été de travailler sur un ensemble documentaire sans connaissance de la langue japhug, ni du chinois, et sans supervision préalable. Le corpus regroupant des ressources s'étalant sur une vingtaine d'années, il était difficile de comprendre d'emblée chaque information annotée. La présence d'un inventaire comprenant l'ensemble des normes et des informations supplémentaires annotées par le linguiste aurait permis un gain de temps considérable dans la construction du script. Un tel inventaire serait particulièrement utile pour les personnes faisant du TAL (Traitement Automatique des Langues). Les conventions varient d'un fichier à l'autre. La mise en place de normes d'écriture est un point essentiel, et permettrait une uniformisation du corpus.

3.1 Décalages entre annotation et audio

De plus, la transcription pouvait parfois présenter des disparités avec l'audio. Par exemple, certains mots dits n'ont pas été transcrits. La majeure partie du temps, cela correspondait aux mots [nɤkinɯ] et [tɕendɤre], plus communément appelés "gap-fillers". Ces choix effectués par le linguiste éloignent l'annotation du signal audio. De plus, certaines parties d'audio n'étaient pas transcrites, ou encore certaines transcriptions comprenaient une partie absente de l'audio (cf. fichier crdo-JYA_HIST-06-HUCHEMENTS1.xml par exemple).

3.2 Bruits parasites

Un point essentiel concerne les fichiers audio. Certains d'entre eux comportent des sons parasites, que MAUS considèrent comme de la parole. L'alignement résultant en est faussé. Il s'agit par exemple d'hésitations (« hum », « heu », etc.), de toussotements et éternuements, de chuchotements en arrière plan, de paroles d'une personne tierce dont le discours n'est pas retranscrit, de bruits de rue, ou de bruits parasites très légers en début et fin de fichier. Sur l'ensemble du corpus, près de la moitié des enregistrements audio présentaient de tels sons. La solution la plus efficace a été de "mettre en silence" les parties parasites. La fonction **mute_sound(path_wav, wav_file)** effectue cette opération ; il suffit de spécifier les intervalles de temps en secondes à traiter.

4 Conclusion et perspectives

Par l'utilisation d'un outil de segmentation automatique, MAUS, les informations temporelles pertinentes ont été extraites et ajoutées dans les transcriptions du corpus japhug. Les documents enrichis sont d'ores et déjà consultables via l'interface web de la collection Pangloss.

La méthodologie développée dans ce projet comporte plusieurs limitations, notamment car certains éléments des enregistrements audio (bruits parasites) et des transcriptions (annotations non adaptées) faussaient l'alignement produit, rendant nécessaire une intervention manuelle. Sur la totalité du corpus

(environ 400 fichiers), seuls 6 fichiers n'ont pas pu être corrigés, du fait que la transcription n'était pas complète.

Il n'a pas été mené d'évaluation objective de la qualité des résultats obtenus : cela aurait nécessité un point de comparaison (par exemple, un alignement temporel de référence, qui aurait été établi de façon entièrement manuelle, pour les mêmes documents). Néanmoins, de l'avis du linguiste déposant (Guillaume Jacques), les résultats obtenus au terme du processus sont concluants pour les données de langue japhug : aucune erreur d'alignement n'a été détectée.

4.1 Intérêt pour la consultation des données

L'existence de chronocodes au niveau de la phrase facilite une meilleure navigation au sein du corpus. En effet, tous les documents de la collection Pangloss ont été dotés d'identifiants DOI (Digital Object Identifier). A ce sujet, on consultera un article (Vasile et al., 2020) qui revient sur les problématiques d'identification de la ressource, et sur le contexte «socio-scientifique» actuel, pour mettre en perspective le choix d'attribuer un DOI à chaque document de la Collection Pangloss. Il est possible d'ajouter au DOI un numéro de phrase, après un dièse : par exemple, `https://doi.org/10.24397/pangloss-0003357#S10` amène droit à la dixième phrase (unité S) du document en question. Cela permet par exemple de citer un exemple précis dans une publication scientifique : les lecteurs peuvent accéder en un clic à l'exemple, dans son contexte d'origine. L'existence d'un chronocode au niveau de la phrase permet d'écouter le passage concerné, sans devoir naviguer tant bien que mal au sein d'un fichier audio dont la durée, rappelons-le, peut dépasser 30 minutes.

4.2 Intérêt pour l'utilisation du corpus japhug dans des expériences en Traitement Automatique des Langues Naturelles

Les transcriptions enrichies de chronocodes ont aussitôt été utilisées dans des expériences de reconnaissance automatique de la parole : transcription automatique de langues à faibles ressources. Ces expériences, qui s'inscrivent dans le fil de travaux exploratoires réalisés depuis plusieurs années (Adams et al., 2017, 2018; Michaud et al., 2018, 2019; Wisniewski et al., 2020a; Michaud et al., 2020), font l'objet d'un exposé lors des présentes Journées scientifiques (Wisniewski et al., 2020b).

Remerciements

Vifs remerciements à Séverine Guillaume, Laurent Besacier et Alexis Michaud pour avoir guidé le travail décrit ici, et au Groupement de Recherche CNRS LIFT (Linguistique Informatique, Formelle et de Terrain), qui m'a orienté vers ce travail dans le cadre d'une mis en relation entre linguistes et étudiants de Traitement Automatique des Langues.

Ce travail a reçu un soutien financier dans le cadre du projet « La documentation computationnelle des langues à l'horizon 2025 » (ANR-19-CE38-0015-04).

Références

Adams, O., Cohn, T., Neubig, G., Cruz, H., Bird, S., and Michaud, A. (2018). Evaluating phonemic transcription of low-resource tonal languages for language documentation. In *Proceedings of LREC 2018 (Language Resources and Evaluation Conference)*, pages 3356–3365, Miyazaki. https://halshs.archives-ouvertes.fr/halshs-01709648.

Adams, O., Cohn, T., Neubig, G., and Michaud, A. (2017). Phonemic transcription of low-resource tonal languages. In *Proceedings of the 2017 Australasian Language Technology Association Workshop (ALTA 2017)*, pages 53–60, Brisbane, Australia. https://halshs.archives-ouvertes.fr/halshs-01656683.

Gippert, J., Himmelmann, N., and Mosel, U. (2006). Language documentation : What is it and what is it good for. In *Essentials of language documentation*, volume 178, pages 1–30. Walter de Gruyter, Berlin.

Golman, J.-P. (2011). EasyAlign : An automatic phonetic alignment tool under Praat. In *Proceedings of InterSpeech 2011*, Florence. International Speech Communication Association.

Jacques, G. (10/05/2015). "Mon travail sur le japhug (1)," Panchronica,. https://panchr.hypotheses.org/264(ISSN2494-775X).

Jacques, G. (2004). *Phonologie et morphologie du japhug (rGyalrong)*. PhD thesis, Univ. Paris-Diderot/Paris VII Paris.

Jacques, G. (2016). Dictionnaire Japhug-chinois-français Version 1.1. https://halshs.archives-ouvertes.fr/halshs-01003734.

Jacques, G. (2019). Japhug. *Journal of the International Phonetic Association*, 49(3) :427–450.

Katsamanis, A., Black, M., Georgiou, P., Goldstein, L., and Narayanan, S. (2011). SailAlign : Robust long speech-text alignment. In *Proc. of Workshop on New Tools and Methods for Very-Large Scale Phonetics Research*, Philadelphia, PA.

Kisler, T., Reichel, U., and Schiel, F. (2017). Multilingual processing of speech via web services. *Comput. Speech Lang.*, 45(C) :326–347.

Macaire, C. (2020). Script python. https://gitlab.com/macairec/stage_lacito.

Michailovsky, B., Mazaudon, M., Michaud, A., Guillaume, S., François, A., and Adamou, E. (2014). Documenting and researching endangered languages : the Pangloss Collection. *Language Documentation and Conservation*, 8 :119–135. https://halshs.archives-ouvertes.fr/halshs-01003734.

Michaud, A., Adams, O., Cohn, T., Neubig, G., and Guillaume, S. (2018). Integrating automatic transcription into the language documentation workflow : experiments with Na data and the Persephone toolkit. *Language Documentation and Conservation*, 12 :393–429.

Michaud, A., Adams, O., Cox, C., and Guillaume, S. (2019). Phonetic lessons from automatic phonemic transcription : preliminary reflections on Na (Sino-Tibetan) and Tsuut'ina (Dene) data. In *Proceedings of ICPhS XIX (19th International Congress of Phonetic Sciences)*, Melbourne. https://halshs.archives-ouvertes.fr/halshs-02059313.

Michaud, A., Adams, O., Guillaume, S., and Wisniewski, G. (2020). Analyse d'erreurs de transcriptions phonémiques automatiques d'une langue « rare » : le na (mosuo). In Benzitoun, C., Braud, C., Huber, L., Langlois, D., Ouni, S., Pogodalla, S., and Schneider, S., editors, *Actes de la 6e conférence conjointe Journées d'Études sur la Parole, Traitement Automatique des Langues*

Naturelles, Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues, pages 451–462, Nancy, France. ATALA. https://hal.archives-ouvertes.fr/hal-02798572.

Schiel, F. (1999). Automatic phonetic transcription of nonprompted speech. In *Proc. Int. Cong. Phon. Sci*, pages 607–610.

Strunk, J., Schiel, F., Seifart, F., et al. (2014). Untrained forced alignment of transcriptions and audio for language documentation corpora using webmaus. In *LREC*, pages 3940–3947.

Vasile, A., Guillaume, S., Aouini, M., and Michaud, A. (2020). Le Digital Object Identifier, une impérieuse nécessité ? L'exemple de l'attribution de DOI à la Collection Pangloss, archive ouverte de langues en danger. *I2D - Information, données & documents*, 2 :156–175.

Wisniewski, G., Guillaume, S., and Michaud, A. (2020a). Phonemic transcription of low-resource languages : To what extent can preprocessing be automated ? In Beermann, D., Besacier, L., Sakti, S., and Soria, C., editors, *Proceedings of the 1st Joint SLTU (Spoken Language Technologies for Under-resourced languages) and CCURL (Collaboration and Computing for Under-Resourced Languages) Workshop*, pages 306–315, Marseille, France. European Language Resources Association (ELRA). https://halshs.archives-ouvertes.fr/hal-02513914.

Wisniewski, G., Michaud, A., Galliot, B., Besacier, L., Guillaume, S., Aplonova, K., and Jacques, G. (2020b). Ouvrir aux linguistes « de terrain » un accès à la transcription automatique. In *Actes des Journées scientifiques du Groupement de Recherche "Linguistique informatique, formelle et de terrain" (LIFT)*, Paris.

5 Annexe : script Python

```
['!)', '"', '"/???/', '(', '(!!!', '(...)', '(?)', '(??)', '(???)', '(agreement',
 '(anticausative)', '(antipassif)', '(associated', '(au', '(autoben)', '(autoben+anticaus)',
 '(autobenefactive)', '(benefactif)', '(calque', '(calque)', '(calque?)', '(cataphoric)',
 '(causatif)', '(causation)', '(causative)', '(causative,', '(cleft)', '(comme)',
 '(complementation', '(complementation)', '(conatif)', '(conative)', '(converb)', '(dative)',
 '(diminutive)', '(en', '(erreur', '(faute)6:40', '(faux:', '(floating)', '(focalisation,',
 '(focus)', '(generic)', '(hybrid?)', '(ideoph', '(ideoph)', '(ideoph.pause).', '(il', '(ils',
 '(imp?', '(imperative', '(in', '(indefinite)', '(infinitive)', '(intensif', '(inverse',
 '(inverse', '(inverse', '(inverse)', '(inverse+associated', '(inverse?', '(nominal', '(pas',
 '(passive)', '(permansive)', '(pluriel)', '(possessive)', '(redp)', '(reduplication',
 '(relation', '(relative)', '(revoir)', '(right', '(superlative)', '(trill)', ')', '*', ',',
 '-', '..', '...', '...,', '....', '/', '/.../', '//', '//?//', '/?/', '/???/', '050)',
 '0805)', '1015)', '200)', '2:00', '?', '?"', '??', '??', '???', '???,', '????', 'X', '\\b',
 'adjectives)', 'causee)', 'ce', 'dans', '(050', 'de', 'de', 'deux', 'deux', 'deux', 'disloc)',
 'emph)', 'entre', 'et', 'fact?', 'fur', 'hybrid)', 'inanimés).', 'inanimés).', 'inverse)',
 "l'original,", 'la', 'lui', 'main)', 'mange', 'manner)(testimonial)', 'mesure)', 'mismatch)',
 'modifiers,', 'motion)', 'motion,', 'n\'avaient', 'particulier)', 'parties', 'passif',
 'preventive)', 'quasi-clivée)', 'qui', 'redp)', 'relative,', 'rien', 'permansif?)', 'sous',
 'strategy)', 'that)', 'tombe', 'tout', 'traduction', 'à', '||||||', '(IMM)', 'khWjNga2',
 '(faute)', '||||||\"', '(demonstrative,', 'pronouns,', 'present)', '(kɯ?', '730)',
 'possessor)']
```

FIGURE 9 – Liste irrelevant_annotations.

```
def extract_information(xml_file):
    """Extract the transcription from xml file"""
    tree = et.parse(xml_file)
    root = tree.getroot()  # lexical resources
    transcript = root.find('FORM').text  # get content from tag FORM
    return transcript
```

FIGURE 10 – Fonction extract_information(xml_file).

```
def get_sampling_rate(file):
    """Get the sample rate of a wav file"""
    with wav.open(file, "rb") as wave_file:
        frame_rate = str(wave_file.getframerate())
    return frame_rate
```

FIGURE 11 – Fonction sampling_rate(file).

```
def convert_mono(path, path_export, file):
    """Convert wav stereo files into mono wav files"""
    sound = AudioSegment.from_wav(path + file)
    sound = sound.set_channels(1)
    sound.export(path_export + file, format="wav")
```

FIGURE 12 – Fonction convert_mono(path, path_export, file).

```
def create_par(text, file, frame_rate):
    """Create PAR file from the transcription previously extracted from xml file"""
    words_tr2 = [e + '\n' for e in text.split('\n') if e]  # split text by lines
    words_tr2 = [i.replace('\ufeff', '') for i in words_tr2]  # remove space at the beginning
    words_tr2 = [i.replace('>', '') for i in words_tr2]  # remove '>'
    digits = [re.findall(r'\(\d+.*\)', l) for l in words_tr2]  # find all the digits of the form '(201)'
    for k, p in enumerate(words_tr2):
        if len(digits[k]) != 0:
            words_tr2[k] = words_tr2[k].replace(digits[k][0], '')  # and delete them
    words_tr2 = [words_tr2[i].split() for i, j in enumerate(words_tr2)]  # split sentences into words
    words_tr2 = [[i for i in nested if (
            i not in irrelevant_annotations and not hanzidentifier.has_chinese(
        i) and 'xxxx' not in i and '---' not in i and '\\c' not in i and 'FICHIER' not in i and '=' not in i)] for
                 nested in
                 words_tr2]  # delete characters for MAUS
    words_tr2 = list(filter(None, words_tr2))  # remove blank lines
    words = [x for z in words_tr2 for x in z if x]  # extract all the words from sublist
    replace_characters = {'-': '', 'g': 'ɡ', 'ʁ': 'R', 'ɯ̽': 'ɯ', 'ɤ̞': 'ɤ', 'ó': 'o', 'ú': 'u', 'G': 'ɣ', '2': 'ø',
                          '4': 'ɾ', 'M': 'ɯ', 'T': 't', 'ʉ': 'y', 'û': 'u', 'é': 'e', 'B': 'b', 'à': 'a', 'á': 'a',
                          ', ': '', 'A': 'a', 'U': 'u', 'ɦ̃': 'h', '@': '', ' (': '(', ') ': ')', '6': 'ɐ', '9': 'œ',
                          '7': 'ɤ', '0': '', 'Z': 'z', 'ů': 'u', 'í': 'i', '1': 'ɨ', '3': ''}
    for k, v in replace_characters.items():
        words = [w.replace(k, v) for w in words]
    words = [w.translate(str.maketrans('', '', s.punctuation)) for w in words]  # delete punctuation
    phonemes = [i for i in words]  # separate each 'letter' of each word
    if len(words) != 0:
        with open(file, 'w+') as f1:  # create the par file
            f1.write('LHD: Partitur 1.3.1' + '\n' + 'SAM: ' + frame_rate + '\n' + 'LBD:' + '\n')
            for i, j in enumerate(words):
                char = ''
                for el in phonemes[i]:
                    char += el + ' '
                f1.write('KAN: ' + str(i) + ' ' + char + '\n')  # add KAN level
            for i, j in enumerate(words):
                f1.write('ORT: ' + str(i) + ' ' + j + '\n')  # add ORT level
            compt = 0
            for i, j in enumerate(words_tr2):  # add TR2 level with '\n' for end of sentence
                for k, m in enumerate(words_tr2[i]):
                    if k == len(words_tr2[i]) - 1:
                        m = m + "\\n"
                        f1.write('TR2: ' + str(compt) + ' ' + m + '\n')
                        compt += 1
                    else:
                        f1.write('TR2: ' + str(compt) + ' ' + m + '\n')
                        compt += 1
        return True
    else:
        return False
```

FIGURE 13 – Fonction create_par(text, file, frame_rate).

```
def extract_info_textGrid(textgrid):
    """Extract information from textGrid generate by MAUS"""
    data = {}
    grid = tg.TextGrid(textgrid)  # extract TextGrid
    time = []
    compt = 0
    for item in grid['TR2-MAU']: # extract timecodes
        if item.text != "" and '\\n' in item.text and len(time) == 0:
            time.append(item.xmin)
            time.append(item.xmax)
        else:
            if item.text != "" and '\\n' not in item.text and len(time) == 0:
                time.append(item.xmin)
            elif '\\n' in item.text and '.' in item.text:
                time.append(round(item.xmax, 2))
            elif '\\n' in item.text:
                time.append(item.xmax)
        if len(time) == 2:
            data[compt] = time
            time = []
            compt += 1
    return data
```

FIGURE 14 – Fonction extract_info_textGrid(textgrid).

```
def add_xml_info(timecode, wav_file, xml_file):
    """Add information for each transcription lines in xml file"""
    tree = etree.parse(xml_file)
    root = tree.getroot()  # lexical resources
    header = root.find('HEADER')
    time_begin = 0
    if header.find("TITLE") is None:  # add information into HEADER
        title = etree.SubElement(header, 'TITLE')
        title.text = xml_file.split('/')[7][:-4]
        sound_file = etree.SubElement(header, 'SOUNDFILE', href=wav_file)
    identifiant = 0
    has_s = False
    sentences = []
    if root.find('FORM') is not None:  # extract transcription from FORM
        transcript = root.find('FORM')
        trans = list(filter(None, transcript.text.split('\n')))
        identifiant = 0
    else:
        has_s = True
        trans = []
        sentences = root.findall('S')
        for i, j in enumerate(sentences):
            if j.find('FORM') is not None:
                identifiant = int(j.attrib['id'][1:])
                time_begin = float(j.find('AUDIO').attrib['end'])
            else:
                trans.append(j.text)
        sentences = [el for el in sentences if el.find('FORM') is None]
    for key, value in timecode.items():
        value[0] += time_begin
        value[1] += time_begin
    num = 0
    for i, j in enumerate(trans):  # for each line
        if '==' in j or 'xxxx' in j or '---' in j or '\\c' in j or '\\b' in j or (
                j.startswith('(') and j.isdigit() and j.endswith(')')) or 'khWjNga2' in j:
            nsmap = {"lang": "fr"}
            tag = etree.SubElement(root, "NOTE", nsmap=nsmap, message=j)  # add NOTE element for specific characters
        elif num < len(timecode):  # add timecode until the end
            if 0 <= identifiant < 9:
                if has_s:
                    tag = sentences[i]
                    tag.set('id', 'S00' + str(identifiant + 1))
                    tag.text = None
                else:
                    tag = etree.SubElement(root, "S", id='S00' + str(identifiant + 1))
             elif 9 <= identifiant < 99:
                 if has_s:
                     tag = sentences[i]
                     tag.set('id', 'S0' + str(identifiant + 1))
                     tag.text = None
                 else:
                     tag = etree.SubElement(root, "S", id='S0' + str(identifiant + 1))
             elif 99 <= identifiant < 999:
                 if has_s:
                     tag = sentences[i]
                     tag.set('id', 'S' + str(identifiant + 1))
                     tag.text = None
                 else:
                     tag = etree.SubElement(root, "S", id='S' + str(identifiant + 1))
             time = etree.SubElement(tag, 'AUDIO', start=str(round(timecode[num][0], 2)),
                                     end=str(round(timecode[num][1], 2)))
             form = etree.SubElement(tag, 'FORM', kindOf="phono")
             form.text = j
             num += 1
             identifiant += 1
         else:
             if not has_s:
                 tag = etree.SubElement(root, "S")
                 tag.text = j
    form = root.find('FORM')
    if form is not None:
        root.remove(form)
    tree.write(xml_file, encoding='utf-8', xml_declaration=True)  # create xml file with the new annotations
    dom = xml.dom.minidom.parse(xml_file)  # or xml.dom.minidom.parseString(xml_string)
    pretty_xml_as_string = dom.toprettyxml()  # correctly print xml file
    pretty_xml_as_string = '\n'.join([line for line in pretty_xml_as_string.split('\n') if line.strip()])
    lines = pretty_xml_as_string.split('\n')
    del lines[1:3]
    lines.insert(1, '<!DOCTYPE TEXT SYSTEM "https://cocoon.huma-num.fr/schemas/Archive.dtd">')
    for k, m in enumerate(lines):
        if '<AUDIO' in m:
            a = m.split(' ')
            if 'end' in a[1]:
                a[1:3] = a[1:3][::-1]
                a[1] = a[1][:-2]
                a[2] = a[2] + '/>'
                lines[k] = ' '.join(a)
    pretty_xml_as_string = '\n'.join(lines)
    pretty_xml_as_string = pretty_xml_as_string.replace("&quot;", "\"")
    with open(xml_file, "wb") as f:
        f.write(pretty_xml_as_string.encode('utf-8'))
```

FIGURE 15 – Fonction add_xml_info(timecode, wav_file, xml_file).