Université de Lorraine
IDMC

Master Thesis

Recognizing lexical units in low-resource language contexts with supervised and unsupervised neural networks

Author:
Cécile Macaire

Supervisors:
Guillaume Wisniewski
Alexis Michaud
Séverine Guillaume

A thesis submitted in fulfillment of the requirements for the degree of Master of Natural Language Processing

in the

Lacito, CNRS

March 1st - August 31st, 2021

Declaration of Authorship

I, Cécile MACAIRE, declare that this thesis titled, "Recognizing lexical units in low-resource language contexts with supervised and unsupervised neural networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITÉ DE LORRAINE
IDMC

Abstract

Lacito, CNRS

Master of Natural Language Processing

Recognizing lexical units in low-resource language contexts with supervised and unsupervised neural networks

by Cécile MACAIRE

Automatic Speech Recognition (ASR) has made significant progress thanks to the advent of deep neural networks (DNNs). In the context of under-resourced languages, for which few resources are available, spectacular achievements has been reported. ASR systems are a step forward for language documentation, as the annotation cost is considerably reduced for field linguists (manually annotated an audio file can take a tremendous amount of time), and the language is preserved and perpetuated through documentation. Previous 'standard' deep neural networks reached very good performances for phonemic transcription (such as with Kaldi and ESPnet approaches).However, these methods only rely on the phoneme-level. In this thesis, we explore recently published ASR approaches which have shown to be effective on low-resource languages to produce word-level audio-aligned transcriptions. The first approach, based on self-supervised learning, is a speech model that uses a Connectionist Temporal Classification (CTC). The second, entitled wav2vec-U, proposes a framework intended to build an ASR system in a fully unsupervised fashion. With few resources at our disposal, we try to assess the usability that can be made from dictionaries. We conducted experiments on two low-resource corpora, the Yongning Na and the Japhug from the Pangloss Collection. The experimental results from the first approach demonstrate powerful word-level transcriptions with competitive error rates. Preliminary results are reported on the second approach. By a coverage measure of dictionaries on the available transcriptions, we show that these resources are not yet usable in the conducted approaches.

Acknowledgements

I warmly thank Alexis Michaud, Guillaume Wisniewski, and Séverine Guillaume for their kindness, patience, knowledge, and pedagogy throughout this internship. Thanks for the precious advice, the proofreading of this report, and the corrections.

I would particularly like to thank Alexis and Séverine for trusting me enough to renew the experience for a second internship. Thanks to Guillaume for the numerous discussions, and all his help throughout the months.

Thanks to Minh Châu, Chiara, Fatima for their warm welcome at Lacito, and for the many meals spent together.

Thanks to Guillaume Jacques for being available to answer my questions, and for his interest in this project. His feedback was essential to complete this work.

Thanks to Oliver Adams, Dan van Esch, Ben Foley, and more generally, to the whole Elpis project team. Our exchanges and your advice have been very helpful.

Thanks to the teaching staff of the NLP master in Nancy, and more particularly to Claire Gardent, for having given me the opportunity to meet Alexis and Séverine, and for the loan of the computer.

Special mention to Elsa for taking the time to correct this thesis, and for her always relevant remarks.

Contents

List of Figures

List of Tables

List of Abbreviations

ASR	**A**utomatic **S**peech **R**ecognition
ANR	**A**gence **N**ationale de la **R**echerche
BN	**B**ottleneck **N**eck
CLD2025	**C**omputational **L**anguage **D**ocumentation 2025
CNN	**C**onvolutional **N**eural **N**etwork
CTC	**C**onnectionist **T**emporal **C**lassification
DNN	**D**eep **N**eural **N**etwork
ESPnet	**E**nd-to-end **S**peech **R**ecognition **net**work
et al.	**et alii**
GAN	**G**enerative **A**dversarial **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
GRU	**G**ated **R**ecurrent **U**nit
HMM	**H**idden **M**arkov **M**odel
INALCO	**I**nstitut **N**ational des **L**angues et **C**ivilisations **O**rientales
KIT	**K**arlsruher **I**nstitut für **T**echnologie
LIG	**L**aboratoire d'**I**nformatique de **G**renoble
LISN	**L**aboratoire **I**nterdisciplinaire des **S**ciences du **N**umérique
LSTM	**L**ong **S**hort-**T**erm **M**emory
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
RNN	**R**ecurrent **N**eural **N**etwork
SHL	**S**hared **H**idden **L**ayer
TCN	**T**emporal **C**onvolutional **N**etwork
TL	**T**ransfer **L**earning
WSJ	**W**all **S**treet **J**ournal
XLSR	**C**ross-lingual **L**earning of **S**peech **R**epresentations

Chapter 1

Context

1.1 Introduction

Over the last decade, Automatic Speech Recognition (ASR) has made significant progress. One dimension of progress comes from the advances of deep neural networks (DNNs), specifically Recurrent Neural Networks (RNNs) and Bi-Directional Recurrent DNNs. Another comes from the enhancement of model architectures, with the advent of end-to-end speech recognition models and Transformers (Vaswani et al., 2017). ASR systems are now part of our everyday life, from voice assistants to the dictation of text messages, e-mails, or home assistants (Szymański et al., 2020). These systems are trained on a large amount of annotated data. Librispeech for instance is a corpus of 960 hours of audio and associated transcriptions. The best current systems obtain an error rate of less than 10% on benchmark datasets (Librispeech, WSJ, Callhome, Fisher, etc.) (Szymański et al., 2020).

Spectacular results are also observable for under-resourced languages, for which few resources are available (Besacier et al., 2014; Esch, Foley, and San, 2019). The term "under-resourced languages" (Krauwer, 2003; Berment, 2004) refers to a language lacking a unique writing system or viable orthography, having a limited presence on the web, an absence of linguistic expertise, and limited electronic resources (monolingual corpus, dictionary, transcribed audio, etc.) (Besacier et al., 2014). Applying Automatic Speech Recognition systems to this type of data has two main objectives. The first concerns documentation from the point of view of language preservation and perpetuation. There is indeed an urgent need to document the world's declining linguistic diversity (Besacier et al., 2014; Thieberger, 2017; Littell et al., 2018; Esch, Foley, and San, 2019). Languages become threatened by various factors (most saliently, the dominance of another language for economic, societal, or political reasons). The second concern is the workload of field linguists: using ASR can help with the annotation of audio files, which can take thousands of hours of work depending on the size of the corpus if done manually. In 2017, a survey was conducted on 51 linguists to get a picture of linguists' practices during transcription. The results showed that one minute of audio data takes an average of 40 minutes to transcribe, which varies according to the difficulty associated to the file (Foley et al., 2018). If field linguists could be freed from (at least some of) the burden of repetitive tasks of data entry, they could then devote the time thus saved to other equally valuable tasks (linguistic analysis, descriptive work).

Previous studies have shown that 'standard' deep neural networks can achieve very good results for phonemic transcription (Michaud et al., 2019; Wisniewski, Michaud, and Guillaume, 2020). This task, which is much simpler than full-fledged ASR, consists in predicting the sequence of phonemes and tones contained in an audio file. These models only require about ten hours of annotated recordings to reach ceiling or near-ceiling accuracy (Michaud et al., 2019; Wisniewski, Michaud, and Guillaume, 2020) – remembering that phonemic transcription can never reach 100% accuracy, given the variability of phonetic realizations in speech: not all phonemes contained in a word-level transcription are actually present in the speech signal. The architecture used in these cases is based on the encoder-decoder type trained with a Connectionist Temporal Classification (CTC) criterion (Graves et al., 2006) (see Section 2.3.5).

However, these methods only rely on the phoneme-level. Word-level transcriptions are the next step for the ASR community for endangered languages. Specifically for linguists, retrieving word-level transcriptions is a way to get a transcription that is as close as possible to the audio (occasional repetitions are not always recognized, as well as word fillers). The linguists will be able to use the transcriptions to document the language (for instance by producing word-level glosses or extracting dictionaries). Finally, it will be easier to correct the produced transcriptions, as it will only be necessary to correct the word boundaries, and not to build words from the phonemes.

Predicting sequences of words is a tall order within the ASR community in the case of low-resource languages. Precisely, in an high-resource language context, moving from a phonemic level to a word level can be resolved thanks to the use of a language model. However, this approach can not be considered here due to the lack of textual content. The main objective of this work is to relax the previously presented methods that work on the phoneme level to transcribe audio recordings into higher-level entities, here words. We will have to limit ourselves to the resources at our disposal, i.e. a small volume of transcriptions (between 5 to 30 hours of transcribed speech), and dictionaries (with around 5,000 lexical entries each).

To address this challenge, we are considering two low-resource languages, Yongning Na and Japhug. We explore two complementary approaches that have proven successful in low-resource contexts. The first, `XLSR` (Baevski et al., 2020; Conneau et al., 2020), is a speech model that uses a CTC, and the second, `wav2vec-U` (Baevski et al., 2021), is a framework that builds speech recognition systems that require no transcribed data at all. We describe, for both models, a method, intended for field linguists, but also for computer engineers to reproduce the experiments carried out. Finally, we try to determine whether the information gathered by field linguists to describe the language (dictionaries, grammars, ...) can be used to compensate for the small amount of available data.

The report is organized as follows. Chapter 1 presents the subject and the background of the project. In Section 1.2, the research lab Lacito where the internship took place is introduced as well as the *Computational Language Documentation by 2025* project (see Section 1.2.2). The technical choices come next in Section 1.2.3. The first approach, consisting in fine-tuning the `XLSR-53 wav2vec 2.0` model, is explained in Chapter 2. The related experiments are displayed in Chapter 3. The second approach based on the `wav2vec-U` framework as well as the preliminary experiments are in Chapter 4. We continue with the work on dictionaries in Chapter 5. We end this work by a discussion and we conclude in Chapter 6.

1.2 Work Environment

The internship took place at Lacito, *Langues et civilisations à tradition orale*, a research unit of French CNRS devoted to the documentation, description and analysis of languages worldwide, with an emphasis on unwritten languages, undocumented languages, and endangered/minority languages generally. Supervision was carried out by

- Guillaume Wisniewski, assistant professor at Université de Paris and member of the *Laboratoire de Linguistique Formelle* research unit (LLF, CNRS),
- Alexis Michaud, CNRS researcher,
- and Séverine Guillaume, engineer in computer science at Lacito, CNRS.

This work is part of the "Computational Language Documentation by 2025" ANR project (CLD2025, ANR-19-CE38-0015-04), the main objective of which is to use computational methods to facilitate the task of documenting endangered languages. This section will present the Lacito (Subsection 1.2.1) and the CLD2025 project (Subsection 1.2.2). It will also describe the equipment and tools used in this work.

1.2.1 Lacito

Lacito is a research department based in Villejuif. It is one of the laboratories of CNRS, French National Centre for Scientific Research, and is affiliated to two French universities: Sorbonne Nouvelle and INALCO (Institut national des langues et civilisations orientales). The laboratory is currently directed by Alexis Michaud and has over 60 members (including researchers, Ph.D. students, affiliate members, engineers and technicians).

Founded in 1976, Lacito aims at exploring the linguistic diversity across the five continents by carrying out fieldwork investigation among speaker communities. It also seeks to describe the languages in their broader social, geographical, and historical dynamics. Specifically, their major interest concerns the under-resourced languages of the world for which very few speakers are known (in opposition to the under-resourced languages spoken by many). Their work focuses on different aspects:

- the immersive fieldwork in language communities (documentation and description of their linguistic practices);
- the academic research in linguistics, language typology and linguistic anthropology;
- the university teaching, Masters and Ph.D. supervisions;
- the long-term archiving of language data and corpora;
- and the communication and awareness-raising for the general public and scientific communities.

1.2.2 Computational Language Documentation by 2025 project (CLD2025)

The emergence of Machine Learning (ML) tools (artificial neural networks) and their performance year after year can contribute efficiently to the execution of specific tasks for language documentation: automatic transcription of audio recordings, automatic glossing of texts, automatic word discovery. The promise of ML based models is to reduce the annotation burden, which results in a time-saving. For example, manually translating tens of hours of speech word by word can take thousands of hours of work. These time-consuming tasks can easily be automated which thus facilitates the field linguists' work.

Natural Language Processing (NLP) is still not widely used in linguistic literature. It can be explained by the novelty of these methods, the lack of easy-to-use interfaces, and the small number of studies showing its effectiveness when few resources are available. The main objective of the CLD2025 project is to implement techniques, models, and interfaces for the documentation and description of languages, especially endangered ones. The project will involve interdisciplinary collaboration between computational scientists and field linguists and focus on the usability of the tools and methods developed.

It is in this perspective that the Elpis project (Foley et al., 2018) is currently jointly developed by a team based in Australia[1] and the Lacito. Elpis is a graphical interface which allows linguists and language workers with basic computational knowledge to build their own speech recognition models. It relies on the `Kaldi` ASR toolkit, and recently on `ESPnet` (Adams et al., 2020). The goal is to save time to linguists by taking away the technical burden of training an ASR model from scratch.

[1]from the Australian Research Council Centre of Excellence for the Dynamics of Language.

The project coordinator of the CLD2025 project is Gilles Adda, from the LISN ("Laboratoire Interdisciplinaire des Sciences du Numérique") and involves the Lacito, LISN, LPP ("Laboratoire de Phonétique et Phonologie"), LIG ("Labratoire d'Informatique de Grenoble"), KIT ("Karlsruher Institut für Technologie") and EmpSprWiss Universität Frankfurt / Institut für Empirische Sprachwissenschaft in Germany.

1.2.3 Technical choices

The internship has involved the implementation and the training of deep learning models. For this purpose, we used, during the first month of the internship, Google Colaboratory[2] which enables to write and execute Python code in a browser. The primary advantage of this platform is the free access to GPUs. We then chose to move on to the Huma-Num servers, which provide access to a GPU Ampere card (NVIDIA A100-PCIE-40GB MIG 4g.20gb) and initialized with a basic version of Python (3.8), via a ready-to-use Anaconda Environment. The deep learning library PyTorch[3] and data science libraries such as Pandas[4] and NumPy[5] were used. The scripts and information related to the project are available in a GitHub repository[6].

[2]`https://colab.research.google.com`
[3]`https://pytorch.org/`
[4]`https://pandas.pydata.org/`
[5]`https://numpy.org/`
[6]`https://github.com/macairececile/internship_lacito_2021`

Chapter 2

First approach: Fine-tuning XLSR-53 wav2vec 2.0 model

This chapter presents a novel approach, called `XLSR`, introduced in Conneau et al., 2020. The goal of this approach is to extract new types of input vectors for acoustic models from raw audios thanks to pre-training and self-supervised learning. The process is divided into two parts: the first part of the model learns cross-lingual representations of the audio signal from a large amount of unlabeled data by pre-training a single model on audio recordings of several languages; the second part uses these representations to fine-tune a model for a specific language on a small amount of labeled data. Since a pre-training step is performed, less data is needed for fine-tuning.

The objective of this section and Chapter 3 is to determine to what extent the use of a pre-trained model on a large amount of data, followed by a fine-tuning on the two low-resource corpora studied[1], allows us to recognize word entities, and to obtain a lower error recognition rate compared to previous studies based on `Kaldi` and `ESPnet` (Adams et al., 2020).

To answer this question, we define some of the terms (see Section 2.1) before explaining state-of-the-art approaches (see Section 2.2) as well as the proposed `XLSR` architecture (see Section 2.3). Chapter 3 will present the experiments and the obtained results.

2.1 Self-supervised learning, pre-training & fine-tuning

The principle of self-supervised learning is to acquire background knowledge from a large database and use it to recognize and understand patterns from a narrowed, less common problem. Self-supervised learning is a popular topic within the NLP community, including approaches such as `word2vec` (Mikolov et al., 2013b), `GloVE` (Pennington, Socher, and Manning, 2014), `XLM-R` (Conneau et al., 2019), `BERT` (Devlin et al., 2018) and others. Recently, a novel architecture entitled Transformers was presented in the paper “Attention Is All You Need” (Vaswani et al., 2017). It solves sequence-to-sequence (seq2seq) tasks by transforming a large quantity of unstructured data into interpretable information sequentially. This architecture was first successfully introduced in NLP tasks such as machine translation and text summarization. Several studies showed the applicability of the Transformers architecture in other domains, such as speech recognition (Dong, Xu, and Xu, 2018; Karita et al., 2019) and image processing (Chen et al., 2021).

Suppose we have an input sequence (words, speech, frames, etc.), the Transformers architecture objective is to associate each element to a vector representation. The way to define the representations is to leave aside some part of the sequence (called the hidden parts) with a mask and, then, try to recover the past or the future hidden sections from the current ones

[1] Yongning Na and Japhug.

(the observed data). In the case of a speech audio sequence, Transformers model learns representations of audio frames. Figure 2.1 presents a speech file cut into n frames, containing 10 milliseconds each.

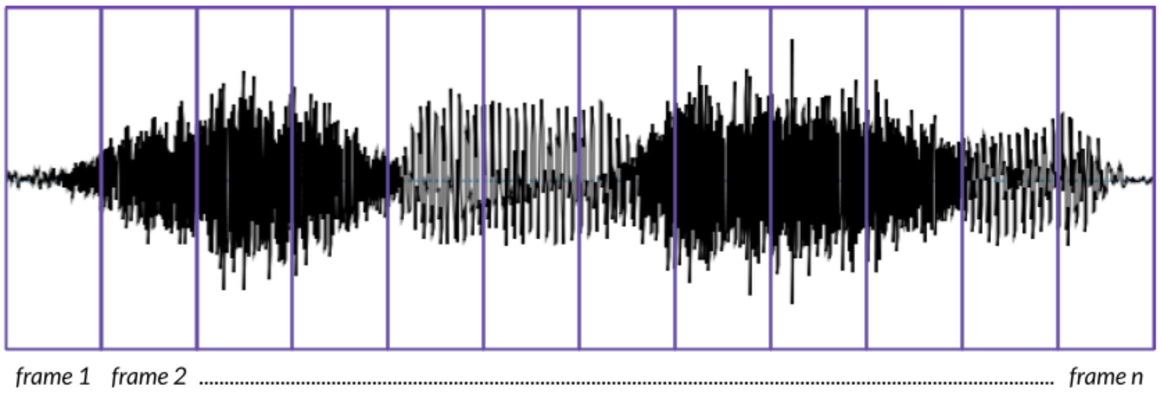


FIGURE 2.1: Schema of a speech file cut into n frames, representing 10 milliseconds each.

In the self-supervised learning approach, the first step is called *pre-training*: the model uses large amounts of unlabeled data to build a specific model (which predicts the parts that have been masked). In doing so, the model learns robust representations. Theses representations are then used in a second step that aims at *fine-tuning* them for a particular task, such as classifying the topic of a text. In this step, labeled data are used to "adapt" the model thanks to a standard supervised learning approach (i.e. minimization of a loss function over a training set).

Before explaining the `XLSR` model, we present the different state-of-the-art approaches.

2.2 State-of-the-art

Current models in speech processing require large amount of labeled data[2] to reach good performances (Amodei et al., 2016). The pre-training of neural networks, is a viable solution to overcome the scenario of restricted labeled data (Schneider et al., 2019; Zhang et al., 2021). Learning representations of speech via pre-training can be done in a supervised (with labeled data)[3] or unsupervised[4] fashion (with unlabeled data).

Table 2.1 summarizes the state-of-the-art approaches as well as the corresponding amount of unlabeled and labeled data used for the training step.

2.2.1 Supervised pre-training

In Heigold et al., 2013, a novel multilingual approach based on deep neural networks (DNNs) is presented which contains a shared feature extraction module to learn speech representations on a large supervised corpus. Specifically, the multilingual architecture is made of a language-independent feature extraction and language specific classifiers. Experiments were conducted by training the network on eleven Romance languages with a total amount of 10k hours of labeled data. It was shown that using multilingual training and transfer learning on

[2]The order of magnitude here is thousands of hours of labeled data.
[3](see Section 2.2.1)
[4](see Section 2.2.2)

Method	**Unlabeled data size**	**Labeled data size**
Supervised pre-training		
DNN multilingual (Heigold et al., 2013)	-	10k h
`SHLMDNN` (Huang et al., 2013)	-	460 h
Multi-lingual Bottleneck features (Veselỳ et al., 2012)	-	125.9 h
Unsupervised pre-training		
`CPC` (Oord, Li, and Vinyals, 2018)	100 h	-
`Speech2Vec` (Chung and Glass, 2018)	500 h	-
`Wav2Vec large` (Schneider et al., 2019)	960 h	-
`Vq-wav2vec` (Baevski, Schneider, and Auli, 2019)	960 h	-
`Wav2vec 2.0` (Baevski et al., 2020)	960 h	-

TABLE 2.1: Summary of the presented state-of-the-art methods with the corresponding amount of unlabeled and labeled training data size.

low resource languages improved the Word Error Rate (WER) by 5% in comparison to the monolingual training.

In Huang et al., 2013, a shared-hidden-layer multilingual DNN (`SHLMDNN`) with language - independent hidden layers and softmax layers dependent from the language is introduced. After the training of this architecture on a multilingual supervised corpus, the shared hidden layers (SHLs) were used in a cross-lingual transfer procedure to distinguish phones from a new language. These hidden layers can be seen as a feature extraction module which carried over phonemic information from multiple languages. It was demonstrated that the transfer of the learned hidden layers reduced the error recognition of a new language, whether the language is in the same family as the languages included in the training corpus or not.

In Veselỳ et al., 2012, a novel language-independent bottle-neck (BN) feature extraction framework based on Multilingual Artificial Neural Network is proposed. As in the previous studies, the features of all the source languages are captured by the hidden layers and the output layer models a unique language. The evaluation of the cross-lingual transfer on 3 languages with a training on a supervised multilingual dataset showed the effectiveness of the model to produce good BN-features even for languages that were not included in the training corpus.

Supervised pre-training became highly popular, but a limitation is that it requires huge amounts of annotated data (parallel corpora of speech and orthographic transcriptions) (Riviere et al., 2020), which are not available in the case of acutely under-documented languages.

2.2.2 Unsupervised pre-training

Unsupervised representation learning has the advantage of employing large amounts of unlabeled speech data (Zhang et al., 2021) to discover a suitable representation of speech frames.

Oord, Li, and Vinyals, 2018 introduces Contrastive Predictive Coding (CPC), an approach able to extract representation from high-dimensional data in an unsupervised learning frame. With the use of autoregressive models, representations were taught by predicting the future in the latent space. The final step to capture information is based on probabilistic contrastive loss which will be more effective in predicting the future samples. The model was trained using a 100-hour subset of the Librispeech benchmark dataset. The effectiveness of this approach to learn powerful representations that achieve strong performances was demonstrated.

Another framework, called `Speech2Vec` (Chung and Glass, 2018), based on a deep neural network architecture was implemented to learn a fixed-length vector representation of audio segments. It can be thought of as a speech implementation of the popular `Word2Vec` model (Mikolov et al., 2013a) where the vectors represent the semantic information of the speech utterances. The closer the word embeddings learned by `Speech2Vec` skipgrams are semantically, the closer the vectors will be in the embedding space. This RNN Encoder-Decoder framework was trained on a 500-hour subset of Librispeech and evaluated on 13 benchmarks datasets. This work demonstrated the robustness of the computed representations via similarity ratings by humans.

Within the Fairseq sequence-to-sequence toolkit released by Facebook AI, a framework called `wav2vec` (and all its variants) (Schneider et al., 2019; Baevski, Schneider, and Auli, 2019) aims to extract new types of speech representations for acoustic models from raw audio. In more details, `wav2vec` (Schneider et al., 2019) is a fully convolutional architecture that takes raw audios as input, and general representations of speech as output. The pre-training approach consists of two networks that are stacked on top of each other. The first one, the *encoder network* $f : X \mapsto Z$, transforms the raw speech samples $x_i \in X$ into a feature representation $z_i \in Z$ every 10 ms with the use of a five-layer convolutional network. The second, called the *context network* $g : Z \mapsto C$ takes the output of the encoder network to compute a single contextualized tensor c_i. The computed representations are used to solve a self-supervised prediction task. Precisely, `wav2vec` generates distractor examples within 10-second audio clips drawn from a proposal distribution. The objective is to distinguish them from a true sample that is k steps in the future by minimizing a contrastive loss. Finally, `wav2vec` is used as an input to an acoustic model, for example, a grapheme-based ASR model. By learning wav2vec representations on 1,000 hours of unlabeled speech from the LibriSpeech dataset, and then training a speech recognition model on these representations, the recognition performances improved as compared to the best-known supervised ASR models (Deep Speech 2 (Amodei et al., 2016), supervised transfer-learning (Ghahremani et al., 2017), etc.).

`Vq-wav2vec` (Baevski, Schneider, and Auli, 2019), a self-supervised learning of discrete speech representations ties in with `wav2vec`, and relies on vector quantized (VQ) representations of audio data. In this case, the vectors take their value in a predefined set instead of continuous values. In addition to the two networks from the `wav2vec` architecture, `vq-wav2vec` adds a quantization network $q : Z \mapsto \hat{Z}$, where Z is a dense representation of the raw audio X, and $\hat{Z}$ is the quantized representation that will be used by the context network C. The quantization network uses either K-means clustering or the Gumbel-Softmax approach (Jang, Gu, and Poole, 2016) as constraints in a Vector Quantized Variational Autoencoders. A deep bidirectional Transformer model called `BERT` (Devlin et al., 2018) was trained on the discretized unlabeled speech data and used as input to an acoustic model. Performances reached state-of-the-art results on Wall Street Journal (WSJ) dataset. The quantization module makes it suitable for algorithms that require discrete data.

The last approach is called `wav2vec 2.0` (Baevski et al., 2020), a framework for self-supervised learning of speech representations. It builds context representations from continuous speech representations and dependencies are obtained by the self-attention mechanism across the entire sequence of latent representations end-to-end. `Wav2vec 2.0` architecture is explained in Section 2.3.

However, the presented unsupervised pre-training state-of-the art studies focused on monolingual representation learning, i.e. the representations are computed for only one language. Moreover, the amount of training data is much larger than what we have (remember that the number of labeled data is between 5 and 30 hours).

A novel approach called `XLSR` (Conneau et al., 2019) bases its architecture on cross-lingual learning, an effective approach for low-resource languages (Lample and Conneau, 2019). Its goal is to build models that learn meaningful speech representations from multiple languages via pre-training and transfer them to the target language (Conneau et al., 2020). This form of transfer learning is useful when occurring between two entities that share some underlying structure. In the case of speech, many languages share common linguistics structures, from phonemes to tones. This approach builds a single multilingual speech recognition model, which is competitive with strong individual models. As for the pre-training approach, `XLSR` uses the learned representations from `wav2vec 2.0`, shared across languages. Specifically, it has been demonstrated that a model pre-trained on 53 languages with more than 56k hours of unlabeled speech data (`XLSR-53`) constructs better speech representations that transfer to low-resource languages (Conneau et al., 2019).

2.3 Model architecture

The first module of the `XLSR` approach aims at learning cross-lingual speech representations thanks to `wav2vec 2.0` introduced by Baevski et al., 2020 and extended to the cross-lingual setting. We can divide the architecture into three modules: the feature encoder, the context network, and the quantization module.

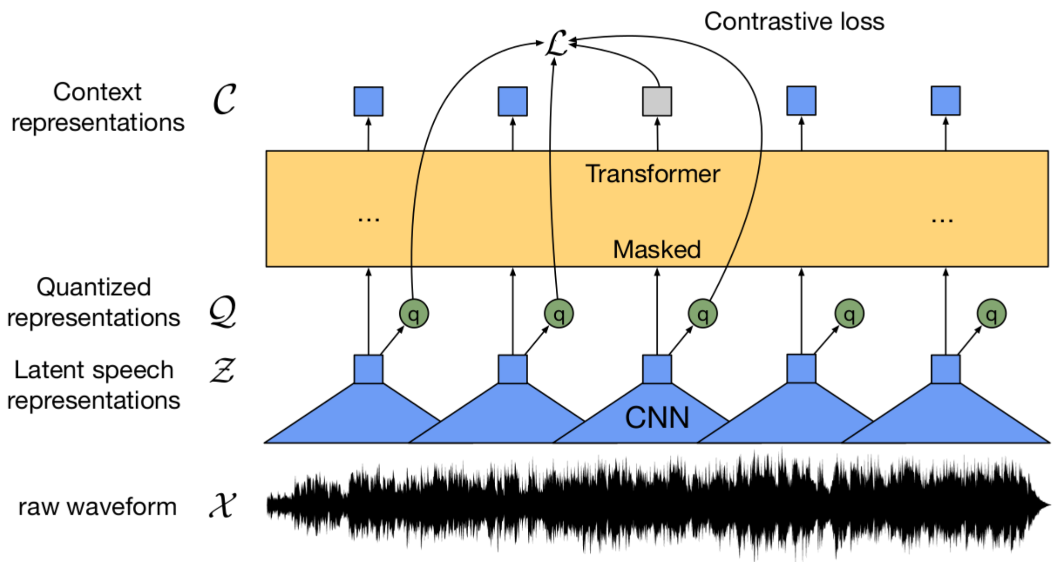


FIGURE 2.2: Illustration of the XLSR model which learns contextualized speech representations. A quantization module over feature encoder representations produces multilingual quantized speech units whose embeddings are then used as targets for a Transformer trained by contrastive learning. The approach uses as input the raw multilingual unlabeled speech data. Reproduced from Baevski et al., 2020.

2.3.1 Feature encoder

The feature encoder aims at encoding the input raw audio X into latent speech representations $z_1, ..., z_T$ for T time-steps. It consists of a multi-layer temporal convolutional network, a normalization layer (Ba, Kiros, and Hinton, 2016) and a GELU activation function (Hendrycks and Gimpel, 2016). Temporal Convolutional Networks (TCNs) can be seen as a variation of

Convolutional Neural Networks (CNNs) for sequence modeling tasks. Several advantages are to be noted: less memory is needed for training in comparison with recurrent architectures (a lot of memory can be used to store partial results from the multiple cell gates), inputs of variable length can be handled, and the performances are better than those of Long Short-Term Memory (LSTM)/Gated Recurrent Unit (GRU) architectures. Layer normalization has shown a tendency to stabilize the hidden state dynamics, but also to reduce the training time. The GELU activation function is used here to enable faster and better convergence of the network. A normalization step to zero mean and unit variance is performed over the raw input audio. The feature encoder takes the raw speech as input because better performances were demonstrated in comparison to spectral feature based system with CNN-based systems (Palaz, Collobert, et al., 2015). Finally, the T time-steps are defined by the total number of strides of the encoder, and that will be fed as input to the context network.

2.3.2 Contextualized representations with Transformers

The context network is implemented with a Transformer architecture (Vaswani et al., 2017; Devlin et al., 2018; Liu et al., 2019). The goal here is to learn contextualized representations $c_1, ..., c_T$ from the latent speech representations $z_1, ..., z_T$. Contextualized representations of each speech frame is the concatenation of the left-to-right and right-to-left representations. The major benefit is to take into account the context, which results in building more powerful representations.

The following explanation is based on blog posts from Jay Alammar[5] and Olivier[6]. The Transformer architecture contains two main components:

- a stack of encoders (six on top of each other),
- a decoding module with a stack of six decoders.

The general idea of the encoder-decoder is to build contextualized representations of speech frames. Their definition depends on all the other frames in the sequence. To do so, the network will employ an attention mechanism where it will learn to look at relevant parts of the sequence to build the representations by combining the linear representations of the other speech frames associated to the learning weights.

Specifically, the encoders contain two sub-layers. The first one, the self-attention layer will focus on other frames in the input sequence during the encoding of a specific frame. By looking at other positions in the input sequence, the self-attention layer will then be able to find clues to best encode. The output of this layer will be fed into a feed-forward neural network layer. The decoder follows the same structure but an attention layer is added between the two, which goal will be to help focusing on relevant parts of the input sequence.

In the `XLSR` approach, a convolutional layer encodes the latent speech representations into relative positional embeddings (a list of vectors). This step takes place in the first encoder of the stack. The other encoders will take the output of the preceding encoder as input. This list of vectors will then be processed by the two layers of the encoder mentioned earlier.

The particularity of the Transformer architecture is the use of a multi-head attention mechanism. Its purpose is to have multiple representation spaces that prevent the representation from being totally biased if one layer (head) of attention is.

The self-attention layer creates three vectors for each input vector (the embedding):

- a 'Query' vector q,
- a 'Key' vector k,

[5]https://jalammar.github.io/illustrated-transformer/
[6]https://ledatascientist.com/a-la-decouverte-du-transformer/

- and a 'Value' vector v.

Each vector (q, k, and v) is defined by the multiplication of the embedding with three matrices (Q, K and V) obtained during the Transformer training process. Each frame from the input sequence will be assigned with a score. The score is calculated by the dot product between the query vector and the key vector and is then divided by the square root of the dimension of the key vectors. This division ensures a stable gradient by minimizing the result of the dot product. A softmax normalization is performed to get a score between 0 and 1. It will give the probability of the i^{th} frame. Finally, each value vector v will be multiplied by this softmax score before being added up. At each time step, the self-attention layer will produce an output.

Each sub-layers which constitutes the encoders, as well as for the decoders has a residual connection followed by a layer-normalization.

The output of the top encoder is transformed into a set of attention vectors k and v that will be processed by the decoding module (i.e. by each decoder layer). The decoder will therefore be able to focus its attention on the relevant information from the input sequence. Finally, as for the encoding module, the output of each decoder is fed into the next decoder in the stack.

The result of the Transformer network $g : Z \mapsto C$ is the contextualized representations from the speech inputs.

2.3.3 Quantization module

The last module of the `XLSR` approach is the use of a quantization module $Z \mapsto Q$ which takes the latent speech representations computed by the feature encoder z and transforms them into a finite set of speech representations via product quantization (Jegou, Douze, and Schmid, 2010). As in Baevski, Schneider, and Auli, 2019, the original representation z is replaced with this quantization module by $\hat{z} = e_i$ from a fixed size codebook $e \in \mathbb{R}^{V \times d}$ where V is the number of representations of size d. In the `XLSR` model, $G = 2$ codebooks with $V = 320$ entries each were chosen. The discretized representation q are obtained via the concatenation of the resulting vectors $e_1, ..., e_G$.

2.3.4 Unsupervised representation learning (pre-training) with `wav2vec 2.0`

To pre-train the model, a contrastive task is resolved over masked latent feature encoder outputs (proportion of time-steps), similarly to the masked language modeling in `BERT` (Devlin et al., 2018). In other words, for each masked time-step, the objective is to correctly identify the quantized latent audio representation from a set of distractors.

Masking

To be more specific, after masking some time-steps of the feature encoder outputs, they will be fed to the context network. However, the quantization module does not use the masked inputs. To define the latent speech representations that will be masked, one defines the starting indices by randomly sampling some proportion p of all time steps. The consecutive time steps M are then masked from every sampled index.

Objective

The objective here is to solve a contrastive task $\mathcal{L}_m$ the principle of which is based on the identification of the true quantized latent speech representations from a set of distractors for a given masked time step. In other words, the learning of speech audio representations is given by the formula:

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{2.1}$$

where $\mathcal{L}_m$ is the contrastive loss, $\mathcal{L}_d$ corresponds to the diversity loss and α is a tuned hyperparameter.

Given the context network output c_t from a t time-step, the contrastive loss is defined as

$$\mathcal{L}_m = -\log \frac{\exp(sim(c_t, q_t)/k)}{\sum_{\tilde{q} \sim Q_t} \exp(sim(c_t, \tilde{q}_t)/k)} \tag{2.2}$$

where q_t is the true quantized latent speech representation, $\tilde{q} \in Q_t$ is the set of quantized candidate representations with K distractors and $sim(c_t, b)$ is the cosine similarity between context representations c_t and the quantized latent speech representations (He et al., 2020; Chen et al., 2020). To increase the use of quantized codebook representations, the diversity penalty $\mathcal{L}_d$ is introduced (Dieleman, Oord, and Simonyan, 2018).

The entropy of the averaged softmax distribution is maximized over the codebook entries for each codebook $\bar{p}_g$ across a batch of utterances with,

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{2.3}$$

where G is the number of codebooks, V the number of entries, and H the entropy.

The pre-training involves the definition of multilingual batches (Devlin et al., 2018; Lample and Conneau, 2019) because the model learns contextualized representations over L languages. The batches are computed by the sampling of the speech samples from a multinomial distribution $(p_l)_{l=1,...,L}$ where $p_l \sim (\frac{n_l}{N}^{\alpha})$ with n_l, a language's l number of pre-training hours, N, the total number of hours, and α, the upsampling factor, i.e. the weight given to high-resource languages versus low-resource languages.

2.3.5 Fine-tuning

Fine-tuning is the part on which the experiments will be based. This step comes after the pre-training of a model on multiple languages. The fine-tuning for speech recognition involves a vocabulary, with C classes (i.e. the number of tokens). The vocabulary is added on top of the context network thanks to a randomly initialized linear projection (Baevski, Auli, and Mohamed, 2019). A classifier on top of the model will represent the output vocabulary, trained on labeled data thanks to a Connectionist Temporal Classification (CTC) loss (Graves et al., 2006).

Connectionist Temporal Classification (CTC)

To give an insight of how the CTC algorithm works, we will base our explanation on Hannun, 2017. Connectionist Temporal Classification is an approach to be considered when the alignment between the audio and the transcription is unknown. Given input sequences $X = [x_1, x_2, ..., x_T]$ such as audio, and output sequences $Y = [y_1, y_2, ..., y_U]$ such as transcripts, the goal is to find the optimized mapping from X to Y. Some challenges are encountered but can be overcome by the CTC algorithm, for example, the variable lengths of the sequences or the variation of the ratio of lengths of X and Y. An output distribution over all possible Y is computed over a given X. This distribution can be used in two ways: the first one to infer a likely output, and the second one, to evaluate the probability of a given output.

The algorithm works in the following way: the probability of an output from an input is computed by adding up the probability of all possible alignments between the two. In a naive approach, an output character will be assigned to each input step. For example, we can consider an input of length 8 and $Y = [b, o, a, t]$. A possible alignment can be seen in Figure 2.3.

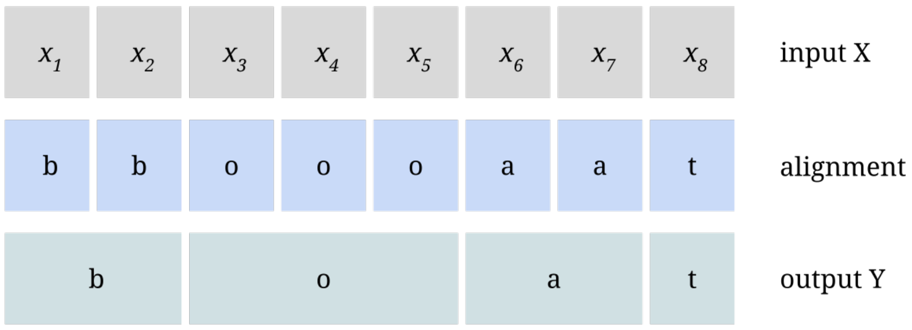


FIGURE 2.3: Illustration of the naive algorithm. An output character is assigned to each input character based on a computed probability.

However, in the case of speech recognition, the speech contains silences or sounds that do not correspond to a specific output. Moreover, it does not take into account an output with multiple characters. For example, the alignment [b, b, u, u, b, b, b, l, e, e] will be considered as 'buble' instead of 'bubble'. In the CTC approach, a new token, the *blank token* ϵ is introduced in the set of allowed outputs. Therefore, the alignments produced by the CTC approach are of the same length. At the end, blank tokens as well as repetitions will be removed.

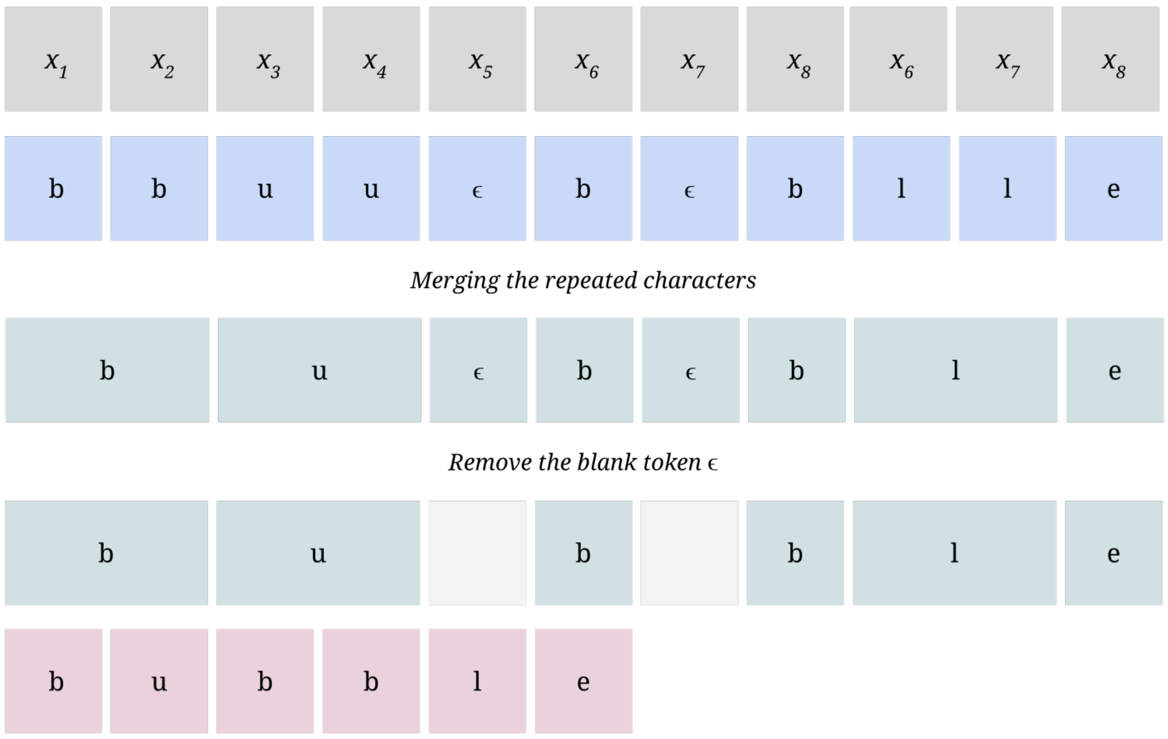

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_6 x_7 x_8
b b u u ϵ b ϵ b l l e
Merging the repeated characters
b u ϵ b ϵ b l e
Remove the blank token ϵ
b u b b l e
b u b b l e

FIGURE 2.4: Illustration of a valid alignment produced by the CTC for a given input sequence $X = [x_1, x_2, ..., x_T]$. The *blank token* ϵ is introduced.

In Figure 2.4, a valid alignment must have a blank token between two identical characters. Several advantages are notable. The alignments between X and Y are monotonic, meaning that we are assuming that source and target sequences are roughly monotonically aligned. Also, the alignment can be described as many-to-one, because one or several input elements can be aligned to the same output character whereas the opposite is not true. Therefore, the length of Y can be larger than the one of X.

The CTC objective is defined as:

$$p(X|Y) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X) \tag{2.4}$$

where $p_t(a_t|X)$ is the probabilities per time-step and the sum marginalizes the probabilities over the set of valid alignments. The probabilities are estimated with a deep neural network, that will take into account the context from the input. The CTC-loss is time and resource expensive, the use of a dynamic programming algorithm overcomes this challenge by merging the alignments that have reached the same output at the same time step. Let's take the example of a sequence $Z = [\epsilon, y_1, \epsilon, y_2, ..., y_U, \epsilon]$. We define α, the CTC score of the merged alignments (the subsequence $Z_{1:s}$) after t time step. The final CTC score will be obtained by the last time-step α, if and only if the previous α time-step score is known.

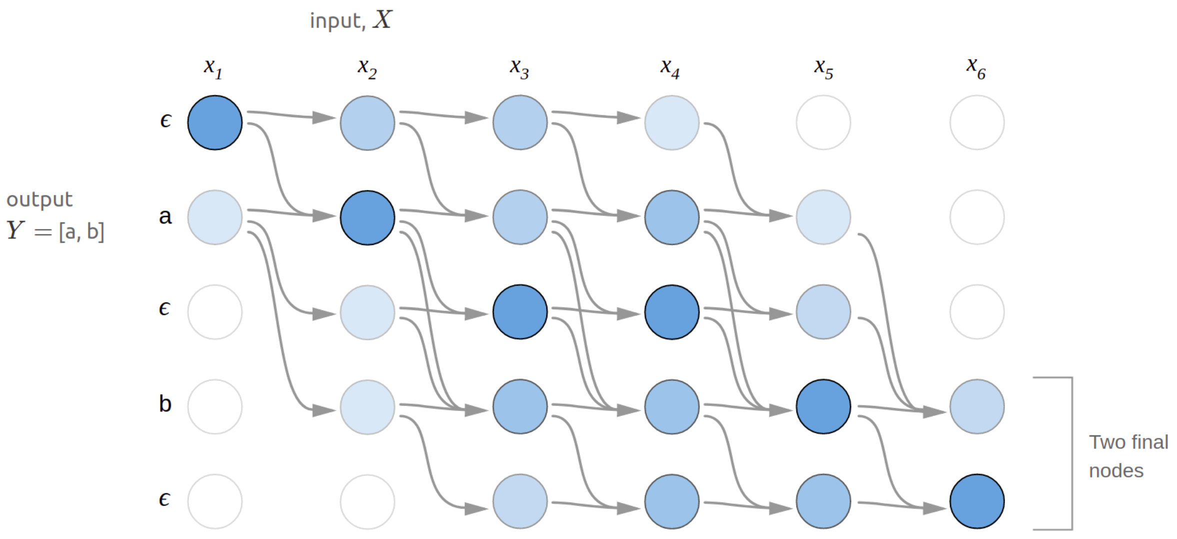


FIGURE 2.5: Illustration of the dynamic programming algorithm computation. Two starting and ending nodes are possible (ϵ is optional) (Hannun, 2017).

Figure 2.5 shows how the computation is performed by the dynamic programming algorithm, where two starting and ending nodes are possible (ϵ is optional). The resulting probability is the sum of the two final nodes. The loss function is computed efficiently.

The following steps involve the computation of the gradient and the training of the model. Specifically, the gradient of the loss function is computed thanks to the unnormalized output probabilities. Let's define a training set $\mathcal{D}$, the negative log-likelihood is minimized by the model's parameters tuning defined by

$$\sum_{(X,Y) \in \mathcal{D}} - \log p(Y|X) \tag{2.5}$$

Finally, the last step is to find the output at each time step that maximize the probability for a given input. Specifically, the alignment with the highest probability is defined by

$$A^* = argmax_A \prod_{t=1}^{T} p_t(a_t|X) \tag{2.6}$$

The resulting alignment comes up with the deletion of repetitions and blank tokens.

Chapter 3

Experiments

The experiments carried out during this work involve two labeled datasets that will be described in Section 3.1. Their goal was to determine the feasibility of fine-tuning the `XLSR` model on two "under-resourced language" corpora and evaluate the performances of the trained models. To do so, a specific pipeline was constructed that relies on the HuggingFace library. This library provides pre-trained models (and in particular the `XLSR-53` model described in the previous chapter[1]) as well as an high-level API to fine tune these models. Our pipeline is based on the tutorial provided by HuggingFace[2] to fine tune the `XLSR` model.

We first introduce the datasets and the languages used in our experiments (see Section 3.1). We continue with a detailed description of the pipeline (see Section 3.2), the evaluation metrics (see Section 3.3), and the results (see Section 3.4). The two last sections present experiments linked to the beam search algorithm (see Section 3.5) as well as complementary experiments (see Section 3.6).

3.1 Datasets

As the objective is to build a system able to automatically transcribe speech into words in a low-resource context, we focus on two corpora to fine-tune the `XLSR` model, the Yongning Na and the Japhug corpora. The data are available in the Pangloss collection (CNRS, 2021a). This collection was initiated in 1995 by the Lacito to provide an online storage for endangered language recordings, with a view to safeguarding and making available the linguistic heritage. This collection is anchored in a desire to make science as widely available as possible (open science). It allows the conservation and referencing of researchers' work through time as well as unlimited access to this work for all. To give a few figures, more than 3,600 audio recordings are available from 170 languages across the world. The audio recordings range from storytelling and songs to conversations and recipes. More than half of the recordings have transcriptions, which is the case for the two languages considered in our work, the Yongning Na (see Section 3.1.1) and the Japhug (see Section 3.1.2).

3.1.1 Yongning Na

Yongning Na (CNRS, 2021c) (also known as *Narua* and *Mosuo*) is a language spoken on the border of China's Yunnan and Sichuan provinces around Lake Lugu (泸沽湖) (see Figure 3.1).

Yongning Na belongs to the *Naish* group of the Sino-Tibetan family. Around 47,000 speakers of Yongning Na were estimated in the Ethnologue database, based on the Summer Institute of Linguistics' own sources (Lewis and (eds.), 2016), but this number is continuously decreasing as more and more people communicate using China's main language (standard Mandarin). The resources were collected by Alexis Michaud. The corpus contains recordings

[1]`https://huggingface.co/transformers/model_doc/xlsr_wav2vec2.html`
[2]This tutorial is available at `https://huggingface.co/blog/fine-tune-xlsr-wav2vec2`

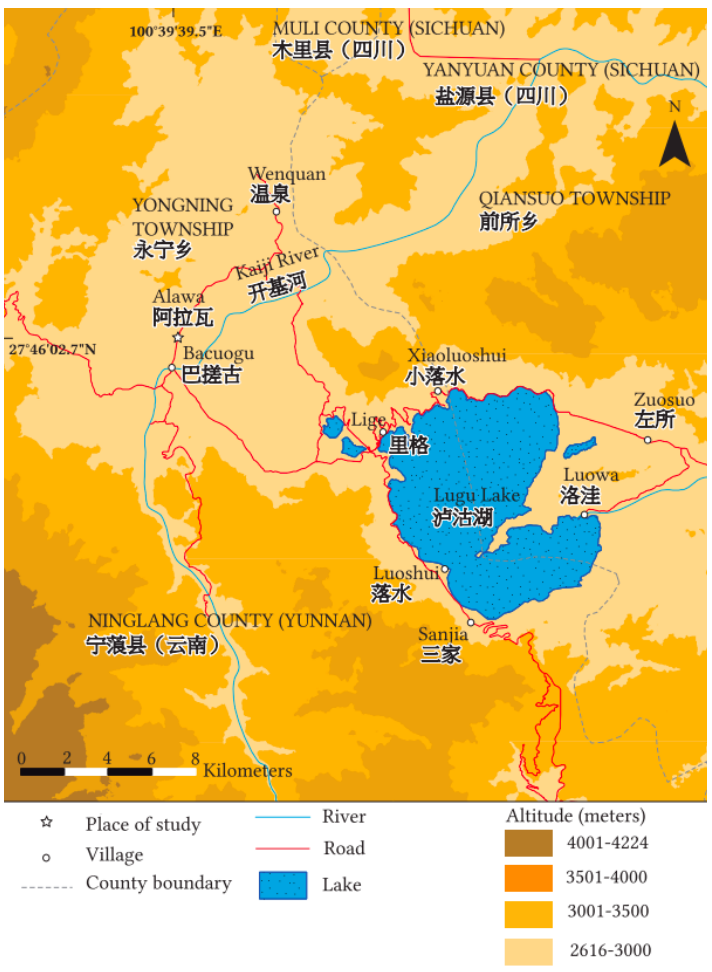


FIGURE 3.1: Map of the Yongning area. Designed by Jérôme Picard. Sources: Geofabrik, ASTER GDEM (a product of METI and NASA) and OpenStreetMap (Michaud, 2017).

of a single speaker: Dashilame LATAMI. The vast majority of the resources were recorded in the Yongning plain (永宁). A detailed description of the language can be found in Michaud, 2017.

3.1.2 Japhug

Japhug (CNRS, 2021b) is spoken by a minority of about 10,000 people living in the Tibetan part of Sichuan (see Figure 3.2) (Jacques, 2015). These resources were collected by Guillaume Jacques, a CNRS researcher specialized in the descriptive and historical linguistic of Sino-Tibetan languages. The work on this language began in 2002, and many publications are available on this subject, notably on the “Phonology and morphology of Japhug (rGyalrong)” (Jacques, 2004) and a Japhug-Chinese-French dictionary (Jacques, 2016). Most of the recordings are from a single speaker, Tshendzin, but other speakers were also recorded. A comprehensive description of the language can be found in the recently published book entitled “*A grammar of Japhug*” (Jacques, 2021).

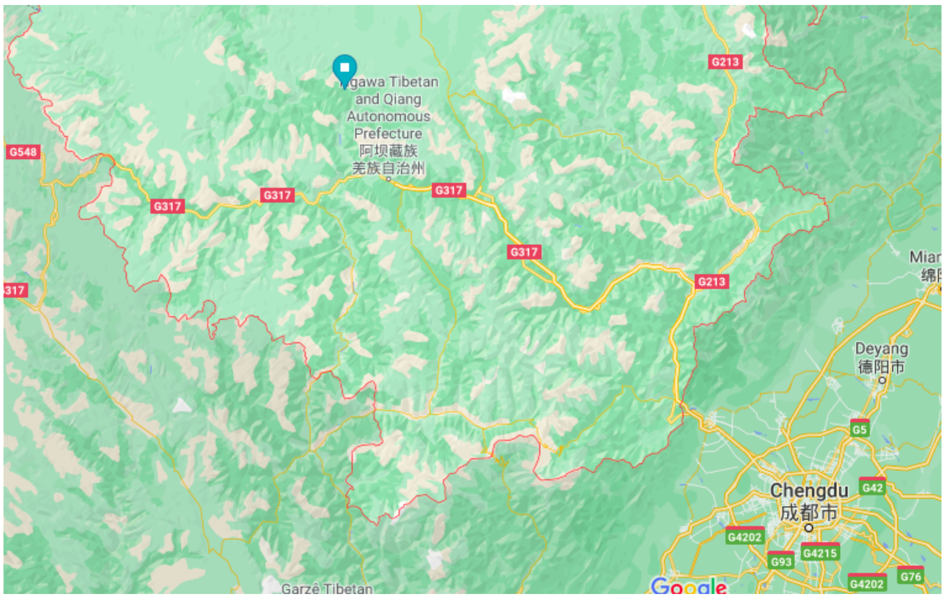
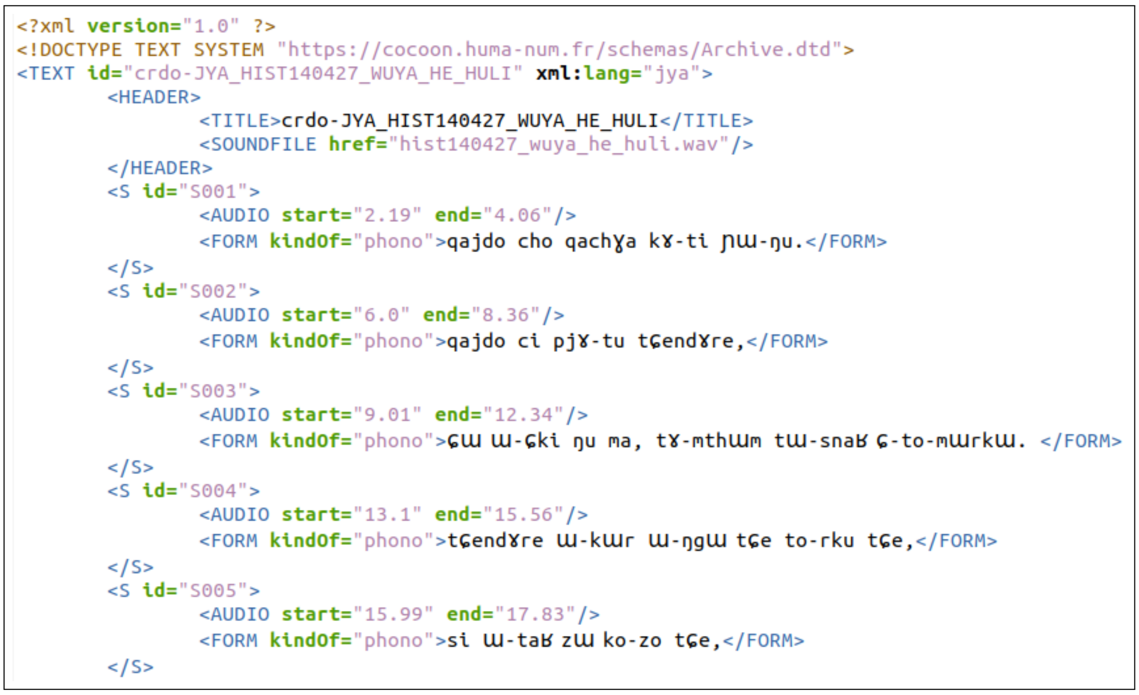


FIGURE 3.2: Location of the Japhug spoken community living in the Tibetan part of Sichuan (Google, 2021a).

Both corpora contain pairs of audios and their transcriptions: the audio are in WAV format with a corresponding IPA-based transcription in the XML format. Each transcription includes sentence-level timecodes. Timecodes are useful in the case of automatic speech recognition because it is easy to recover small audio segments, as the model can only encode very short ones (up to 30 seconds).



```
<?xml version="1.0" ?>
<!DOCTYPE TEXT SYSTEM "https://cocoon.huma-num.fr/schemas/Archive.dtd">
<TEXT id="crdo-JYA_HIST140427_WUYA_HE_HULI" xml:lang="jya">
        <HEADER>
                <TITLE>crdo-JYA_HIST140427_WUYA_HE_HULI</TITLE>
                <SOUNDFILE href="hist140427_wuya_he_huli.wav"/>
        </HEADER>
        <S id="S001">
                <AUDIO start="2.19" end="4.06"/>
                <FORM kindOf="phono">qajdo cho qachɣa kɤ-ti ɲɯ-ŋu.</FORM>
        </S>
        <S id="S002">
                <AUDIO start="6.0" end="8.36"/>
                <FORM kindOf="phono">qajdo ci pjɤ-tu tɕendɤre,</FORM>
        </S>
        <S id="S003">
                <AUDIO start="9.01" end="12.34"/>
                <FORM kindOf="phono">ɕɯ ɯ-ɕki ŋu ma, tɤ-mthɯm tɯ-snaʁ ɕ-to-mɯrkɯ. </FORM>
        </S>
        <S id="S004">
                <AUDIO start="13.1" end="15.56"/>
                <FORM kindOf="phono">tɕendɤre ɯ-kɯr ɯ-ŋgɯ tɕe to-rku tɕe,</FORM>
        </S>
        <S id="S005">
                <AUDIO start="15.99" end="17.83"/>
                <FORM kindOf="phono">si ɯ-taʁ zɯ ko-zo tɕe,</FORM>
        </S>
```

FIGURE 3.3: Example of an XML transcription. The `<HEADER>` describes the title and the soundfile name. The `<S>` tag refers to a specific sentence with the corresponding timecodes.

Figure 3.3 shows an example of a transcription associated to an audio recording. The tag `<S>` indicates the beginning of the sentence and `</S>` indicates the end. In between, we find the timecodes defined with an `<AUDIO>` tag, and the transcription in International Phonetic Alphabet (IPA) written in a `<FORM>` tag.

Table 3.1 presents the statistics associated to each corpus.

Corpus	**Yongning Na**	**Japhug**
Number of files	57 <audio, xml>	357 <audio, xml>
Number of sentences	2,484	31,864
Total duration (in minutes)	209.52 (≈ 3h30)	1907.57 (≈ 31h47)
Number of speakers	1 female speaker	2 male and 2 female speakers

TABLE 3.1: Corpus information and statistics.

The key differences between the two are the size of the corpora. The Na corpus has less than 4 hours of recordings corresponding to a "typical" situation for a low-resource language. In opposition, Japhug corpus has over 30 hours of labeled audio recordings, but is still much lower than the benchmark datasets in ASR. Furthermore, there are multiple speakers in the Japhug corpus, but only a single one in the Na dataset. It is worth noting that most of the corpora in the Pangloss collection have very few or no annotated audio files, and very limited resources (a few minutes recorded). The use of these two corpora will allow us to have an upper bound on the performance of the `XLSR` approach. Moreover, the field linguists, experts of these two languages, were ready to collaborate and to give us feedback on the outputs generated by the model.

3.2 Pipeline

Figure 3.4 gives an overview of our pipeline. The first steps consists in preprocessing the data (see Section 3.2.1). The vocabulary is then defined from the transcriptions (see Section 3.2.2). The text is tokenized and the features are extracted from the speech files (see Section 3.2.3). The pretrained model is loaded and fine-tuned on the previously processed data (see Section 3.2.4). The final phase is the generation of the predictions, namely the decoding step.

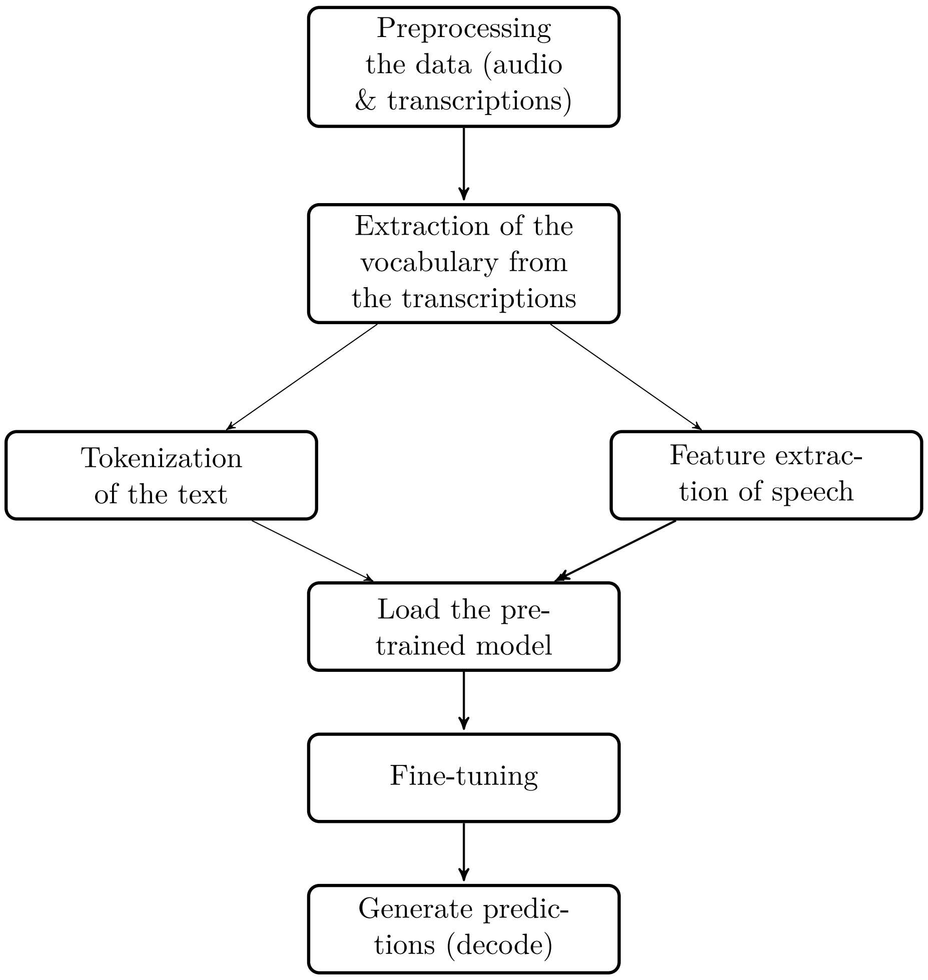

Preprocessing the data (audio & transcriptions)
Extraction of the vocabulary from the transcriptions
Tokenization of the text
Feature extraction of speech
Load the pre-trained model
Fine-tuning
Generate predictions (decode)

FIGURE 3.4: Pipeline developed in the context of this work to fine-tune a pre-trained `XLSR` model on the two studied low-resource corpora, Yongning Na and Japhug.

3.2.1 Preprocessing

A *preprocessing step* is required to fine-tune the `XLSR` model pretrained on 53 languages. The model requires the use of high quality speech files (no background noises, silences, etc.). Transcriptions should be aligned at the sentence level with the corresponding audio, and cleaned (by removing the punctuation or converting the transcriptions to lowercase, for instance).

More precisely, the preprocessing consists of the following steps:

- Each audio file is cut according to the corresponding sentence segments in the transcription, which creates a .tsv file (see Appendix A.1 for details),
- The data are split into train, validation, and test sets, which represent respectively 70, 15, and 15% of the total amount of data,
- The transcriptions are then 'cleaned' before fine-tuning the pretrained model. Cleaning consists first in removing punctuation marks, spaces and line breaks. Without a language model, it is much harder to classify speech chunks to special characters because they do not correspond to a specific sound unit. Then, some language-specific preprocessing rules are applied. These rules require knowledge of the conventions used to annotate the corpus. For instance, the Na corpus contains specific characters such

as '...' or '↑' that have to be removed, as well as the transcriptions between square brackets `[...]`. These rules were inspired by the ones developed by Oliver Adams in **`persephone`**[3] (Adams et al., 2018). For the Japhug corpus, specific rules were developed to remove comments by the linguist (such as `(superlative)`, `(causative)`, etc.). The corpus contains small portions of speech in Chinese, transcribed in Chinese characters. We made the choice to keep these information because removing them would result in a mismatch between the audio and the transcription. Finally, the space between words is replaced by '|'.

As an example, the sentence from the audio file entitled "Dog: How dog and man exchanged their lifespan (version 2)"[4] from the Na corpus:

Ref: ʈʂʰɯ˧ne˧-ʝi˥ | tʰi˧-tɕɯ˧-ɲi˥-tsɯ˩ ◊ -mv̩˩. |

becomes:

Ref_processed: ʈʂʰɯ˧ne˧ʝi˥|tʰi˧tɕɯ˧ɲi˥tsɯ˩|mv̩˩

`Ref` stands for the *reference sentence*, i.e. the initial sentence from transcription. Here, the '|' refers to a tone group boundary, not a word boundary. `Ref_processed` is the sentence resulting from the preprocessing, and the sentence that we want to retrieve using the model. As we see, the hyphen is deleted, as well as the '◊' and the tone group boundary, which do not refer to a specific phoneme.

3.2.2 Definition of the vocabulary

For each corpus that will be used during the fine-tuning of the pretrained model, a vocabulary is generated and is fed to the tokenizer[5]. The vocabulary is the list of symbols (or tokens) that will be recognized by the model. The model will be able to predict the tokens during the decoding process powered by the CTC module (see Section 2.3.5). We do not consider phoneme units but character units. Here, a phoneme is encoded in several units if the phoneme contains multiple characters (e.g. ʈʂʰ encoded into 3 units ʈ, ʂ, and ʰ). As the model needs to learn how to predict word boundaries (or else the model would print a sequence of characters, which is not our goal here), the space is encoded into a special character. We chose the same as in the tutorial: a pipe symbol '|'.

The vocabulary size is 57 for Na and 92 for Japhug. This vocabulary includes two special tokens: `[UNK]`, the unknown token (to deal with characters not encountered in the training set), and `[PAD]`, the token used for padding (a main component of the CTC algorithm).

3.2.3 Tokenizer and feature extractor

The tokenizer's goal is to convert the text into the corresponding token IDs. It will also process the model's output format to text. The tokenizer takes as arguments the vocabulary file, the `[UNK]` token, the `[PAD]` token, and the word delimiter token | . The feature extractor transforms the speech signal into the model's input format. The following arguments are taken into account:

- the feature size — the input of speech models are a sequence of feature vectors. While the duration of this sequence will certainly vary, the size of the features should not. With `wav2vec 2.0`, the feature size is 1 because the pre-trained model used raw speech signal.

[3] `https://github.com/persephone-tools/persephone/blob/master/persephone/datasets/na.py`
[4] `https://doi.org/10.24397/pangloss-0004660#S21`
[5] `https://huggingface.co/transformers/model_doc/wav2vec2.html#wav2vec2ctctokenizer`

- the sampling rate — this value has to be set according to the sampling rate on which the model is pre-trained.
- the padding token ID value,
- do_normalize — decides whether the input should be normalized (zero-mean-unit-variance) or not. In the case of speech models, the performances are better when the input is normalized.
- return_attention_mask — influences whether the model should use an attention mask for batched inference (which is the case with the XLSR model) or not.

If we take the example of a Japhug sentence,

stu kɯwxti chondɤre nɯ ɯpa nɯ tɯlɤt ni wuma ʑo pjɤɕqraʁndʑi

the tokenizer will convert it into a sequence of token ids $[25, 11, 15, 47, 20, 34, 23, 5, 11, 26, ...]$ and the feature extractor will extract the features from speech as a sequence of vectors of floats.

3.2.4 Fine-tuning

The `XLSR` model is pretrained on 53 languages before being fine-tuned on one of the two corpora considered in the present work (Japhug and Na). The pretrained model was trained with resources from 3 corpora:

- the Multilingual LibriSpeech dataset (Pratap et al., 2020) which includes 8 languages and 50k hours of audio files,
- the CommonVoice corpus[6], a multilingual corpus of read speech containing more than two thousand hours of speech data in 38 languages (Ardila et al., 2019),
- and the Babel corpus[7], a multilingual corpus of conversational telephone speech from IARPA, with Asian and African languages. In this case, resources from 10 languages were considered (Gales et al., 2014).

Table 3.2 reports the hyperparameters and the training arguments used to fine-tune the model. The values of these parameters are those recommended in the HuggingFace documentation.

parameter	value
`pretrained model`	`wav2vec2-large-xlsr-53`
`attention_dropout`	0.1
`hidden_dropout`	0.1
`feat_proj_dropout`	0.1
`mask_time_prob`	0.075
`layerdrop`	0.1
`ctc_loss_reduction`	mean
`train_batch_size`	8
`num_train_epochs`	60
`fp16`	True
`learning_rate`	3e-4

TABLE 3.2: Value of the hyperparameters used to fine-tune the XLSR model.

[6] https://voice.mozilla.org/en/languages
[7] https://catalog.ldc.upenn.edu/byyear

3.3 Evaluation metrics

Two evaluation metrics were used to track the performances along the training time: the Word Error Rate (WER)[8] and the Character Error Rate (CER)[9] (Morris, Maier, and Green, 2004). These are two standard metrics used to evaluate automatic speech recognition systems. The distance between the hypothesis of the model and the reference sequence is computed as the lowest number of modifications required to correct one into the other. These measures are derived from the Levenshtein distance. Dynamic string alignment is performed to align the sequence predicted by the ASR system (hypothesis) and the reference sequence. The lower the value, the better the performance of the ASR system. Note that the spaces predicted by the model are used to identify the words. This is why space is a specific character in the vocabulary.

The WER works on the word-level and is defined as follows:

$$\begin{aligned} WER &= \frac{S+D+I}{N} \\ &= \frac{S+D+I}{S+D+C} \end{aligned} \tag{3.1}$$

where S is the number of substitutions, D the number of deletions, I the number of insertions, C the number of correct words and $N = S + D + C$.

The following example (A) gives a reference sentence, and a corresponding hypothesis that could be predicted by an ASR model.

Ref: kɯki nɤki qaliaʁ lɯlu phaʁrgot nɯra ɲɯŋu
Hyp: kɯki nɤki qaliaʁ lɯlu

Here, we can see that the ASR model predicted 3 deletions, empathized in red:

$$\begin{aligned} WER &= \frac{0+3+0}{7} \\ &= \frac{3}{7} \approx 42.8\% \end{aligned} \tag{3.2}$$

The CER is computed on the character level. It follows the same formula (3.1) but C describes the number of correct characters, and N corresponds to the number of characters in the reference sentence. In this case, for example (A), the ASR model, on the character level, predicted 19 deletions:

$$\begin{aligned} CER &= \frac{0+19+0}{40} \\ &= \frac{19}{40} = 47.5\% \end{aligned} \tag{3.3}$$

Considering another example (B):

Ref: iɕqha nɯ kupa ɣɯ nɤki tɕhaŋkha nɯ tɕe fse wo
Hyp: iɕqha nɯ kupa ɣɯ nɤki tɕhaŋkha nɯ tse se wo

, the ASR model applied, on the word level, and on the character level, 1 substitution and 1 deletion, given in red, where:

[8] https://huggingface.co/metrics/wer
[9] https://huggingface.co/metrics/cer

$$WER = \frac{1+1+0}{10} = \frac{1}{5} = 20\% \qquad (3.4) \qquad\qquad CER = \frac{1+1+0}{44} = \frac{1}{22} \approx 4.54\% \qquad (3.5)$$

A final metric was defined, but only used to assess the performance of the Na model on the test set, the Phoneme Error Rate (PER). It is based on the same principle as the two previous metrics, but the PER is computed on the phoneme-level where C is the number of correct phonemes, and N is the number of phonemes in the reference sentence. The PER was not computed for the Japhug because this language consists of 8 vowels and 50 consonantal phonemes, all occurring as simple onsets (Jacques, 2021). In this case, the PER of the Japhug corresponds to the CER value. The following example gives a reference sentence cut into phoneme units, with the corresponding hypothesis that can be produced by an ASR model. Here, `<S>` defines a space.

Ref: m ɤ ˧ b i ˧ z e ˧ <S> ə ˧ z ɯ ˩ <S> h ĩ ˧ <S> tɕ^{h} i ˧ <S> t^{h} ɑ ˧ m ɤ ˧ ʝ i ˧ z e
Hyp: m ɤ ˧ b i ˧ z e ˧ <S> ə ˧ z ɯ ˩ <S> h ĩ ˧ <S> tɕ^{h} i ˧ <S> t^{h} ɑ ˩ m ɤ ˩ ʝ i ˩ z e

In this case, the ASR model made 3 substitutions where:

$$PER = \frac{3+0+0}{32} = \frac{3}{32} \approx 9.37\% \qquad (3.6)$$

3.4 Results

This section presents the results of the experiments described above. In Section 3.4.1, we report the performances on the test sets on both corpora. The results are supplemented with several examples to determine the sources of possible errors from the model predictions. The phoneme error rate associated with the Yongning Na experiments is explored in Section 3.4.2.

3.4.1 Performances on the test sets

The main objective is to build ASR systems that can recognize word-level entities in a low-resource context. To do so, the pre-trained xlsr-53 model was fine-tuned on different dataset sizes from the Yongning Na and Japhug corpora. The performances were evaluated on the test data, respectively 32 minutes for the Na, and 21 for the Japhug. The WER, the CER and the PER are reported in Table 3.3.

Model	**Training size (in minutes)**	**WER (%)**	**CER (%)**	**PER (%)**
`xlsr-na-180`	180	41.51	7.97	6.59
`xlsr-jya-600`	600	18.56	7.44	-

TABLE 3.3: WER, CER, and PER on the Na test set when training on Na low-resource labeled data setups of 180 minutes. WER and CER on the Japhug test set when training on Japhug low-resource labeled data setups of 600 minutes.

Previous work (Adams et al., 2020) reports CER scores on the same corpora but with different training and test sets with `ESPnet`, an end-to-end neural network-based speech recognition toolkit (Watanabe et al., 2018). For Na, our model (`xlsr-na-180`) outperforms `ESPnet` by more than 7 percentage points (pp) with a CER of 7.97% while `ESPnet` achieves a CER of 14.5%. The performance on the Japhug increased by 5.36pp, from 12.8% with `ESPnet` to 7.44% with the best proposed model `xlsr-jya-600`.

The CER scores of both models are very low, which makes their use in “real life” possible. We see in the examples below the prediction from the `xlsr-na-180` model of a reference sentence. Complementary predictions can be found in the Appendix (see Figure A.2).

The first one, below, will help identify the type of errors encountered on phonemes and the correctness of word boundaries definition.

Ref: ɖʐe˧ dʑɤ˩˥ ɖɯ˧mɤ˧kv̩˧tsɯ˥ mv̩

Hyp: ɖʐe˧ tɕɤ˩˥ ɖɯ˧mɤ˧kv̩˩tsɯ˥ mv̩

We use a red font to highlight the differences between the two strings. We notice the error on two phonemes, but especially the wrong prediction of the Mid tone ˧ into a Low tone ˩. However, boundaries between words are well-defined. It is not the case in the second example in which two words are combined:

le˧dʑi˧se˥dʑo˩

The last example shows a word that is split into two parts in the prediction:

ʝi˧lv̩˧ kv̩˧˥z

To get a better idea of how many times spaces are incorrectly predicted, and thus, of how many times words are incorrectly defined on the test set, we calculated the number of insertions and deletions associated to the space. These are reported in Table 3.4.

Model	**Correct spaces**	**Number of insertions**	**Number of deletions**
`xlsr-na-180`	2686	186	182

TABLE 3.4: Number of correct spaces, insertions and deletions associated to word boundaries observed on the test set of the xlsr-na-180 model.

374 sentences make up the test data. To give an order of magnitude, in almost all test sentences, two words are combined, or one word is split into two parts. There is a high error rate specifically for the definition of space.

Here are some hypotheses predicted by the `xlsr-jya-600` model (complementary results are reported in Appendix A.3):

Ref: tɤmu kɤtsa ci pjɤtundʑi tɕe

Hyp: tɤmɯ kɤtsa ci pjɤtu tʑɕe tɕe

Ref: tɕendɤre nɤki tshɤ tshɤnmu nɯ tɤrga kɯ pjɤsɤre

Hyp: tɕendɤre nɤki tshɯ tshɤtnmu nɯ tɤrga kɯ pjɤsere

The same error is observed for the word prediction where one word is split into two parts for the first Japhug example. We also specified the number of errors associated to the definition of space in Table 3.5.

Model	Correct spaces	Number of insertions	Number of deletions
`xlsr-jya-600`	2460	160	100

TABLE 3.5: Number correct spaces, insertions and deletions associated to word boundaries observed on the test set of the xlsr-jya-600 model.

For the Japhug, 350 sentences are in the test set. To approximate, about one third of sentences contain a space insertion or deletion. In general, the error rate at the space level is quite low for Japhug.

Another statement is the misprediction of vowels from the second Japhug example: ɤ predicted into ɯ and ɤ predicted into e.

When analyzing the predictions of the fine-tuned models in more detail, two major observations were made:

- the main incorrect predictions for the Na come from the tones (uni tones and bi tones),
- wholly mistaken assumptions of Japhug reference sentences, meaning that the audio does not match the reference sentence.

To support these comments, we conducted two other evaluations (cf. Table 3.6) by removing the tones for the first one (thanks to a python script), and removing the sentences with non-matching audio and transcriptions. This last step was done manually, by listening to the audio corresponding to a test sentence for which the reference and hypothesis were in red, for example:

Ref: cai ɯjwaʁ ɯtaʁ ri ɲɯβze ɲɯŋu
Hyp: bɣɤʑu qhe ʑɯrɯʑɤri

Model	Experiment	WER (%)	CER (%)
`xlsr-na-180`	Removing tones	33.78	7.27
`xlsr-jya-600`	Removing unmatched <audio,transcription>	17.21	6.22

TABLE 3.6: WER and CER on the test set by removing tones from the predictions of xlsr-na-180 model, and by removing unmatched audio-transcription pairs from the predictions of xlsr-jya-600 model.

By comparing with the results from Table 3.3, the CER of the Na decreases from 7.97% to 7.27%. The difference is of over 1 point for the CER of the Japhug: from 7.44% down to 6.22%. We made the choice not to delete sentences with one or two missing words in the transcriptions even though it generates a higher error rate, because this kind of error was not just occasional and correcting it would have been very cumbersome. For instance:

Ref: ɯku ɯtaʁ kutɯtɯɣ ʑo tɕe tɕendɤre
Hyp: ra ɲɤsɯso ri ɯku ɯtaʁ kutɯtɯɣ ʑo tɕe tɕendɤre

3.4.2 Phoneme Error Rate

The last but not least evaluation was conducted on the phonemes (on the Na data only). To be more specific, the Na corpus contains uni-phones such as ə or ʑ, bi-phones such as dʑ or jæ, and tri-phones such as tɕʰ or wæ̃. To compute the PER, we decided to plot a confusion matrix where each row corresponds to a phoneme from the reference matched with its predicted phoneme. This method is easy to implement and useful to see how each phoneme was predicted. Because of the matrix size [23,606 × 23,606], the algorithm only retrieves the phonemes that were incorrectly predicted or the confusion matrix of a specific filled in phoneme.

Table 3.7 below shows examples of reference phonemes and their predicted phonemes. The star symbol ∗ is a mark of deletions and substitutions.

Ref. phoneme	Pred. phoneme	Ref. phoneme	Pred. phoneme
*	<space>	k*	kʰ
**	ɭ̇	ŋ	g
*dʑ	ʈʂʰ	˩	˧
*z	dz	ʈ*ʰ	ʈʂʰ
*ɑ	wæ	p	b
*ʂ	ʈʂ	p*	pʰ
mmm...	*mm...	i	ɯ
əəə...	əə*...	ɑ	ɤ

TABLE 3.7: Examples of reference phonemes and a corresponding example of a prediction by the model.

Some interesting errors are the definition of spaces, the hesitations, and some vowels and tones. In Figure 3.5, we can see the confusion matrix computed for the specific phoneme ˩˥. We clearly see how the phoneme was predicted. In most cases (i.e. 328 times), the phoneme was correctly predicted. However, in a few examples, the hypothesis made by the model is wrong. In 8 cases, ˩˥ became ˧˥, and in 7 cases, it became a single tone ˩.

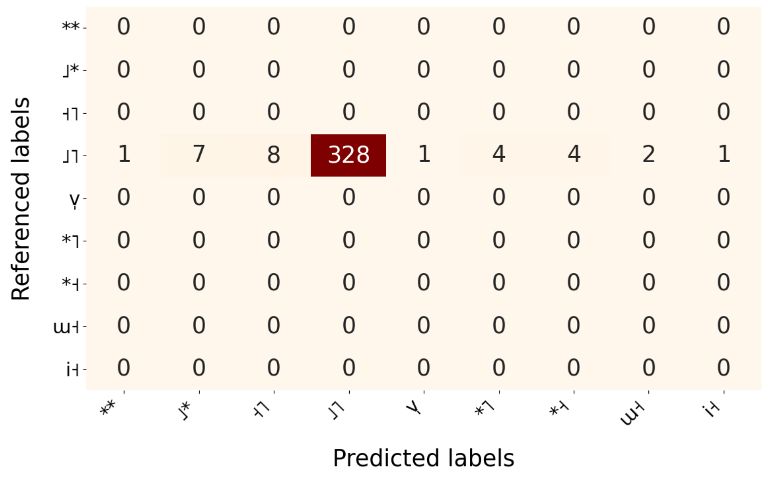


FIGURE 3.5: Confusion matrix of the reference phoneme ˩˥ and its predictions.

We have seen that the predictions made by the model contain several issues. Several methods to improve the error rate specifically at the word level will be presented in Section 3.5.

3.5 Beam search decoding

As seen in Section 2.3.5, the CTC decoder makes a conditional independence assumption over the characters in the output sequence (Viterbi lattice) at each timestep t. *Greedy search*, the default method, is an algorithm that will select the token with the highest probability of the n-th window from a CTC matrix, knowing the previously predicted tokens. It is defined as:

$$w_t = argmax_w P(w|w_{1:t-1}) \tag{3.7}$$

where w_t is the token probability at timestep t.

However, Greedy search's main limitation is the lack of consideration for other alignments that could have a much higher probability. Let's suppose we have the alignments [a, a, ϵ] and [a, a, a]. They individually have lower probability than [b, b, b], but the sum of their probabilities is higher. Greedy search will predict $Y = [b]$ as the best hypothesis, at the expense of $[a]$. Thankfully, the beam search algorithm alleviates this problem where the algorithm states that [a, a, a] and [a, a, ϵ] result in the same output.

3.5.1 Top k hypotheses

Beam search algorithm keeps a fixed number of beam hypotheses at each time step. Hidden high probability word sequences are less likely to be missed. For example, if we take a beam size of 4, at each time step, the algorithm will keep track of the 4 most likely hypotheses. Another feature that is available in the beam search method is the generation of the top beams. Simply, instead of printing the best hypothesis, we can set the k parameter corresponding to the number of the highest scoring beams that should be returned.

The WER and the CER oracle scores are computed on this top k prediction list, where Table 3.8 describes the oracle scores by using the `xlsr-na-180` model, and by the `xlsr-jya-600` model. Let us suppose that, for N input sentences, the beam search algorithm generates k hypotheses. In total, it creates $N \times k$ predictions. For each input sentence, the oracle method takes the prediction that maximizes the CER and WER scores over the list of k hypothesis. Finally, the CER and WER are computed on this new set of hypotheses list of size N. The oracle score gives an upper bound on the gains that can be achieved during the CTC decoding with beam search.

Model	K	Oracle CER (%)	Oracle WER (%)
`xlsr-na-180`	50	7.05	36.51
	100	6.88	35.54
	150	6.78	35.05
	200	6.71	34.5
	250	6.67	34.2
`xlsr-jya-600`	50	6.81	16.41
	100	6.65	15.79
	150	6.56	15.61
	200	6.50	15.53
	250	6.46	15.31

TABLE 3.8: Oracle WER and CER scores on top k hypotheses with the xlsr-na-180 and the xlsr-jya-600 models on the test sets.

The hypotheses generated by the top k beam search decoding algorithm show an overall improvement of 1 pp for the CER for both models in comparison with the results from Table 3.3 (6.67% against 7.97% of CER for the Na, and 6.46% against 7.44% of CER for the Japhug).

The WER score gains more than 6 pp for the Na (34.2% against 41.51% on Table 3.3), and over 3 pp for the Japhug (15.31% against 18.56% in Table 3.3). Moreover, the greater the number of hypotheses generated, the better the performance.

The tool VisTools created by OpenNMT provides a visualization of the top k beam search output. We adapted the code to fit our implementation. The core of the implementation is a Python script, which takes a json file as input with three pieces of information for each hypothesis: the predicted IDs of the tokens, the scores associated to each token, and the tokens. The output file is in an HTML format, which must be opened in a browser. The first node is the root, and the other nodes each represent a symbol with their corresponding scores. The path splits when the predicted symbol of a hypothesis is different from the best hypothesis. In the end, the directed graph has n-paths with, for each, the final node with the overall score. This visualization makes it easier to see the differences between the assumptions and the final score. An example of the visualization of the beginning of the oriented graph can be seen in Figure 3.6.

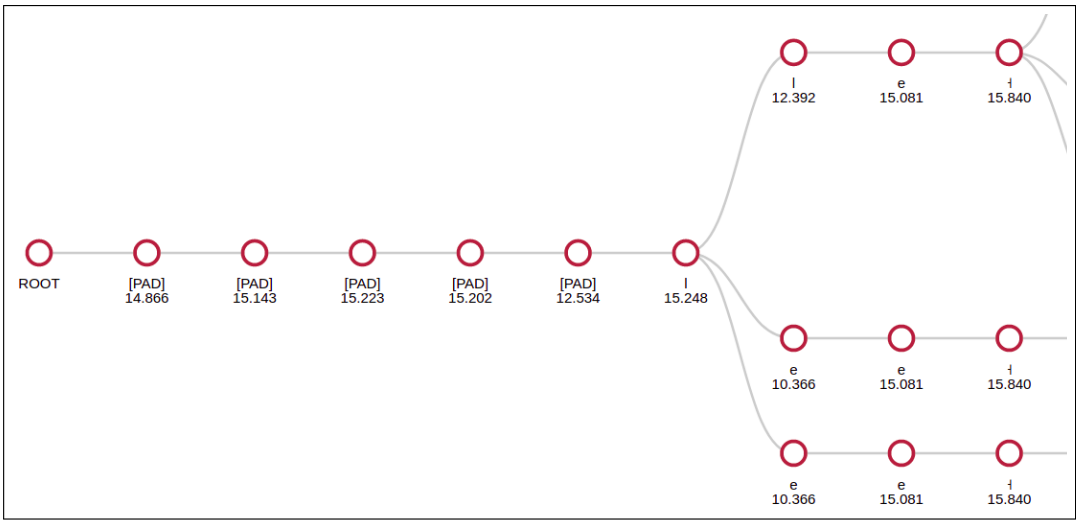


FIGURE 3.6: Beginning of an oriented graph to visualize the 10-best hypotheses generated by the beam search algorithm of a Na test set sentence.

3.5.2 Word-based language model with KenLM

A standard method in ASR to improve the predictions of the model is to use the beam search algorithm with an independently trained language model (LM). The use of a language model has been shown to significantly improve the accuracy of speech recognition systems (Heafield et al., 2013). The language model probability can be included into the beam search, and will re-score the list of n-best hypotheses (Hrinchuk, Popova, and Ginsburg, 2020). The language model can be included as a factor using,

$$Y^* = \underset{Y}{\operatorname{argmax}}\, p(Y|X) \cdot p(Y)^{\alpha} + L(Y)^{\beta} \tag{3.8}$$

where $p(Y|X)$ is the CTC conditional probability, $p(Y)^{\alpha}$ the language model probability, and $L(Y)^{\beta}$ the word insertion bonus. However, the integration of a LM can be difficult in the current CTC systems. The use of a LM in a CTC was only tested on a LM constructed with a large amount of data – the benchmark Librispeech defines a LM with 200K words, other corpora contain approximately 1M words (Ruder, 2021). Moreover, the LM is defined with a different corpus that the one to learn the phonetic model. These are not the conditions found for Japhug and Na where the corpora are resource-limited and in an IPA-based format.

We describe here the use of a word-based language model. The chosen library[10] supports a KenLM n-gram language model (Heafield, 2011), which uses the modified Kneser-Ney smoothing. CTCdecode library was adopted because it proposes an implementation of CTC beam search decoding for PyTorch, and several people have demonstrated its efficiency for `wav2vec 2.0` decoding[11]. To construct the KenLM for the Na, 2,110 sentences were used. For the Japhug, we built a KenLM with 28,660 sentences. The data come from the same corpora as those used to fine-tune the `XLSR` model (i.e. the training data).

3.5.3 Optimization of the LM parameters

Different parameters had to be set to decode the CTC output with a word-based KenLM language model:

- labels — tokens used to train the model (the vocabulary),
- α — weight associated to the language model probabilities,
- β — weight associated with the number of words within the beam,
- the beam width — extent of the beam search, the higher the value, the higher the probability of finding the top beams.
- cutoff_top_n — cutoff number in pruning, meaning that only the top n characters with the highest probability in the vocabulary will be taken into account,
- blank_id — index of the CTC blank token.

An optimization script was proposed by the Language Technologies Unit of Prifysgol Bangor University[12] to train a KenLM language model[13] with Optuna, an open source hyperparameter optimization python library[14] (Akiba et al., 2019). In this specific case, this framework was used to set α and β parameters. Both beam width and top k hypotheses parameters have been initialized to 100. The python script was adapted to fit the requirements regarding the previously implemented scripts.

This optimization step was applied to different n sizes of KenLM language models, specifically on 2-, 3-, and 4-grams, on the two fine-tuned models for each language on the test sets (cf. Table 3.9).

	`xlsr-na-180`	`xlsr-jya-600`
`2-gram KenLM`	$\alpha = 2.77$, $\beta = 0.12$	$\alpha = 2.058$, $\beta = 2.97$
`3-gram KenLM`	$\alpha = 2.54$, $\beta = 0.01$	$\alpha = 1.89$, $\beta = 0.32$
`4-gram KenLM`	$\alpha = 2.51$, $\beta = 0.0048$	$\alpha = 1.85$, $\beta = 0.006$

TABLE 3.9: α and β parameters set up by the Optuna optimization framework to train different n-gram KenLM language models.

[10] https://github.com/parlance/ctcdecode
[11] https://discuss.huggingface.co/t/language-model-for-wav2vec2-0-decoding/4434
[12] http://techiaith.bangor.ac.uk/
[13] https://github.com/techiaith/docker-wav2vec2-xlsr-ft-cy/blob/main/train/python/train_kenlm.py
[14] https://optuna.org/

3.5.4 Results

Table 3.10 refers to the WER and CER scores for the predictions computed by the best fine-tuned models of each language when decoding with different n-gram size KenLM on the test set.

Model	n-gram KenLM	WER (%)	CER (%)
`xlsr-na-180`	2	42.51	10.88
	3	42.18	10.54
	4	42.13	10.53
`xlsr-jya-600`	2	19.15	8.04
	3	19.28	8.1
	4	19.28	8.1

TABLE 3.10: WER and CER on the test sets with the xlsr-na-180 and the xlsr-jya-600 models by using different n-gram KenLM language models.

The word-based KenLM language model is counter-intuitive. It does not outperform the previous results from Table 3.3. With `xlsr-na-180`, we observe a WER score of 42.13% with the 4-gram KenLM language model, where the WER score in Table 3.3 is equal to 41.51%. The same behavior is seen with `xlsr-jya-600` with a WER score of 19.15% against 18.56% in Table 3.3 with the 2-gram KenLM language model.

The comparison between the oracle scores (cf. Table 3.8) and the KenLM language models on the test set shows a higher gain by the oracle scores. The WER is equal to 34.2% when computing the oracle WER on the top 250 hypotheses with the `xlsr-na-180` against 42.13% using a 4-gram KenLM. Similarly, we observe for the `xlsr-jya-600` a WER of 15.31% from the oracle WER top 250 hypotheses against 19.28% with the 4-gram KenLM.

By training several models with less training data on the Japhug, for example with 250 minutes, the KenLM gains are greater on the test data. With the `xlsr-jya-250`, the WER on the test set is 23.9%, which drops to 20.28% with a 4-gram KenLM.

3.5.5 Learning curves

An important aspect of the training of a ML model consists in assessing how the model acts with different training dataset sizes. Learning curves are a suitable measure to diagnose problems such as underfitting or overfitting, and see if the datasets are correctly representative. It is also a way to find out how much data is needed to get “correct” performance (this depends on the linguists’ requirements), and whether the approach can be generalized to other languages.

Figure 3.7 reports the learning curves on the Na and the Japhug data. For different training sizes, the graph shows the CER on the training set. In this case, we see how much data are required to get high performances.

We observe a significant decrease of the CER scores when the training size increases from 12 minutes to 24 minutes. It continues to decrease for the Na, but the performances on the Japhug reaches a threshold around 150 minutes of training size with a CER between 11 and 12%. However, the global observation here is that overall good performances are reached with less than 1 hour of training data. It shows that this model can be trained from very little labeled data and having more training data does not necessarily bring more information. This opens the door wide to a large-scale use of this approach for linguistic documentation,

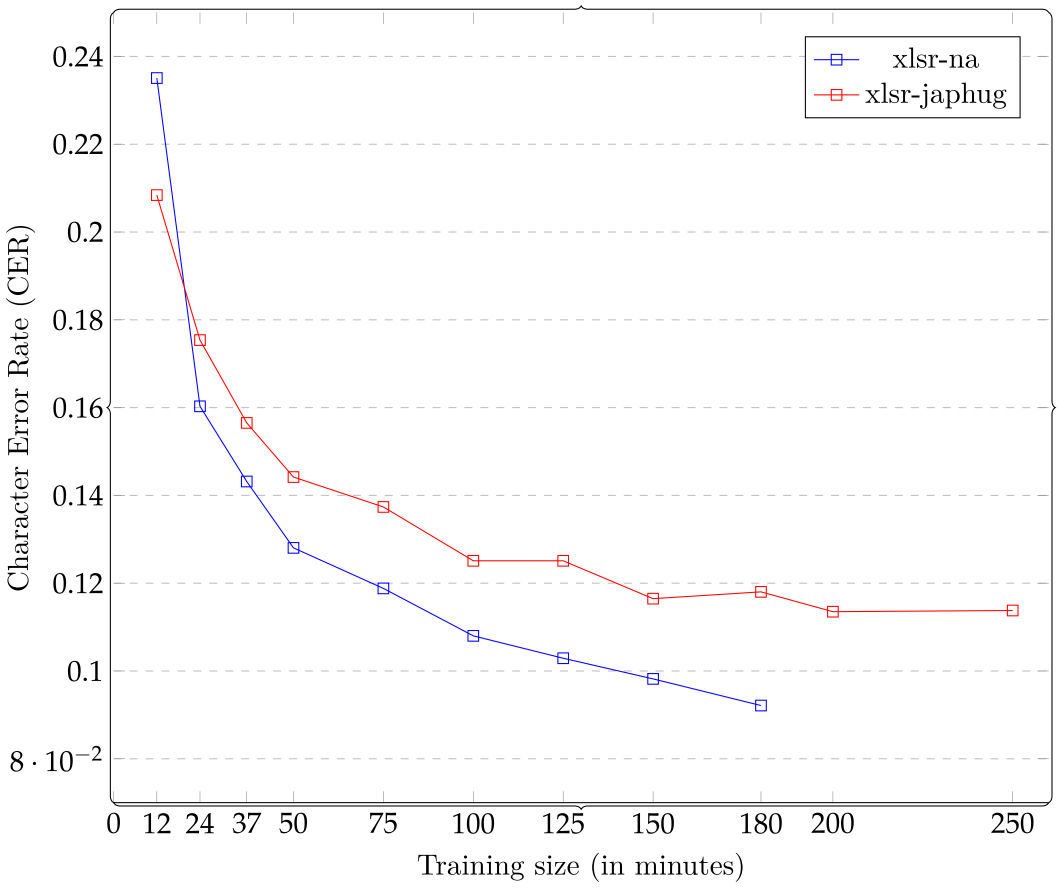


FIGURE 3.7: CER with respect to different training sizes (in minutes) when fine-tuning the XLSR-53 pre-trained model on the two low-resource corpora, the Yongning Na and the Japhug.

as the amount of labeled data corresponds to an amount that can realistically be expected in language fieldwork settings (a corpus size that field linguists can produce manually).

3.6 Complementary experiments

The results reported in the previous two sections raise several questions. The ultimate goal to build Automatic Speech Recognition models for the language documentation workflow is to use the models to produce the transcription of untranscribed speech files. While WER and CER allows us to evaluate the performance of the system quantitatively, these scores do not indicate whether the systems will be useful in practice. In this context, the Section 3.6.1 presents experiments and feedback on the produced transcriptions by linguists on speech files of both studied languages. Through numerous discussions in the course of the present project, an interesting research path has emerged as clearly worthy of investigation: estimating to what extent a fine-tuned model could possibly be used to transcribe speech files of a related language (see Section 3.6.2). Another issue that recurs in exchanges with linguists is the treatment of multilingualism: code-switching and blending of materials from two or more languages is common in minority languages, and hence, is present in fieldwork corpora. Thus, as explained in Section 3.2.1, the Japhug transcriptions contain Chinese characters. We will see in Section 3.6.3 how the model behaves with Chinese characters converted into Pinyin format, an international transcription (romanization) of Mandarin Chinese pronunciation.

3.6.1 Predicting unseen speech files

An interesting experiment consists in using the best fine-tuned models on unseen audio files to evaluate the quality of the output, and its usability in the intended workflow: as part of language documentation, description and conservation.

The audio file entitled "Appeal to the gods to settle a quarrel"[15], available in the Pangloss Collection, was used as a test file for the Na language. The speech was cut into small segments of 15 seconds. The first 5 segments were corrected and evaluated by Alexis Michaud. Table 3.11 prints the transcriptions suggested by the fine-tuned model.

Ref:	ə˧ʝi˧-ʂɯ˥ʝi˩-dʑo˩, ə˩-gi˩, zo˩no˥, hĩ˧ ʈʂʰɯ˧-dʑo˩, əəə... ɖʐwæ˧ ɖʐwæ˥-hwɤ˩ hwɤ˩, mmm... pi˧-dʑo˩, ʈʂɯ˧ʈʂɯ˩ ɻæ˧ɻæ˧ tʰv̩˧, ɖʐwæ˧ ɖʐwæ˥-hwɤ˩ hwɤ˩ tʰv̩˩ pi˧-kv̩˩ mæ˩,
Hyp:	ə˧ʝi˧ʂɯ˥ʝi˩dʑo˩ ə˩gi˩ zo˩no˥ hĩ˧ʈʂʰɯ˧dʑo˩ əə... ɖʐwæ˧ ɖʐwæ˥hoɤ˩ mə... pi˧dʑo˩ ʈʂɯ˧ʈʂɯ˩ ɻæ˧ɻæ˧tʰv̩˧ ɖʐwæ˧ ɖʐwæ˥hwɤ˩hɤ˩ tʰv̩˩ pi˧kv̩˩mæ˩
Ref:	tʰi˩˥ ə˧ʝi˧-ʂɯ˥ʝi˩-dʑo˩ tʰi˩˥ zo˩no˥ mmm... sɯ˧pʰi˧-ki˧ qwɤ˧ qwɤ˩ bi˩-kv̩˩ mæ˩.
Hyp:	tʰi˩˥ ə˧ʝi˧ʂɯ˥ʝi˩dʑo˩ tʰi˩˥ zo˩no˥ mm... sɯ˧pʰi˧ki˧ qwɤ˧qwɤ˩bi˩kv̩˩mæ˩
Ref:	sɯ˧pʰi˧-ki˧ le˧-qwɤ˧ qwɤ˩-se˩-dʑo˩ tʰi˩˥ sɯ˧pʰi˧ no˧ʈ̩˩ dʑɤ˩ pi˧ mɤ˧-ʁo˧, njɤ˧ʈ̩˩ dʑɤ˩ pi˧ mɤ˧-ʁo˧, ɲi˧ʑi˩ do˧bv̩˧ lɑ˧-kv̩˥-ze˥ mæ˩ !
Hyp:	sɯ˧pʰi˧ki˧ le˧qwɤ˥qwɤ˩se˩dʑo˩ tʰi˩˥ sɯ˧pʰi˧ no˧ʈ̩˩ dʑɤ˩ pi˧ mɤ˧ʁo˧ njɤ˧ʈ̩˩ dʑɤ˩ pi˧ mɤ˧ʁo˧ ɲi˧ʑi˩ do˧bv̩˧ lɑ˩kv̩˩ze˩mæ˩
Ref:	tʰi˩˥, do˧bv̩˧ lɑ˧˥ dʑo˩ tʰi˩˥, wɤ˩˥ le˧-ɖʐwæ˧ ɖʐwæ˥ le˧-ɖʐwæ˧ ɖʐæ˥ le˧-ɖʐwæ˧ ɖʐwæ˥ -dʑo˩ tʰi˩˥, mɤ˧-tsɤ˧, mmɤ˧ho˧ ho˥...
Hyp:	tʰi˩˥ do˧bv̩˧ lɑ˧˥ dʑo˩ tʰi˩˥ wɤ˩˥ le˧ɖʐwæ˧ɖʐæ˥ le˧ɖʐwæ˧ʐwɤ˥ le˧ʐwæ˧ʐwæ˥dʑo˩ tʰi˩˥ mɤ˧tsɤ˧ mv̩˧ho˧ho˧
Ref:	ʈʰæ̃˧ ho˩ho˥ mɤ˧tʰɑ˧ ho˧ho˥ mɤ˧tʰɑ˩dʑo˩ tʰi˩˥ əəə... sɯ˧pʰi˧ɳɯ˧ no˧sɯ˩kv̩˩ tʰɑ˧ɖʐwæ˧ ɖʐwæ˥ze˩ ɖæ˩mi˧qo˧ kɤ˧ʈʂɯ˩ ʝi˩ hõ˩ ! pi˧kv̩˩tsɯ˩ mv̩˩
Hyp:	ʈʰæ̃˧ ho˩ho˥ mɤ˧tʰɑ˧ ho˧ho˥ mɤ˧tʰɑ˩dʑo˩ tʰi˩˥ əəə... sɯ˧pʰi˧ɳɯ˧ no˧sɯ˩kv̩˩ tʰɑ˧ɖʐwæ˧ ɖʐwæ˥ze˩ ɖæ˩mi˧qo˧ kɤ˧ʈʂe˩ hĩ˩hõ˩ pi˧kv̩˩tsɯ˩ mv̩˩

Table 3.11: Samples of the predicted transcriptions by the xlsr-na-180 model of the "Appeal to the gods to settle a quarrel" speech file. In red, the deletions, insertions and substitutions.

As a global view, Alexis Michaud points out the high quality of the predictions, probably at the (unbridgeable) upper limit of what is possible in a "phonemic" level (without filtering through a word search or using a language model, or even checking that the sequences are phonologically well formed). Taking the example of mis-transcription of /hwɤ˩/ as /hoɤ˩/, a first thing to note is that [oɤ] and [wɤ] are phonetically really close, so that, from a phonetic point of view, the mistake is not at all egregious. Using a phonotactic system would allow the detection of wrong sequences such as /hoɤ˩/, as the sequence /oɤ/ is not well-formed in the Na language: the two vowels /o/ and /ɤ/ cannot follow each other inside the same syllable. The actual sequence can only be a semi-vowel combined with a vowel, such as /wɤ/, which, together with the (correctly detected) initial consonant and the tone, yields the syllable /hwɤ˩/.

Among the global remarks, Alexis Michaud also pointed out:

- the misprediction of /hwɤ/ twice in the first sentence,
- the mistakes in tones such as /˧/ in /˥/,

[15]https://doi.org/10.24397/pangloss-0004857

- errors in the fourth sentence when the speaker repeats 3 times, very quickly, the same expression, swallowing their words a little.

Model	Words count	WER (%)	CER (%)
`xlsr-na-180`	71	38.46	5.73

TABLE 3.12: WER and CER of the predictions by the xlsr-na-180 model of the unseen speech file entitled "Appeal to the gods to settle a quarrel".

Overall, by computing the CER and the WER scores from the examples below, we see that the scores are really close to what was observed on the test set with the `xlsr-na-180 model` (cf. Table 3.3). Keep in mind that these scores were only measured on a set of 5 sentences. A future work would be to generate transcriptions for several hundred audio segments to be corrected afterwards, and reports the scores.

Guillaume Jacques has provided the audio file entitled *hist150908_qianli_xundi.wav* to transcribe. Because the speech file lasts more than 36 minutes, we only cut the beginning of the file (approximately 2 minutes) into small segments of 10 seconds resulting in 15 samples to predict. Guillaume Jacques corrected the generated transcriptions which were taken as the reference sentences. Table 3.13 shows the CER and WER scores between the reference and the predicted sentences.

Model	Words count	WER (%)	CER (%)
`xlsr-jya-600`	236	5.48	1.34

TABLE 3.13: WER and CER of the predictions by the xlsr-jya-600 model of 15 speech segments from the unseen speech file entitled *hist150908_qianli_xundi.wav*.

The prediction computed by the model is printed in the example below as well as the reference sentence. Words starting with an @ refer to Chinese words.

Ref: tɕendɤre nɯ ɯqhu tɕe tɕendɤre kɯki @zhangxiaobing nɯnɯ @henan nɯtɕu lorɤʑi qhe
Hyp: tɕendɤre nɯ ɯqhu tɕe tɕendɤre kɯki @zhangxiaobin nɯnɯ @huolan nɯtɕu lorɤʑi qhe

The CER and WER scores are very low. A few errors (written in red) are observed in this example — as well as in other predictions (see appendix A.5). From a general point of view, Guillaume Jacques described the predictions as "an impressive result and beyond expectations". The main errors come either from poorly audible parts of sentences, which are also a problem for him, or from Chinese words. Concerning the latter, an example is *henan* 'Henan Province' , which is transcribed as *huolan*: as noted above for Na, the transcription here cannot be said to be inaccurate from a phonetic point of view, given the local pronunciation of Chinese (Sichuanese). If one tried to write Sichuanese in Pinyin (a system designed for Beijing Mandarin), then *huolan* would be a very good match. Thus, all in all, the performance obtained was deemed truly brilliant.

3.6.2 Transfer learning on another language

As explained in Joshi et al., 2020, Transfer Learning (TL) is getting attention in ASR systems as a way to transfer knowledge about one language to the processing of another. With this in mind, we wonder if the knowledge acquired during the fine-tuning of the model on Japhug data could be use to transcribe audio files in a related language, namely Situ.

Situ is a language spoken in Sichuan, China. It is part of the rGyalrong language branch, the same as for the Japhug. The corpus was provided by Shuya ZHANG and contains two speech files, "Mao he laohu"[16] and "Stongsen Mnyasca"[17] with their corresponding transcriptions. The same data preprocessing as in Section 3.6.1 was applied involving the split of the speech files into small segments. To conduct the experiment, 3 segments of the "Mao he laohu" file were created. The `xlsr-jya-600` model was used. The predictions are displayed in Table 3.14.

Ref:	kəscâ-j kəscâ-j mənaŋorɐnə, tɐrú nə khə́ŋ ɲɟe mənaŋorɐ nə, chiɛ́-s na-kə́-nə-ntɕ ka-tsə̂ nə̌-ŋɐs.
Hyp:	tkɯtrɤr kɯrse ɯjto re nɤ tɤru tanɯmnditɤɕe zno kɣts kotcu ma
Ref:	majnə tɐrú kə mənaŋorɐ nə, khə́ŋ tə rə-chə̂ nəvlé-ŋ râ rɐ-kə-səsô-u nə́-ŋɐs ko.
Hyp:	tɤjrɯβɣɯno re nɤ rchi nɯ brɤŋgraʁ ra ɣɯ pɯkɯsɤnɤ suko
Ref:	tɐrú tə kə-tsitsí, avə, kə-tsitsí wo-ka-viɛ ~ viɛ̂ kə-dân tə kənə, kə-mkhâs na-kə-ŋôs nə̌-ŋɐs.
Hyp:	tɤʁru tɤkɯtɕɯtɕi tutsij kɯda nɯ ŋɯ nɯ kɯmqhazna akɯŋu nɤ

TABLE 3.14: Predictions from the xlsr-jya-600 model on 3 segments from the "Mao he laohu" speech file.

We notice that the predictions do not match the reference transcriptions. This result was to be expected because the fine-tuned model only knows the vocabulary on which it was trained, in this case, the vocabulary of the Japhug language (not to mention significant differences in phonetics and phonology between Japhug and Situ). Even though some predictions are phonetically close to the reference transcriptions, such as tɐrú predicted as tɤru, and tə kə-tsitsí predicted as tɤkɯtɕɯtɕi, the conclusion for the present is that, clearly, the model created for one language only gives high-level performance for the same language, and across-the-board application to a related language should not be expected to yield acceptable results (results that can profitably be used by a linguist for further work).

3.6.3 Handling of Chinese characters in the Japhug transcriptions

The Japhug corpus contains audio files in which the speaker talks in both Japhug and Southwestern Mandarin.

From the file entitled *hist-14-tApitaRi*[18], we initially had a sentence with no Pinyin romanization of passages in Chinese, which were transcribed using Chinese characters. The sentence thus switched from International Phonetic Alphabet (for Japhug) to Chinese characters, thus: tɕe ɯ-me nɯnɯ andi 小水沟 kɯre tha-zmɤrʑaβ-nɯ tɕe, nɯre thɯ-ɣe. The new version provided by the linguist (Guillaume Jacques) includes a Pinyin transcription, e.g. tɕe ɯ-me nɯnɯ andi @xiaoshuigou kɯre tha-zmɤrʑaβ-nɯ tɕe, nɯre thɯ-ɣe. Note that the Pinyin characters are in common with the Japhug characters. The preprocessing step on these new data involved the deletion of the @ mark, but also of the Chinese characters that were kept in addition to the Pinyin transcription. By taking 10,000 sentences as training set, we obtain the following CER score according to the epoch number in Figure 3.8 during the training time. The same hyperparameters as in Section 3.2.4 were used to fine-tuned the `XLSR-53` model.

[16] https://doi.org/10.24397/pangloss-0007314
[17] https://doi.org/10.24397/pangloss-0007316
[18] https://doi.org/10.24397/pangloss-0003507

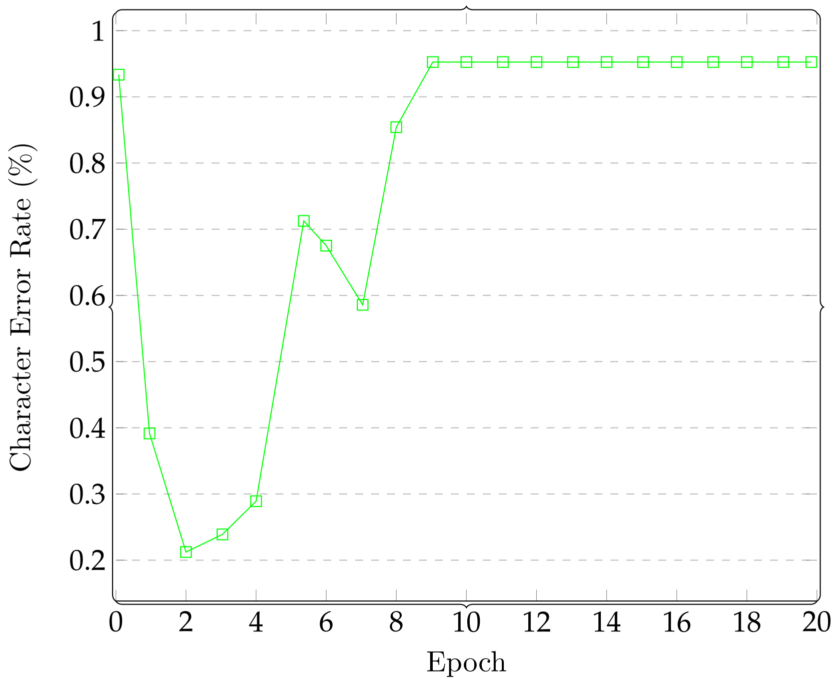


FIGURE 3.8: Character error rate on the training set for Japhug as training progresses (up to 20 epochs), using the XLSR-53 model.

The learning curve shows that the model is unable to learn. The best CER score is seen on epoch 2 with an error rate of more than 20%. We can explain the behavior of the training because the Pinyin transcriptions have common characters with the Japhug transcriptions, but it does not necessarily refer to the same sound. To support this statement, we conducted an evaluation on the test set which assessed the character error rate by:

- taking all the transcriptions,
- retrieving only the words in Pinyin and their predictions,
- keeping only the transcriptions and corresponding predictions without the Pinyin parts.

Each character of the reference is retrieved with the corresponding hypothesis. Then the Error Rate (ER) is computed by dividing the number of false predictions with the number of occurrences of the character. The scores are displayed in Table 3.15.

We notice a higher error rate on the Pinyin transcriptions on the character level. Especially, if we take the character *c*, the CER on the Pinyin transcriptions is equal to 66.67% in comparison with the Japhug transcriptions with a score of 13.1%. This is to be expected, since *c* and *ch* in Pinyin transcribe aspirated affricates, /ts^h/ and /tʃ^h/, whereas in the International Phonetic Alphabet it refers to a palatal stop: the sounds have different places of articulation and modes of articulation.

Generally, the CER score on the Pinyin is far worse than the score associated to the Japhug transcriptions. A simple solution to implement would be to transform the Pinyin transcriptions into a specific encoding that does not have common characters with the Japhug one.

Characters	**ER global (%)**	**ER on Pinyin (%)**	**ER without Pinyin (%)**
x	13.16	66.67	8.57
c	14.86	66.67	13.10
j	16.67	66.67	14.76
y	63.64	60.00	0.00
e	8.49	55.56	8.16
r	7.20	50.00	7.03
u	13.59	40.91	12.18
g	14.67	38.46	9.68
d	11.24	33.33	10.00
w	20.41	33.33	20.00
i	15.01	33.33	14.18
b	19.51	33.33	17.14
o	13.15	30.77	12.78
n	9.78	27.59	9.15
l	12.61	22.22	10.91
h	10.42	12.50	10.65
s	8.16	7.69	8.18
a	8.50	6.45	8.45
k	8.13	0.00	8.51
z	15.52	0.00	14.91
p	7.17	0.00	6.76
m	9.41	0.00	9.43
q	14.50	0.00	14.62
t	7.52	0.00	7.66
f	9.46	0.00	9.59

TABLE 3.15: Comparison of the Error Rate per characters by taking all the predictions (Error Rate global), by taking only the predictions of Pinyin transcriptions (Error Rate on Pinyin), and by taking the predictions without the Pinyin (Error Rate without Pinyin).

Chapter 4

Second approach: Wav2vec Unsupervised

As explained in Section 1.1, a second approach that has proven to be efficient in building ASR systems in a low resource context was recently proposed in Baevski et al., 2021. The main advantage to build unsupervised systems is that it does not require any labeled data. An unsupervised approach could save a huge amount of human labeling costs for developing ASR systems by focusing on massive unlabeled speech data.

In this Chapter, we first introduce the state-of-the-art approaches for unsupervised speech recognition (see Section 4.1). We continue with the description of the framework (see Section 4.2), and the experiments carried out (see Section 4.3). We finally discuss about the obtained results (see Section 4.4).

4.1 State-of-the-art

Thanks to the advent of semi-supervised learning (Xu et al., 2020; Park et al., 2020), and self-supervised learning (Oord, Li, and Vinyals, 2018; Chung and Glass, 2018; Chung et al., 2019; Baevski et al., 2020), an important breakthrough has been observed on speech recognition performance on the famous English Librispeech benchmark (Panayotov et al., 2015). As we have seen in Chapter 2, these approaches require transcribed speech data. Specifically for low-resource languages, these types or data are not always available. Notably, there are only speech recognition systems for 125 languages in the famous Speech-to-text Google tool (Google, 2021b).

Successful results have been observed in machine translation systems with the use of no labeled training data (Conneau et al., 2017; Lample et al., 2017; Artetxe et al., 2017). A few works focusing on speech recognition have been conducted in an unsupervised fashion.

We can cite the work of Yeh et al., 2018 in which a fully unsupervised learning algorithm was proposed. This framework is intended to solve two sub-problems: (1) learning a phoneme classifier by taking a set of phoneme segmentation boundaries, and (2) using a classifier to refine the phoneme boundaries. A novel unsupervised cost function was introduced for the resolution of the first sub-problem entitled Segmental Empirical Output Distribution Matching (SEODM) based on the work of Liu, Chen, and Deng, 2017. The second sub-problem uses an approximate MAP approach inspired by the work of Wang, Chung, and Lee, 2017. The experiments were conducted on the TIMIT benchmark dataset[1], an acoustic-phonetic continuous speech corpus which provides broadband recordings of 630 speakers of American English with the corresponding time-aligned orthographic, phonetic and word transcriptions. A Phoneme Error Rate (PER) of 41.6% was computed. While this does not surpass the results from the state-of-the-art supervised systems, it has shown the feasibility of building unsupervised speech recognition systems.

[1]https://catalog.ldc.upenn.edu/LDC93S1

Two complementary papers based their approach on adversarial learning. The first one by Liu et al., 2018 proposed an unsupervised phoneme recognition system, or in other words, a mapping between audio signals and phoneme sequences without any phoneme-labeled audio data. The method first clusters the embedded acoustic tokens, and then uses a Generative Adversarial Network (GAN) to propose a mapping between the cluster sequences and the unknown phoneme sequences. The preliminary results showed an unsupervised phoneme recognition accuracy of 36% on the TIMIT dataset. In Chen et al., 2019, they proposed a GAN, with a Generator G and a Discriminator D which improve their performances by learning from both. A complementary module, a set of Hidden Markov Models (HMMs) was developed, whose purpose is to refine is to refine the generated labels of the GAN. In comparison with the previous state-of-the-art approaches, a PER of 33.1% was achieved on the TIMIT dataset.

These previous studies have shown that unsupervised speech recognition is possible. However, the error rates remain high and the experiments were only conducted on the TIMIT benchmark dataset. The proposed model `wav2vec-U`, or `wav2vec Unsupervised` (Baevski et al., 2021), leverages self-supervised speech representations from `wav2vec 2.0` (Baevski et al., 2020) to segment unlabeled audio data with a k-means clustering method, and learn, with adversarial training, a mapping between the representations and the phonemes. The experiments were conducted on multiple benchmark datasets, as well as different settings and languages. On the famous benchmark TIMIT dataset, a PER of 11.3% was computed as well as a WER of 5.9% on the Librispeech benchmark. Moreover, an evaluation on European languages and non-European low-resource languages was presented which demonstrated the viability of this approach. It is in this low-resource context that the experiments in this work were carried out (see Section 4.3). We first present the model in Section 4.2.

4.2 The `wav2vec-U` framework

A global view of the framework is printed in Figure 4.1. This approach is not end-to-end, meaning that we have to launch and train different modules to obtain the results.

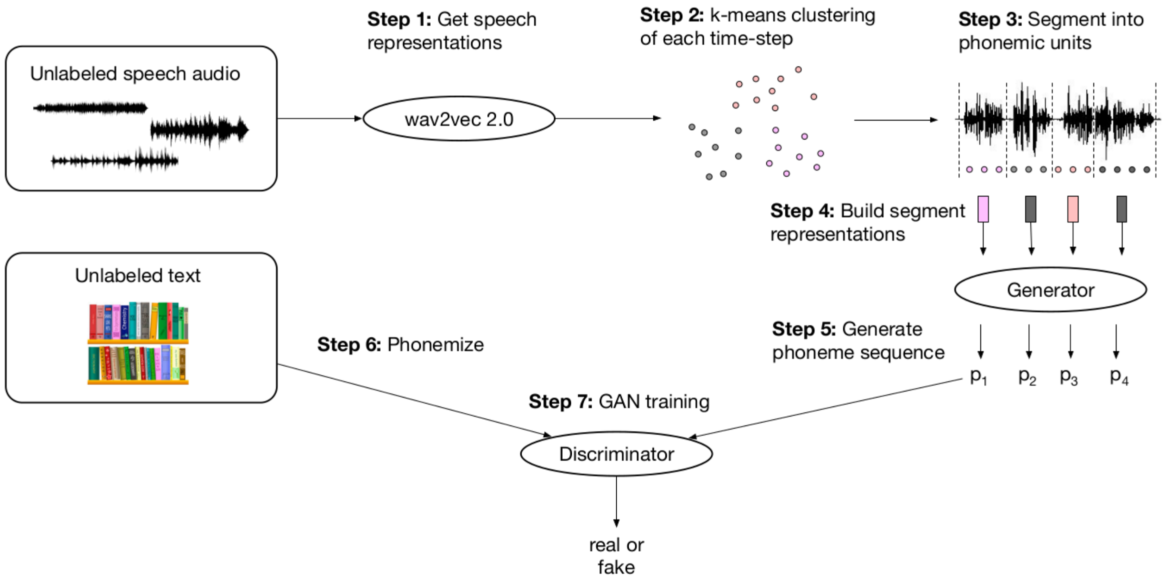

Unlabeled speech audio
Step 1: Get speech representations
wav2vec 2.0
Step 2: k-means clustering of each time-step
Step 3: Segment into phonemic units
Step 4: Build segment representations
Generator
Step 5: Generate phoneme sequence
p1 p2 p3 p4
Unlabeled text
Step 6: Phonemize
Step 7: GAN training
Discriminator
real or fake

FIGURE 4.1: Illustration of the wav2vec Unsupervised framework taken from Baevski et al., 2021.

The first step is the learning of self-supervised representations with `wav2vec2.0` using only unlabeled speech files (see Section 4.2.1). The identification of clusters in the representations comes in second place with the use of a k-means clustering technique (see Section 4.2.2). Then, the segment representations are built by mean pooling `wav2vec 2.0` representations with a Principal Component Analysis (PCA) to keep the most important features and reduce the dimensionality of the data (see Section 4.2.3). These representations are then fed to the generator to produce a phoneme sequence which is used as input to the discriminator similarly to phonemicized unlabeled text (see Section 4.2.4). The last step is the training of a GAN model (see Section 4.2.5).

4.2.1 Self-supervised Learning of Speech Audio Representations

First of all, the representations of speech audio signal are learnt using self-supervised learning from `wav2vec 2.0` model[2]. Specifically, the context Transformer network $c_1, ..., c_T$ representations will be used.

This step involves a preprocessing treatment over the speech file and the deletion of silences. In the case of audio data, it may happen that some parts do not correspond to any transcription, i.e. silence sections. `rVad` (Tan, Dehak, et al., 2020) proposes an unsupervised voice activity detection model able to identify which segments in the speech file correspond to silences. The identified sections are then removed.

4.2.2 Speech audio segments identification

Once we have the speech audio representations, we identify the speech audio segments. The purpose being to get segments from speech that correspond to meaningful units and can thus be matched with phonemes. A simple method to apply from the `wav2vec 2.0` speech representations $c_1, ..., c_T$ is to perform k-means clustering to identify K clusters. The Faiss library[3] implements a fast clustering method (Johnson, Douze, and Jégou, 2019). Then, once the k-means clustering method produces the clusters, each contextual representation c_t is labeled into the corresponding cluster ID $i_t \in 1, ..., K$. A boundary between speech segments is introduced if the cluster ID changes. In a global perspective, segmentation is a key feature to predict correct output sequence if the input representation boundaries are properly defined (Chung et al., 2018).

4.2.3 Audio segment representations

The next step following the segmentation of speech audio representations is the construction of audio segment representations. A Principal Component Analysis (PCA) is performed over all speech representations from the `wav2vec 2.0` training set output. In few words, PCA is a dimensionality-reduction method intended to reduce the dimensionality of large datasets (Jaadi, 2021). It tries to create the best data distribution representation by finding the most relevant combination of features. A main advantage is the efficiency to visualize and analyze data for machine learning algorithms. For a specific segment, the corresponding PCA representations are mean-pooled. The segment will have an associated average representation thanks to a selection of the most important features during the PCA. Pairs of adjacent segment representations are also mean-pooled to mitigate the effects of segment boundary noises. The final output gives sequences of speech segment representation $S = s_1, ..., s_t, S \in \mathcal{S}$ for a given utterance.

[2]See Section 2.3

[3]`https://github.com/facebookresearch/faiss`

4.2.4 Preprocessing of unlabeled text data

`Wav2vec-U` involves the use of unlabeled text data. A preprocessing step is performed whose goal is to create suitable units for unsupervised learning.

We distinguish, first, the phonemicization of the text. Each sequence of words Y that makes up the text is converted into a corresponding sequence of phonemes $P = [p_1, ..., p_M] \in O^*$, with O as the phoneme dictionary. It is indeed easier to learn a mapping between phonemes and speech audio segments in comparison with words or letters.

The second step goes by the name of silence token insertion to deal with the silences still encountered in the speech audio. Precisely, unsupervised silence removal that was applied over the speech audio is not entirely accurate. The unsupervised model may therefore label some audio segments with a phonemic silence token, i.e. `<SIL>`. Silence markers are added into the unlabeled text data, otherwise it will lead to difficulties during the adversarial learning. Indeed, the model will predict a phoneme to label silences which decreases the performance. This silence token is defined in the beginning, end and inside according to a defined rate of silence token insertion of the phonemicized unlabeled text sentences.

4.2.5 Model architecture

The unsupervised speech recognition model architecture is implemented with a Generative adversarial network (GAN) (Goodfellow et al., 2020).

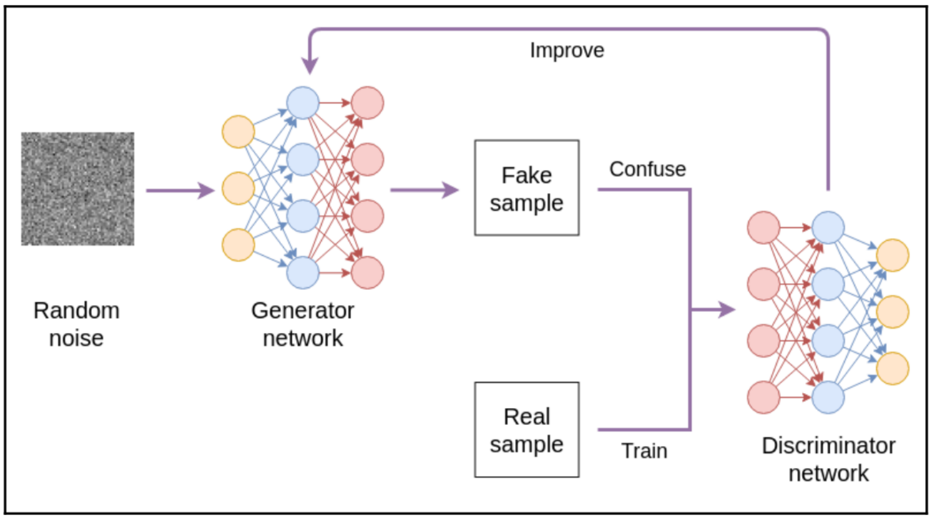


FIGURE 4.2: Basic process of a GAN showing the interaction between the generator and the discriminator networks (Hany and Walters, 2019).

As seen in Figure 4.2, the GAN is composed of:

- a generator network $\mathcal{G}$ which generates fake samples. The goal is to make the fake samples as close as possible to the real samples, indiscernible by the discriminator.
- a discriminator/critic network $\mathcal{C}$, a classification network, whose job is to tell whether a given sample is false or true.

In other words, the generator does everything possible to deceive the discriminator into making a wrong decision, while the discriminator does everything possible to distinguish fake samples from true ones.

Figure 4.3 illustrates the transformation of the generator output and the phonemicized text to feed them to the discriminator. The input of $\mathcal{G}$ is a sequence of T segment representations $S = [s_1, ..., s_T]$ mapped to a sequence of M phonemes $P = [p_1, ..., p_M]$. The generator

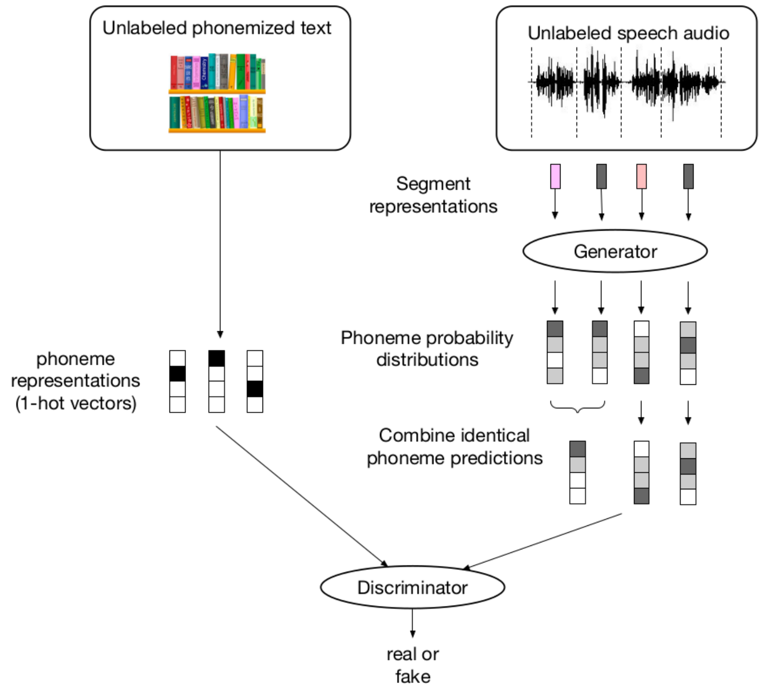

Unlabeled phonemized text
Unlabeled speech audio
Segment representations
Generator
Phoneme probability distributions
phoneme representations (1-hot vectors)
Combine identical phoneme predictions
Discriminator
real or fake

FIGURE 4.3: Illustration of how generator outputs and real phonemicized text are converted into inputs to the discriminator. This schema is taken from Baevski et al., 2021.

network predicts, for each segment, a distribution over the phoneme set O. The phoneme with the highest probability is the output. Moreover, when the same phoneme is consecutively predicted for a set of segment, the average is computed across all possible phoneme predictions ($M \leq T$). The discriminator takes as input a sequence $P^r \in \mathcal{P}^r$ of one-hot vectors of dimension $|O|$ corresponding to the phoneme representations of the phonemicized text or a sequence of outputs from the generator P. Both networks are implemented as a single layer convolutional neural network (CNN), with the discriminator which indicates the probability of a sample to be from the data distribution.

Objective

The GAN objective (Goodfellow et al., 2020) is described as

$$\min_{\mathcal{G}} \max_{\mathcal{C}} \mathop{\mathbb{E}}_{P^r \sim \mathcal{P}^r} [\log \mathcal{C}(P^r)] - \mathop{\mathbb{E}}_{S \sim \mathcal{S}} [\log(1 - \mathcal{C}(\mathcal{G}(S)))] - \lambda \mathcal{L}_{gp} + \gamma \mathcal{L}_{sp} + \eta \mathcal{L}_{pd} \tag{4.1}$$

where $P^r \in \mathcal{P}^r$ is the phonemicized unlabeled text, $\mathcal{G}(S)$ corresponds to the output of the generator (the transcription) when segment representations S are inputs coming from unlabeled speech audio. The first term trains the discriminator to give real transcriptions a high probability. The second term urges the discriminator to assign to generator outputs a low probability. Furthermore, the GAN objective is defined with:

- a gradient penalty $\mathcal{L}_{gp}$ whose goal is to stabilize the training,
- a segment smoothness penalty $\mathcal{L}_{sp}$ to encourage the generator to generate comparable outputs for adjacent segments,
- and a phoneme diversity loss $\mathcal{L}_{pd}$ to penalize the generator network's poor use of the phoneme vocabulary at the batch level.

4.2.6 Unsupervised Cross-Validation Metric

To assess the performance of the proposed unsupervised speech recognition model, a novel metric was introduced entitled unsupervised cross-validation metric. It based its computation on two quantities:

- the language model entropy, which indicates how fluent a given transcription is. This quantity is computed with a language model p_{LM} trained on phonemicized text data.
- the vocabulary usage, which gives the number of phoneme vocabulary used by the model via Viterbi decoding. It is a great indicator to know if the model has a deviant behavior, meaning if it outputs trivial transcriptions.

4.2.7 Decoding

The best checkpoint identified after the GAN training (chosen using the unsupervised metric introduced in the previous paragraph) is used to generate phone labels. The decoding involves a KenLM language model which works better on features before the adjacent timestep mean-pooling step.

4.3 Experiments

We recall that our objective is to know if it is possible to build automatic speech recognition models in a complete unsupervised fashion, but specifically on the two studied low-resource languages, the Yongning Na and the Japhug. We followed the pipeline of the framework described in Section 4.2 that was applied to low-resource languages from CommonVoice in Baevski et al., 2021. We took the code available in the fairseq library[4]. To summarize, we distinguish two main steps: the preparation of audio data, and the preprocessing of the text data.

4.3.1 Datasets

The same corpora were used as for the `XLSR` approach (see Section 3.1). From the preprocessed data, described in Section 3.2.1, three files were created:

- **`train.wrd`** containing the 'cleaned' IPA-based transcription. The first three sentences of the training set file are:

  ʐwæ˩mi˩tʰv̩˩v̩˩˥ dʑo˩ tʰi˩˥ ə˧mi˧ɻæ˩ɳɯ˩ le˧ʐɤ˧
  mɑ˩pv̩˩˥ no˧ ə˧sɯ˥ sɯ˧ʝi˧ mæ˩
  hĩ˧ mo˥hĩ˩dʑo˩ ʑi˧qʰwɤ˧qo˧ tʰv̩˧ne˧ʝi˥ tʰi˧dzi˩

- **`train.ltr`** which is the transcriptions cut by characters. The first three sentences of the training set file cut by characters are:

  ʐ w æ ˩ m i ˩ tʰ v̩ ˩ v̩ ˩ ˥ | d ʑ o ˩ | tʰ i ˩ ˥ | ə ˧ m i ˧ ɻ æ ˩ ɳ ɯ ˩ | l e ˧ ʐ ɤ ˧ |
  m ɑ ˩ p v̩ ˩ ˥ | n o ˧ | ə ˧ s ɯ ˥ | s ɯ ˧ ʝ i ˧ | m æ ˩ |˩
  h ĩ ˧ | m o ˥ h ĩ ˩ d ʑ o ˩ | ʑ i ˧ qʰ w ɤ ˧ q o ˧ | tʰ v̩ ˧ n e ˧ ʝ i ˥ | tʰ i ˧ d z i ˩ |

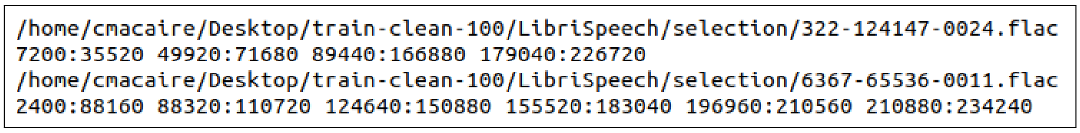
[4]The code is available in https://github.com/pytorch/fairseq/tree/master/examples/wav2vec/unsupervised

- **`train.phn`** containing the transcriptions cut by phonemes. The first three sentences of the training set file cut by phonemes are:

  ʐ wæ ˩ m i ˩ tʰ v̩ ˩ v̩ ˩˥ dʑ o ˩ tʰ i ˩˥ ə ˧ m i ˧ ɻ æ ˩ ɳ ɯ ˩ l e ˧ ʐ ɤ ˧
  m ɑ ˩ p v̩ ˩˥ n o ˧ ə ˧ s ɯ ˥ s ɯ ˧ ʝ i ˧ m æ ˩
  h ĩ ˧ m o ˥ h ĩ ˩ dʑ o ˩ ʑ i ˧ qʰ wɤ ˧ q o ˧ tʰ v̩ ˧ n e ˧ ʝ i ˥ tʰ i ˧ dz i ˩

4.3.2 Preparation of audio data

The preparation of audio data involves, first, the creation of audio files with no silences. To do so, we applied the `rVad` python library[5] which gives the boundaries time on which a silence is identified. Figure 4.4 gives an example of the output produced by the `rVad` method.



```
/home/cmacaire/Desktop/train-clean-100/LibriSpeech/selection/322-124147-0024.flac
7200:35520 49920:71680 89440:166880 179040:226720
/home/cmacaire/Desktop/train-clean-100/LibriSpeech/selection/6367-65536-0011.flac
2400:88160 88320:110720 124640:150880 155520:183040 196960:210560 210880:234240
```

FIGURE 4.4: Beginning of the output file given by the rVad python library with the first line corresponding to the path, and the second line with the silence intervals.

A python script[6] then removes the identified silences.

Next, a bash script[7] preprocesses the audio data. Precisely, the speech audio representations were extracted using the self-supervised learning method from the `wav2vec2.0` model. A pretrained model was used, and the number of the layer from which the representations need to be extracted had to be specified. Once the representations were computed, the k-means clustering method was applied to identify the speech audio segment. Each audio file is encoded as a sequence of cluster IDs corresponding to each identified audio segments. An example of an output produced by the k-means is displayed in Table 4.1.

Audio path:	crdo-NRU_F4_DOG2_Dog2S021.wav
Phoneme sequence:	ʈʂʰ ɯ ˧ n e ˧ ʝ i ˥ tʰ i ˧ tɕ ɯ ˧ ɲ i ˥ ts ɯ ˩ m v̩ ˩
Cluster IDs:	117 26 118 118 103 103 103 0 103 92 7 96 10 125 104 104 79 100 96 32 80 12 12 104 104 88 88 88 88 31 7 96 86 101 97 97 30 30 30 114 43 22 124 124 32 33 33 127 33 110

TABLE 4.1: Output produced by the k-means clustering method on the audio file “crdo-NRU_F4_DOG2_Dog2S021.wav” given the phoneme sequence.

Finally, a PCA with a mean-pool step was performed to construct audio segment representations.

[5]https://github.com/zhenghuatan/rVADfast

[6]https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/unsupervised/scripts/remove_silence.py

[7]https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/unsupervised/scripts/prepare_audio.sh

4.3.3 Preprocessing unlabeled textual data

The preprocessing step associated to the text data was performed with a bash script[8] which has been modified to match the chosen corpora. From the IPA transcriptions, a dictionary was created with words sorted by their frequency (see the first column of Table 4.2) as well as a list of words.

Words dictionary		**Phonemization**		**Phonemes dictionary**	
tʰi˩˥	929	tʰ i ˩˥	929	˧	5733
əəə...	385	əəə...	385	˩	5632
mv̩˩	31	m v̩ ˩	31	i	2617
ʈʂʰɯ˧ne˧ʝi˥	213	ʈʂʰ ɯ ˧ n e ˧ ʝ i ˥	213	ɯ	2328

Table 4.2: Results of the preprocessing step on unlabeled text data from left to right, the words dictionary, the phonemicized words, and the phonemes dictionary.

The next step involved the phonemicization of the text. However, this is not required here because the Yongning Na and the Japhug text data are already in a phonemic format with IPA-based transcriptions. From the list of words, the **`phones.txt`** was created which contains the words cut into phonemes (see the middle column of Table 4.2). A lexicon was generated with a first column containing the words, and a second column with the words into the corresponding sequence of phonemes. The list of phonemes with their corresponding frequency was then added into a phoneme dictionary (see the last column of Table 4.2). The final step was the silence token insertion according to a specific insertion rate.

<SIL> ʐwæ ˩ m i ˩ tʰ v̩ ˩ v̩ ˩˥ dʑ o ˩ tʰ i ˩˥ <SIL> ə ˧ m i ˧ ɻ æ ˩ ɳ ɯ ˩ l e ˧ ʐ ɤ ˧ <SIL>
<SIL> m ɑ ˩ p v̩ ˩˥ n o ˧ <SIL> ə ˧ s ɯ ˥ s ɯ ˧ ʝ i ˧ <SIL> m æ ˩ <SIL>

As we see in the Na examples below, the silence token `<SIL>` is defined at the beginning, the end, and inside the phonemicized sentence according to a specific insertion rate.

Different phoneme-based kenLM language models were defined (4-gram and 6-gram).

The following table 4.3 recaps the parameters set during the preprocessing steps. These were defined according to the one specified in the documentation of wav2vec-U.

parameter	value
`PCA dimensionality`	512
`index layer`	14
`number of clusters`	128
`pretrained wav2vec2.0 model`	XLSR-53

Table 4.3: Preprocessing parameters.

[8] `https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/unsupervised/scripts/prepare_text.sh`

4.3.4 GAN training

A GAN model was trained to build an unsupervised ASR model. The data preparation on the speech audio data and on the unlabeled text data is mandatory to enable the generator to map speech units to text in an unsupervised way. A script was written which specifies:

- the path to the mean-pooled audio segment representations,
- the path to processed text data,
- the KenLM 4-gram phoneme language model,
- the config name file.

Table 4.4 presents the chosen hyperparameters. A complete description of the parameters can be found in the documentation[9].

parameter	value
`code_penalty`	2,4
`gradient_penalty`	1.5,2.0
`smoothness_weight`	0.5,0.75,1.0
`seed`	range(0,5)
`batch_size`	160
`max_update`	150,000

TABLE 4.4: Hyperparameters of the GAN training.

4.4 Results

A first experiment was conducted by taking all the available data from each corpus, 180 minutes as the training size for the Na, and around 25 hours of data for the Japhug. This allows to know the viability of this approach by comparing the obtained results with the one assessed in the article. The results are displayed in Table 4.5.

Dataset	**Training size (in minutes)**	**Valid size (in minutes)**	**Test size (in minutes)**	**UER valid (%)**	**UER test (%)**
Na	180	30	30	86.03	86.3
	42	30	30	89.59	89.02
	30	30	30	83.37	83.48
Japhug	1500	190	190	100	100
	180	30	30	86.48	86.6

TABLE 4.5: UER on the valid and test sets of the Yongning Na and the Japhug with different training set sizes.

[9] https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/unsupervised/config/gan/w2vu.yaml

We notice an Unsupervised Error Rate (UER) of 86.03% on the valid set for the Na, and a UER of 100% for the Japhug. The most likely interpretation is that the GAN 'simply' does not learn a mapping between speech segments and phonemes. We can see it very well in the proposed output of the GAN from the Na:

Ref: h æ ˧ n i ˩ tʰ i ˧ ʑ i ˧ m ɤ ˥ k v̩ ˩ m æ ˩ tʰ i ˧ dz i ˩ kʰ ɯ ˩
Hyp: tʰ m ə ˧ m ɤ ˩ h i ˥ n o ˥ dʑ o ˩ n v̩ ˩ wɤ v̩ wɤ ˩ h ĩ ˥ ˧ ɯ o ˥ z m ˩

In the case of the Japhug model, the predictions are empty.

When using the same amount of training data for the Japhug, we obtain the same UER score as for the Na experiment. An example of an output printed by the model is

Ref: t ɕ e i ɕ q h a n ɤ k i n ɯ r g ɤ t p u n ɯ k ɯ i ɕ q h a k ɯ r ɯ t s h o ŋ w a ɣ ɯ n ɯ n ɯ m ɤ k ɯ p e n ɯ k ɤ ɤ t ɯ ɣ n ɯ n ɯ r a t o k o t s o
Hyp: a m ɕ l d β r a t ɕ p s h p ʂ β m k β s β u β ɣ p i ɣ ʂ a β v t u n e x t ɕ β k h b n a β k ɯ β z t z e

We considered applying the GAN on a smaller amount of training data. By taking 30 minutes as the training set from the Na corpus, we gained over 3 percentage points on the UER score. However, the performance is still very low in comparison with the experiments carried out in Baevski et al., 2021 on several low-resource corpora with a PER score of 25% for the Tatar and the Kyrgyz, and 52.6% PER for the Swahili.

Several factors could explain these observations. A first thought that might be considered is a preprocessing stage wrongly executed. However, we were not able to detect any issues with this step. A second hypothesis could be the misidentification of speech audio representations. But we used the same pre-training model as in Chapter 2 to fine-tune the `XLSR` model. Had the representations been biased, we would not have been able to observe the computed performances.

A solution that is still being explored is to reproduce the experiments on the Librispeech benchmark dataset. Checking out the documentation, and the feedback and discussions on forums, it seems that other people are likewise experimenting difficulty getting similar results on these data reported in Baevski et al., 2021, with an unusually high error rate score.

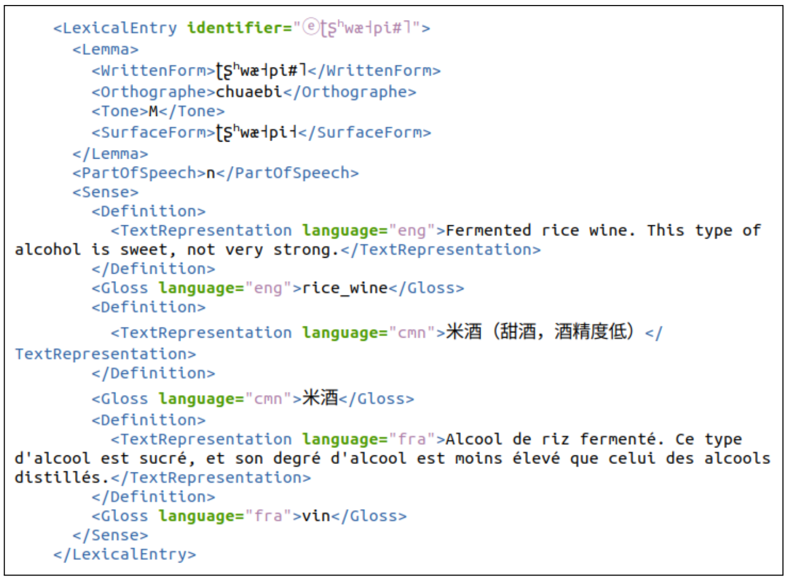
Chapter 5

Work on dictionaries

As presented in Section 1.1, we want to determine how to leverage the various sources of information found in resources gathered by field linguists. The objective is to compensate for the small amount of available training data (transcribed audio). Specifically, this chapter will focus on the dictionaries that are available for the two low-resource languages that were explored in the presented approaches (`XLSR`, `wav2vec-U`), i.e. the Yongning Na & Japhug languages.

5.1 Structure of the dictionaries

The two dictionaries that will be studied are the Na dictionary and the Japhug dictionary. The Japhug dictionary was written by Guillaume Jacques, and the Na dictionary by Alexis Michaud. They are both available in the Pangloss Collection[1] in the PDF, HTML and XML formats. The latter format will be used to extract the data. In more detail, the Japhug dictionary (Japhug-Chinese-French)[2] contains over 7,000 entries, and the Na dictionary (Na-English-Chinese)[3] about 3,000 entries. Figure 5.1 shows an example entry from the Na dictionary. Each lexical entry as a unique ID, and its related forms. The associated senses (meanings) are described, with a corresponding translation. Some of them contain a text representation with an example of a sentence in which the lexical entry is found.



```
    <LexicalEntry identifier="ⓔʈʂʰwæ˧pi#˥">
      <Lemma>
        <WrittenForm>ʈʂʰwæ˧pi#˥</WrittenForm>
        <Orthographe>chuaebi</Orthographe>
        <Tone>M</Tone>
        <SurfaceForm>ʈʂʰwæ˧pi˧</SurfaceForm>
      </Lemma>
      <PartOfSpeech>n</PartOfSpeech>
      <Sense>
        <Definition>
          <TextRepresentation language="eng">Fermented rice wine. This type of
alcohol is sweet, not very strong.</TextRepresentation>
        </Definition>
        <Gloss language="eng">rice_wine</Gloss>
        <Definition>
          <TextRepresentation language="cmn">米酒（甜酒，酒精度低）</
TextRepresentation>
        </Definition>
        <Gloss language="cmn">米酒</Gloss>
        <Definition>
          <TextRepresentation language="fra">Alcool de riz fermenté. Ce type
d'alcool est sucré, et son degré d'alcool est moins élevé que celui des alcools
distillés.</TextRepresentation>
        </Definition>
        <Gloss language="fra">vin</Gloss>
      </Sense>
    </LexicalEntry>
```

Figure 5.1: Example of a lexical entry from the Na dictionary.

[1] https://pangloss.cnrs.fr/dictionnaires
[2] https://pangloss.cnrs.fr/dictionaries_content/japhug/dictionary.pdf
[3] https://pangloss.cnrs.fr/dictionaries_content/na/dictionary_eng_mp3.pdf

5.2 Dictionaries coverage

A first interesting measure is the coverage of the dictionaries on the transcriptions. This will show whether the dictionaries can provide additional information. If coverage of 90% is observed, a correction can be made on the transcriptions (9 words out of 10). On the other hand, coverage of 10% will not bring any information, especially if the correction is made on the word level. For this purpose, the first step included the extraction of the lexical entries and of the linked information for each.

From the Na dictionary, we extracted:

- the <`SurfaceForm`> content of each lexical entry,
- the <`WrittenForm`> content of each lexical entry.

And from the Japhug dictionary, we extracted:

- the <`RelatedForm`> of each lexical entry,
- the <`variantForm`> of the lexical entry lemma,
- and the lexeme of the entry from the <`Lemma`> tag.

Table 5.1 recaps the number of lexical entries extracted from each dictionary.

Dictionary	Number of lexical entries
Na	4,147
Japhug	7,649

TABLE 5.1: Number of extracted lexical entries from each dictionary.

The Table 5.2 below prints some lexical entries that were extracted from both dictionaries. No cleaning steps were performed (removing specific characters, punctuation marks, lowercase, etc.).

Yongning Na	Japhug
si˧	ndzɯɣ
to˧to˧	tɤŋgɤr
ʁo˧qʰwɤ˩	aʑaʁ
gæ˧ɻæ˩	kɤtɕhɯ
ʝi˧qv̩˥	nɯɟɯɣɟɯɣ
mv̩˩zɯ˩ni˥mi˩	sɤmtshɤr
ʐwæ˩mi˩˥	sthɯt
hĩ˧mo˩	rɤmbɯmbri
qʰv̩˩ɖʐæ˩	aŋgɤrŋgɤr
tʰo˧li˧-kʰv̩˧˥	tɯ-rqɤpa

TABLE 5.2: Example of lexical entries extracted from both dictionaries.

The second step included the extraction of the transcriptions and the split into words. To do so, for each file, we retrieved the content of the `<FORM>` tag within the `<S>` tag. An example of a transcription file in the XML format can be seen in Figure 3.3.

A preprocessing step was required to match the words extracted from the dictionaries. Characters such as punctuation marks (?, !, :, {, },), etc.) and special characters (‘D’, ‘F’, ‘̃’, ‘...’, ‘=’, ‘↑’, ‘:’) had to be removed.

The total number of words in the Na corpus is equal to 6,070 words, and 19,617 words in the Japhug corpus.

Linguists will be able to use the implemented script to help them develop dictionaries.

5.2.1 Coverage on the corpora

In this Section, the coverage of dictionaries on all available text data from both corpora is given. The goal is to find out if the words in the transcriptions can be found in the corresponding dictionary and thus if it is useful to use this information in the ASR task.

Yongning Na corpus

Considering the Na corpus, the dictionary coverage is equal to 13.69% when all the transcriptions are taken into account. Some examples of words that are not in the dictionaries and whose frequency is equal or higher than 20 are: əəə..., mɣ̩˩, dʑo˩, ə˩gi˩, mmm..., ɖɯ˧ɣ̩˧, pi˧zo˩, ʈʂʰɯ˧ne˧, pi˧dʑo˩, ɖɯ˧ɭɯ˧.

Different observations could be pointed out:

1. Some of the words printed above are visible in the examples of a specific lexical entry in the dictionary but is not a lexical entry on its own. The words ə˩gi˩ and ɖɯ˧ɣ̩˧ are described in many examples but are not a specific lexical entry.

2. From the list, some words are filled in the dictionary with a different ending tone, i.e. ˩ instead of ˩˥.

3. Some words are associated with another part to form a lexical unit, such as li˩ which is combined to di˩ to form the lexical unit di˩li˩.

4. Some words contain the character ‘-’, for example the word ɖɯ˧-kʰɣ̩˧˥, which is not the case for the corresponding lexical unit in the dictionary written as ɖɯ˧ kʰɣ̩˧˥.

To support these remarks, we conducted complementary experiments that, first, retrieve the examples associated to each lexical units, and second, generate all the possible forms of the words by changing the final tone. The results are displayed in Table 5.3.

Experiment	**Word count**	**Coverage (%)**
Retrieving words from examples (2)	6,797	24.27
Tonal variability (3)	19,299	18.11
(2) + (3)	32,273	29.24

Table 5.3: Dictionary coverage and word count on the Na corpus by adding the retrieved words from the examples from the dictionary, by varying the ending tones of words from the dictionary, and by combining both in the known lexical entries list.

Table 5.4 below gives a comparison of the presence of words that are not in the dictionary with the different experiments. When the '-' is printed, the word is contained in the extracted list retrieved during the experiment.

Experiment	**Baseline (1)**	**(2)**	**(3)**	**(2) + (3)**
Examples	ə˩-gi˩	ə˩-gi˩	ə˩-gi˩	ə˩-gi˩
	ɖɯ˧-v̩˧	-	ɖɯ˧-v̩˧	-
	pi˧-zo˩	pi˧-dʑo˩	pi˧-zo˩	pi˧-dʑo˩
	ʈʂʰɯ˧ne˧	-	ʈʂʰɯ˧ne˧	-
	pi˧-dʑo˩	pi˧-dʑo˩	-	-
	ɖɯ˧-ɭɯ˧	-	ɖɯ˧-ɭɯ˧	-
	mv̩˩	-	-	-
	ʈʂʰɯ˧-dʑo˩	ʈʂʰɯ˧-dʑo˩	ʈʂʰɯ˧-dʑo˩	ʈʂʰɯ˧-dʑo˩
	ə˧ʝi˧-ʂɯ˥ʝi˩-dʑo˩	ə˧ʝi˧-ʂɯ˥ʝi˩-dʑo˩	ə˧ʝi˧-ʂɯ˥ʝi˩-dʑo˩	ə˧ʝi˧-ʂɯ˥ʝi˩-dʑo˩
	ɖɯ˧-tɑ˧˥	-	ɖɯ˧-tɑ˧˥	-
	pi˧-tsɯ˩	pi˧-tsɯ˩	pi˧-tsɯ˩	pi˧-tsɯ˩
	dʑo˩	-	-	-

TABLE 5.4: Comparison of words not in the dictionary (Baseline) according to the different experiments. The '-' confirms the presence of the word in the extracted list.

We clearly see that retrieving words from the examples of each lexical units (2) reduces the number of unknown words. The greatest improvement comes from the combination of retrieving words from examples of the lexical units and the generation of the tonal variability of each (29.24% of coverage).

Despite this improvement, more than 70% of the words from the transcriptions are not in the dictionary, which is far from maximum coverage.

Japhug corpus

Considering the Japhug corpus and all the available transcriptions, the dictionary coverage is equal to 37.58%. Here are some examples of words that are not a lexical entry in the dictionary but whose frequency is equal or higher than 20 are: tɕe, ɲɯ, smɤnmimitoʁ, tɯ, pjɤ, iɕqha, nɤkinɯ, nɤki, nɯreri, qhe, chondɤre, tɤpɤtso.

We clearly see the presence of fillers such as nɤkinɯ or nɤki. The coverage score can be explained by the absence of inflected form of verbs and nouns which constitutes a major part of the corpus.

5.2.2 Coverage on the training data

We are also interesting to know the coverage of the dictionaries on the data sets (i.e. train and test sets) corresponding to what was used during the training and the evaluation of the presented automatic speech recognition approaches (`XLSR` & `Wav2vec-U`).

Table 5.5 summarizes the number of words in each training corpus.

Language	Training size (in minutes)	Words count
Na	180	5,483
Japhug	1500	16,583

Table 5.5: Number of words in each training corpus.

Table 5.6 below describes the coverage scores from both languages.

Language	Experiments	Coverage (%)
Na	Baseline (1)	14.34
	Retrieving words from examples (2)	25.5
	Tonal variability (3)	18.95
	(2) + (3)	30.48
Japhug	-	40.33

Table 5.6: Coverage of the dictionaries on the training data from Na and Japhug corpora.

5.2.3 Coverage on the test data

We conducted the same experiments but on the test data. The number of words in the test set of the Japhug corpus and on the Na corpus is displayed in Table 5.7.

Language	Test size (in minutes)	Words count
Na	30	1,541
Japhug	20	956

Table 5.7: Number of words in each test corpus.

Language	Experiments	Coverage (%)
Na	Baseline (1)	22.32
	Retrieving words from examples (2)	38.29
	Tonal variability (3)	28.81
	(2) + (3)	43.02
Japhug	-	54.9

Table 5.8: Coverage of the dictionaries on the set data from Na and Japhug corpora.

For both corpora, the coverage of dictionaries is higher on the test sets in comparison with the train sets and when all the data are taking into account.

We have seen that the coverage scores of dictionaries on the transcriptions from the Na and the Japhug corpus are less than 50%. Specifically on the Na corpus, by retrieving the words from the examples of each lexical unit and by generating the tonal variability on the dictionary words, the coverage score slightly increase. Despite this, many words are still missing from the dictionary. The pre-processing step on the transcriptions has to be taken into account in the calculation of these scores. Indeed, it is possible that some words have been modified and thus generate a bias in the results. However, these scores still allow us to obtain an order of magnitude of the coverage of the dictionaries on the studied corpus. Feedback from a linguist (Alexis Michaud) suggests that procedures for word segmentation would need to be adapted in order to improve the results: splitting words where hyphens are found in the transcriptions.

Chapter 6

Discussion & Conclusion

This work focused on the automatic speech recognition of audio recordings, to provide transcripts for linguists to use down the line in the language documentation workflow. There were multiple challenges: (i) recognizing entities from a higher level, here words, than had been done in previous work (that 'only' recognized phonemes), which involves correctly identifying word boundaries, and (ii) dealing with a scarce-resource context, where labeled data are only available in small amounts: in this case, in a low-resource context from oral languages which lack a unique writing system and online resources (extensive transcribed audio recordings, corpora of texts with translations, etc.). Providing a system able to predict words from a speech file is a need for field linguists and the language documentation workflow as an automated approach to producing sound-aligned transcriptions will considerably reduce the workload.

The first considered approach consists in fine-tuning a pre-trained `XLSR wav2vec 2.0` model on two low-resource corpora (the Yongning Na and Japhug corpora of the Pangloss Collection, an online open archive). This approach fulfilled the task of predicting word sequences: a quantitative analysis, which consists of calculating the smallest number of modifications to be made to correct the model's prediction into the reference sentence, outperformed previous studies (based on `Kaldi` and `ESPnet` approaches). Through discussions with expert linguists, a qualitative analysis was performed. Alexis Michaud pointed out the high quality of the Na predictions, which is at the upper limit of what is possible to achieve at a "phonemic" level (recognizing sounds, not words). The same observation was made by Guillaume Jacques for Japhug. It is a major leap forward for linguistic documentation. Guillaume Jacques states: "It took me less than three minutes to correct the model's predictions". Through the discussions with NLP researchers and linguists, it appears that there is no need to further improve the observed performances on the word recognition: the problem of automatically discovering word boundaries, which constitutes the goal of a great many research projects – we can cite, for instance, the PhD thesis of Pierre Godard entitled "Unsupervised word discovery for computational language documentation" (Godard, 2019) – can be considered as solved as soon as a modest amount of annotated data is available. More broadly, obtaining a reliable transcription at the word level opens up wide perspectives. A link can be created with dictionaries, by adding lexical information. We have seen that the coverage of Na and Japhug dictionaries on the available transcriptions is still limited, yet these resources are essential for the understanding and documentation of a language. The corpora can thus be enhanced. Glosses could be easily added (text with interlinear glosses), and a translation could be offered with a tool to facilitate the production of "Pangloss-like" documents, which have not only a transcription (in sentences, broken down into words) but also the whole philological apparatus used by linguists for research purposes.

From a technical point of view, `XLSR` implementation is fast, inexpensive in computational resources, and easy to set up. All we have to do is train the neural network and use a script that runs each step from start to finish. In comparison with the previous approach, `ESPnet`, Benjamin Galliot, an expert programmer participating in the development of Elpis, took around one month to train it and get a working ASR system on the same corpora. Moreover, `XLSR` fine-tuning only requires the use of a single GPU and takes less than a day to train. This first approach is not only operable, but also used. Indeed, the pipeline to follow was explained to Séverine Guillaume, the engineer in charge of the Pangloss Collection, who will be able to transcribe new audios from the fine-tuned models of the Na and Japhug, but also to train a new model according to the resources at her disposal. Around 60 languages in the Pangloss collection contain more than 30 minutes of labeled data, and around 45 languages include at least 1 hour of labeled data (Japhug and Na are counted in).

We must not forget the many remaining challenges. The experiments were conducted only on two low-resource languages, on monolingual corpora, and with only a few speakers. From the learning curves, the trend tells us that less than an hour of labeled data is sufficient to achieve low error rates. But how well can we generalize this observation to other languages? And how do we give feedback to field linguists on the quantity and quality of data to be provided for the ASR task? It is thus necessary to keep up a two-way exchange between linguists and NLP experts in this task. In particular, a seminar presentation was given to present the experiments carried out in this work and the computer tools available for linguistic documentation for field linguists (especially PhD students, who as budding linguists stand to gain a lot from leveraging the power of state-of-the-art computational tools). Moreover, a potential limitation we could pointed out concerns the word boundary symbol. The standard convention for separating words, the pipe symbol, was used with a different meaning in the training corpus (as a tonal group boundary). Another notation would be appropriate to avoid confusion. Fine adjustments are to be made according to the characteristics of the language studied, and the characteristics of the reference transcriptions.

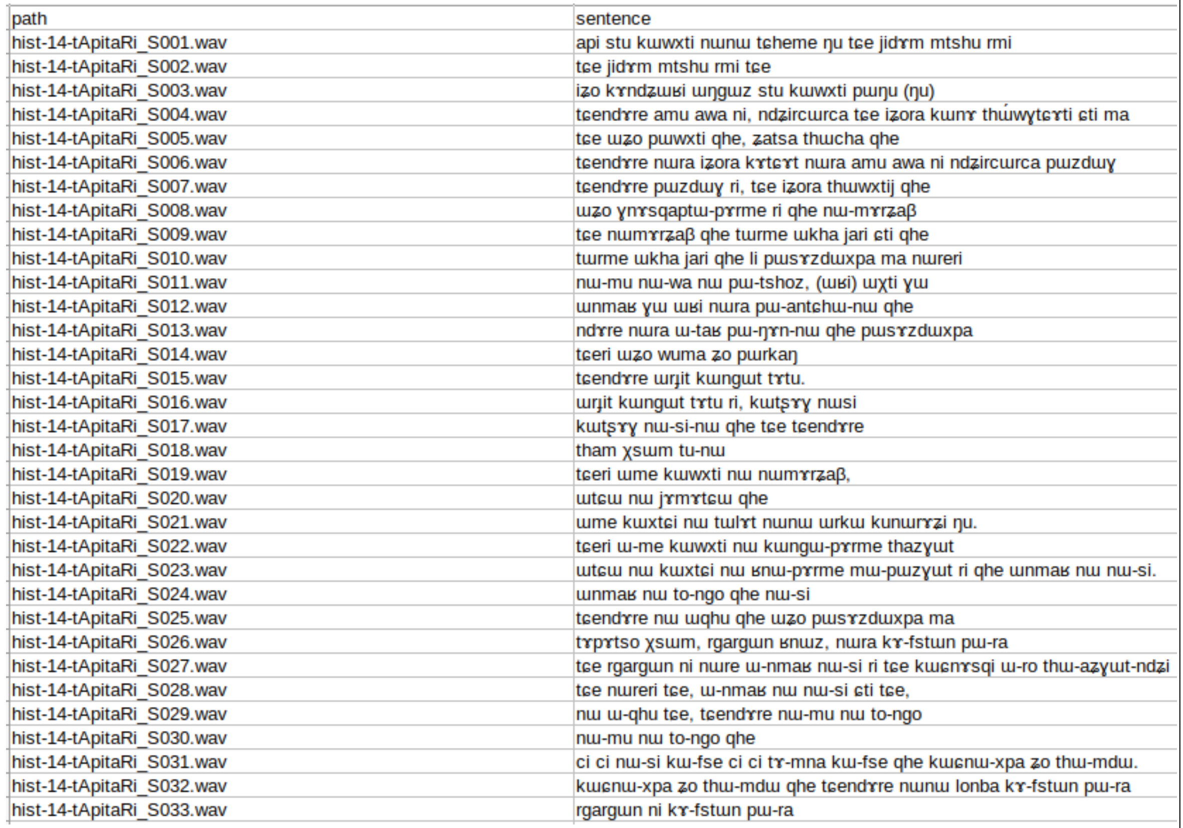
The second approach, entitled `wav2vec-U`, fully unsupervised, did not achieve the expected results on the same studied corpora. It was not possible to overcome the initial disappointing results, due to the limited literature available on the topic and the low amount of replication of this approach by other NLP researchers so far. However, even if unsupervised methods have many advantages from the point of view of technical deployment (it is easier to obtain unlabeled data, it reduces the complexity compared to supervised methods), supervised methods appear suitable for workflows in linguistic documentation, in view of the fact that corpus production constitutes a labor of love for field linguists, who standardly carry out the work of annotating their corpus themselves, producing “beautiful data”: handcrafted data sets with a high degree of precision.

This internship allowed me to learn about the latest approaches and trends applied to ASR. Through the knowledge taught throughout the NLP Master, I was able to understand and grasp the issues of each studied method. The installation of the different libraries on the server posed some difficulties at the beginning, but these were quickly solved by the help of the technical expert at the *Très Grande Infrastructure de Recherche* Huma-Num.

Appendix A

Appendix

A.1 .tsv file



path	sentence
hist-14-tApitaRi_S001.wav	api stu kɯwxti nɯnɯ tɕheme ŋu tɕe jidɤm mtshu rmi
hist-14-tApitaRi_S002.wav	tɕe jidɤm mtshu rmi tɕe
hist-14-tApitaRi_S003.wav	iʑo kɤndʑɯʁi ɯŋgɯz stu kɯwxti pɯŋu (ŋu)
hist-14-tApitaRi_S004.wav	tɕendɤre amu awa ni, ndʑircɯrca tɕe iʑora kɯnɤ thɯ́wɣtɕɤti ɕti ma
hist-14-tApitaRi_S005.wav	tɕe ɯʑo pɯwxti qhe, ʑatsa thɯcha qhe
hist-14-tApitaRi_S006.wav	tɕendɤre nɯra iʑora kɤtɕɤt nɯra amu awa ni ndʑircɯrca pɯzdɯɣ
hist-14-tApitaRi_S007.wav	tɕendɤre pɯzdɯɣ ri, tɕe iʑora thɯwxtij qhe
hist-14-tApitaRi_S008.wav	ɯʑo ɣnɤsqaptɯ-pɤrme ri qhe nɯ-mɤrʑaβ
hist-14-tApitaRi_S009.wav	tɕe nɯmɤrʑaβ qhe tɯrme ɯkha jari ɕti qhe
hist-14-tApitaRi_S010.wav	tɯrme ɯkha jari qhe li pɯsɤzdɯxpa ma nɯreri
hist-14-tApitaRi_S011.wav	nɯ-mu nɯ-wa nɯ pɯ-tshoz, (ɯʁi) ɯχti ɣɯ
hist-14-tApitaRi_S012.wav	ɯnmaʁ ɣɯ ɯʁi nɯra pɯ-antɕhɯ-nɯ qhe
hist-14-tApitaRi_S013.wav	ndɤre nɯra ɯ-taʁ pɯ-ŋɤn-nɯ qhe pɯsɤzdɯxpa
hist-14-tApitaRi_S014.wav	tɕeri ɯʑo wuma ʑo pɯrkaŋ
hist-14-tApitaRi_S015.wav	tɕendɤre ɯrʝit kɯngɯt tɤtu.
hist-14-tApitaRi_S016.wav	ɯrʝit kɯngɯt tɤtu ri, kɯtʂɤɣ nɯsi
hist-14-tApitaRi_S017.wav	kɯtʂɤɣ nɯ-si-nɯ qhe tɕe tɕendɤre
hist-14-tApitaRi_S018.wav	tham χsɯm tu-nɯ
hist-14-tApitaRi_S019.wav	tɕeri ɯme kɯwxti nɯ nɯmɤrʑaβ,
hist-14-tApitaRi_S020.wav	ɯtɕɯ nɯ jɤmɤtɕɯ qhe
hist-14-tApitaRi_S021.wav	ɯme kɯxtɕi nɯ tɯlɤt nɯnɯ ɯrkɯ kunɯrɤʑi ŋu.
hist-14-tApitaRi_S022.wav	tɕeri ɯ-me kɯwxti nɯ kɯngɯ-pɤrme thazɣɯt
hist-14-tApitaRi_S023.wav	ɯtɕɯ nɯ kɯxtɕi nɯ ʁnɯ-pɤrme mɯ-pɯzɣɯt ri qhe ɯnmaʁ nɯ nɯ-si.
hist-14-tApitaRi_S024.wav	ɯnmaʁ nɯ to-ngo qhe nɯ-si
hist-14-tApitaRi_S025.wav	tɕendɤre nɯ ɯqhu qhe ɯʑo pɯsɤzdɯxpa ma
hist-14-tApitaRi_S026.wav	tɤpɤtso χsɯm, rgargɯn ʁnɯz, nɯra kɤ-fstɯn pɯ-ra
hist-14-tApitaRi_S027.wav	tɕe rgargɯn ni nɯre ɯ-nmaʁ nɯ-si ri tɕe kɯɕnɤsqi ɯ-ro thɯ-aʑɣɯt-ndʑi
hist-14-tApitaRi_S028.wav	tɕe nɯreri tɕe, ɯ-nmaʁ nɯ nɯ-si ɕti tɕe,
hist-14-tApitaRi_S029.wav	nɯ ɯ-qhu tɕe, tɕendɤre nɯ-mu nɯ to-ngo
hist-14-tApitaRi_S030.wav	nɯ-mu nɯ to-ngo qhe
hist-14-tApitaRi_S031.wav	ci ci nɯ-si kɯ-fse ci ci tɤ-mna kɯ-fse qhe kɯɕnɯ-xpa ʑo thɯ-mdɯ.
hist-14-tApitaRi_S032.wav	kɯɕnɯ-xpa ʑo thɯ-mdɯ qhe tɕendɤre nɯnɯ lonba kɤ-fstɯn pɯ-ra
hist-14-tApitaRi_S033.wav	rgargɯn ni kɤ-fstɯn pɯ-ra

Figure A.1: Beginning of the generated .tsv file with the audio file path in the 'path' column, and the corresponding transcriptions in the 'sentence' column.

A.2 xlsr-180-na predictions

Ref:	tʰi˩˥ hĩ˧ ʈʂʰɯ˧ʂæ˧ʂæ˧hĩ˧qo˩ le˧tsʰɯ˩ɲi˩mæ˩
Hyp:	tʰi˩˥ hĩ˧ ʈʂʰɯ˧ ʂæ˧ʂæ˧hĩ˧qo˩ le˧tsʰɯ˩ɲi˩mæ˩
Ref:	hĩ˧ gi˩˥ wɤ˩˥ dzɑ˧hĩ˧ki˧ ʂe˧bi˧v̩˩tʰv̩˩hĩ˩ʈʂʰɯ˩dʑo˩ mɤ˧tsɤ˧ze˧ pi˧zo˩ tʰi˩˥ le˧wo˧tʰo˥tɕo˩ le˧tsʰɯ˩zo˩
Hyp:	hĩ˧ dʑi˧ dzɑ˧hĩ˧ki˧ ʂe˧bi˧vo˩tʰv̩˩hĩ˧ ʈʂʰɯ˧dʑo˩ mɤ˧tsɤ˧ze˩ pi˧zo˩ tʰi˩˥ le˧wo˧tʰo˥tɕo˩ le˧tsʰɯ˩zo˩
Ref:	ə˧mv̩˧ki˥ ʈʂʰɯ˧ne˧ʝi˥ ʝi˧lo˧ mɤ˧dʑɤ˩ ɲi˩ze˩mæ˩
Hyp:	ə˧mv̩˧ki˥ ʈʂʰɯ˧ne˧ʝi˥ ʝi˧lo˧ mɤ˧dʑɤ˩ ɲi˩ze˩ mæ˩
Ref:	ni˧mi˧ʈʂʰɯ˧ zo˩no˥ go˧mi˧ʈʂʰɯ˧ hĩ˧ki˧ki˩ pi˧dʑo˩ ni˧mi˧ ɲi˥ze˩mæ˩ ə˩gi˩
Hyp:	ni˧mi˧ʈʂʰɯ˧ zo˩no˥ go˧mi˧ʈʂʰɯ˧ hĩ˧ki˧ki˩ pi˧dʑo˩ ni˧mi˧ɲi˥ze˩mæ˩ ə˩gi˩
Ref:	dʑɯ˧ʈʂʰwæ˧
Hyp:	dze˧ʈʂʰwɤ˥
Ref:	njɤ˧ ə˧si˧ɳɯ˧ zo˩no˥ ə˧sɯ˩kv̩˩ no˧sɯ˩kv̩˩ zo˩no˥ le˧ʐwɤ˩pi˩dʑo˩
Hyp:	njɤ˧ ə˧si˧ zo˩no˥ ə˧sɯ˩kv̩˩ no˧sɯ˩kv̩˩ zo˩no˥ le˧ʐwɤ˩ pi˩dʑo˩
Ref:	ʈʂʰɯ˧ne˧ʝi˥ le˧tɕɯ˧ɭɯ˧ le˧tɕɯ˧ɭɯ˧ ʈʂʰɯ˧ne˧ʝi˥ le˧gv̩˩
Hyp:	ʈʂʰɯ˧ne˧i˥ le˧tɕɯ˧ɭɯ˧ le˧tɕɯ˧ɭɯ˧ ʈʂʰɯ˧ne˧ʝi˥ le˧gv̩˩
Ref:	mv̩˧dze˧ ki˧ɲi˥tsɯ˩ mv̩˩ qʰɑ˧dze˧ ki˧ɲi˥tsɯ˩ mv̩˩ ɕi˧ɭɯ˧ ki˧ɲi˥tsɯ˩ mv̩˩
Hyp:	mv̩˧dze˧ ki˧ɲi˥tsɯ˩ mv̩˩ qʰɑ˧dze˧ki˧ɲi˥tsɯ˩ mv̩˩ sɕi˧ɭɯ˧ ki˧ɲi˥tsɯ˩ mv̩˩
Ref:	njɤ˧ t njɤ˧ le˧ʂv̩˧ɖv̩˧zo˥ no˧ki˧ ʂe˧tsʰɯ˩ɲi˩
Hyp:	njɤ˧ njɤ˧ le˧ʂv̩˧ɖv̩˧zo˥ no˧ ki˧ ʂe˧tsʰɯ˧ɲi˩
Ref:	tʰi˩˥ ʈʂʰæ˧ɣɯ˧ki˩hĩ˩ dʑɤ˩˥ mɤ˧dʑo˩
Hyp:	tʰi˩˥ ʈʂʰæ˧ɣɯ˧ki˩hĩ˩ dʑɤ˩˥ mɤ˧dʑo˩
Ref:	wɤ˩˥ le˧tɕɯ˩tɕɯ˩ le˧po˧tsʰɯ˧˥ wɤ˩˥ ɬi˧di˩ tɕʰi˧kv̩˥mæ˩
Hyp:	wɤ˩˥ le˧tɕɯ˩tɕɯ˩ le˧po˧tsʰɯ˧˥ wɤ˩˥ ɬi˧di˩ tɕʰi˧kv̩˧˥ mæ˩
Ref:	kʰv kʰv̩˩mi˩ʈʂʰɯ˥dʑo˩ njɤ˧le˧gv̩˧ ʐo˩˥ ɖɯ˧mɤ˧kv̩˧tsɯ˥ mv̩˩
Hyp:	kʰv̩˧˥ no˧ kʰv̩˩mi˩ʈʂʰɯ˥dʑo˩ njɤ˧ le˧gv̩˧ ʐo˩˥ ɖɯ˧mɤ˧kv̩˧tsɯ˥ mv̩˩
Ref:	wɤ˩˥ kɯ˧ʈʂʰwɤ˩ ɖɯ˧tɕʰi˧ gv̩˩ dʑɤ˩tsʰo˧ hĩ˧ɻæ˧ki˥
Hyp:	wɤ˩˥ kɯ˧ʈʂʰwɤ˩ ɖɯ˧tɕʰi˩ gv̩˩ dʑɤ˩tsʰo˧hjæ˧ki˧
Ref:	zo˩no˥ ʈ ʈæ˧bɤ˧ʈʂʰɯ˧dʑo˩ ʈʂʰæ˧ɣɯ˧ mɤ˧dʑo˧ɲi˥ ʈʂʰo˧lɑ˧kv̩˥
Hyp:	zo˩no˥ ʈ ʈæ˧bɤ˧ʈʂʰɯ˧dʑo˩ ʈʂʰæ˧ɣɯ˧ mɤ˧dʑo˧ɲi˥ ʈʂʰo˧lɑ˧kv̩˥
Ref:	tʰi˩˥ ə˧mi˧ le˧ʂɯ˧ɲi˥ho˩ze˩wɤ˩ æ˧ʂæ˧tɑ˩mv̩˩dʑo˩
Hyp:	tʰi˩˥ ə˧mi˧ le˧ʂɯ˧ɲi˥ho˩ze˩wɤ˩˥ æ˧ʂæ˧tɑ˩mv̩˩dʑo˩
Ref:	əəə... ʐv̩˧tsʰi˩gv̩˩hã˩ ɖɯ˩kʰv̩˩ mmm... lv̩˩ kʰɯ˩zo˩kv̩˥pi˩
Hyp:	əəə... ʐv̩˧tsʰi˩gv̩˩hã˩ ɖɯ˧kʰv̩˩ lv̩˧kʰɯ˩zo˩kv̩˩ pi˩
Ref:	hæ̃˧pɤ˧ tʰi˧mv̩˧kʰɯ˧se˥ dʑo˩ tʰi˩˥ dʑɯ˧ki˥ tʰi˧ki˩kʰɯ˩
Hyp:	hæ̃˧pɤ˧ tʰi˧mv̩˧kʰɯ˧se˥dʑo˩ tʰi˩˥ ɖʑɯ˧ki˥ tʰi˧ki˩kʰɯ˩
Ref:	dʑɤ˩tsʰo˧dʑo˧ ə˧ʝi˧ʂɯ˥ʝi˩ mmm... dʑo˩ kɤ˧kɤ˩ tʰi˧dzi˩dʑo˩
Hyp:	dʑɤ˩tsʰɯ˧dʑo˥ ə˧ʝi˧ʂɯ˥ʝi˩ ... dʑo˩ kɤ˧kɤ˩ tʰi˧dzi˩dʑo˩

A.3 xlsr-jya-600 predictions

Ref:	rtazga nɯ pjɯ́wɣta ŋu
Hyp:	ɤzga nɯ pjɯ́wɣta ŋu
Ref:	tɕe ma nɤkinɯ ɯru nɯnɯ fsapaʁndza pe
Hyp:	tɕe ma nɤkinɯ ɯru nɯnɯ fsapaʁndza pe
Ref:	pɯnɯcinɯ ʑo tɕe nɯ kunaχthɤβ tɕe
Hyp:	pɯnɯcinɯ ʑo tɕe tɕe nɯ kunaχthɤβ tɕe
Ref:	tɤrɤku ra ɣɯmɯrki ra ma me
Hyp:	tɤrɤku ra ɣɯmɯrki ra ma me
Ref:	tɕe li kɯnɯkho joɕe
Hyp:	tɕe li kɯnɯkho joɕe
Ref:	iɕqha tɤtɕɯ tɯlɤt nɯnɯ nɤkinɯ kɯmaʁ li kɯrɤtsɣe ra nɯrca joɕe tɕe
Hyp:	ɕqha tɤtɕɯ tɯlɤt nɯnɯ nɤkinɯ kɯmaʁ li kɯrɤtsɣe ra nɯrca joɕe tɕe
Ref:	ma nɯnɯ zrɯɣ nɯ li saʁnɤt
Hyp:	ma nɯnɯ zrɯɣ nɯ li saʁnɤt
Ref:	nɯnɯ nɯɬoʁ ʑo tɕe tɕe tukɯnɯna pjɤjɤɣ ɲɯŋu
Hyp:	nɯnɯ nɯɬoʁ ʑo tɕe tɕe tukɯnɯna pjɤjɤɣ ɲɯŋu
Ref:	tuʁndi pjɯsat nɯra ɲɯŋgrɤl tɕe tɕe
Hyp:	tuʁndi pjɯsat nɯra ɲɯŋgrɤl tɕe tɕe
Ref:	si nɯ apɯndzur qhe
Hyp:	si nɯ apɯmdzu qhe ki
Ref:	longtou ɯpɤrthɤβ ri nɯro ki ʑo tustunɯ ra
Hyp:	noŋthɯɣ pɤrthɤβ ri nɯro ki ʑo tustunɯ ra
Ref:	sɯŋgɯ ri kurɤʑi ɕti tɕe
Hyp:	sɯŋgɯ ri kurɤʑi ɕti tɕe
Ref:	χsɯɣjɤn tɤatɕhɯza tɕe tɕendɤre nɯ spjaŋkɯ ɲɯβzea ŋu tɕe
Hyp:	χsɯɣjɤn tatɕhɯza tɕe tɕendɤre spjaŋkɯ ɲɯβzea ŋu tɕe
Ref:	kɤtaʁ kɤ́wɣthɯ tɕe tɕendɤre
Hyp:	kɤtaʁ kɤ́wɣthɯ tɕe tɕendɤre
Ref:	tɕendɤre ʑmbrɯ ɯtaʁ toɕe tɕe nɯ ɕɯŋgɯ ɯʑo ɯsɤtɕha nɯ tɕu nɤki yalishan kɤti nɯnɯtɕu
Hyp:	tɕendɤre ʑmbrɯ ɯtaʁ toɕe tɕe nɯ ɕɯŋgɯ ɯʑo ɯsɤtɕha nɯtɕu nɤki yalishan kɤti nɯnɯtɕu
Ref:	ɲɤlɤt tɕe tɕendɤre iɕqha nɯ
Hyp:	ɲɤlɤt tɕe tɕendɤre iɕqha nɯ
Ref:	ɯku ɯtaʁ kutɯtɯɣ ʑo tɕe tɕendɤre
Hyp:	ra ɲɤsɯso ri ɯku ɯtaʁ kutɯtɯɣ ʑo tɕe tɕendɤre
Ref:	nɯ ɯtaʁ nɯtɕu iɕqha pjɯrɯ chonɤ rɯnbotɕhi nɯra kundzoʁ ɲɯŋgrɤl tɕe tɕe iɕqha
Hyp:	nɯ ɯtaʁ nɯtɕu iɕqha pjɯrɯ chonɤ rɯnbotɕhi nɯra kundzoʁ ɲɯŋgrɤl tɕe tɕeiɕqha

A.4 Predictions of an unseen Na speech file

Ref:	ə˧ʝi˧ʂɯ˥ʝi˩dʑo˩ ə˩gi˩ zo˩no˥ hĩ˧ ʈʂʰɯ˧dʑo˩ əəə... ɖʐwæ˧ɖʐwæ˥hwɤ˩hwɤ˩ mmm... pi˧dʑo˩ ʈʂɯ˧ʈʂɯ˩ ɻæ˧ɻæ˧ tʰv̩˧ ɖʐwæ˧ɖʐwæ˥hwɤ˩hwɤ˩ tʰv̩˩ pi˧kv̩˩ mæ˩
Hyp:	ə˧ʝi˧ʂɯ˥ʝi˩dʑo˩ ə˩gi˩ zo˩no˥ hĩ˧ʈʂʰɯ˧dʑo˩ əə... ɖʐwæ˧ ɖʐwæ˥hɤ˩hoɤ˩ mə... pi˧dʑo˩ ʈʂɯ˧ʈʂɯ˩ ɻæ˧ɻæ˧tʰv̩˧ ɖʐwæ˧ ɖʐwæ˥hwɤ˩hɤ˩ tʰv̩˩ pi˧kv̩˩mæ˩
Ref:	tʰi˩˥ ə˧ʝi˧ʂɯ˥ʝi˩dʑo˩ tʰi˩˥ zo˩no˥ mmm... sɯ˧pʰi˧ki˧ qwɤ˧qwɤ˩ bi˩kv̩˩ mæ˩
Hyp:	tʰi˩˥ ə˧ʝi˧ʂɯ˥ʝi˩dʑo˩ tʰi˩˥ zo˩no˥ mm... sɯ˧pʰi˧ki˧ qwɤ˧qwɤ˩bi˩kv̩˩mæ˩
Ref:	sɯ˧pʰi˧ki˧ le˧qwɤ˧qwɤ˩se˩dʑo˩ tʰi˩˥ sɯ˧pʰi˧ no˧ɻ̩˩ dʑɤ˩ pi˧ mɤ˧ʁo˧ njɤ˧ɻ̩˩ dʑɤ˩ pi˧ mɤ˧ʁo˧ ɲi˧ʑi˩ do˧bv̩˧ lɑ˧kv̩˥ze˩ mæ˩
Hyp:	sɯ˧pʰi˧ki˧ le˧qwɤ˥qwɤ˩se˩dʑo˩ tʰi˩˥ sɯ˧pʰi˧ no˧ɻ̩˩ dʑɤ˩ pi˧ mɤ˧ʁo˧ njɤ˧ɻ̩˩ dʑɤ˩ pi˧ mɤ˧ʁo˧ ɲi˧ʑi˩ do˧bv̩˧ lɑ˩kv̩˩ze˩mæ˩
Ref:	tʰi˩˥ do˧bv̩˧ lɑ˧˥ dʑo˩ tʰi˩˥ wɤ˩˥ le˧ɖʐwæ˧ɖʐwæ˥ le˧ɖʐwæ˧ɖʐwæ˥ le˧ɖʐwæ˧ɖʐwæ˥ dʑo˩ tʰi˩˥ mɤ˧tsɤ˧ mɤ˧ho˧ho˥...
Hyp:	tʰi˩˥ do˧bv̩˧ lɑ˧˥ dʑo˩ tʰi˩˥ wɤ˩˥ le˧ɖʐwæ˧ɖʐæ˥ le˧ɖʐwæ˧ʐwɤ˥ le˧ʐwæ˧ʐwæ˥dʑo˩ tʰi˩˥ mɤ˧tsɤ˧ mv̩˧ho˧ho˧
Ref:	ʈʰææ̃˧ ho˩ho˥ mɤ˧tʰɑ˧ ho˧ho˥ mɤ˧tʰɑ˩dʑo˩ tʰi˩˥ əəə... sɯ˧pʰi˧ɳɯ˧ no˧sɯ˩kv̩˩ tʰɑ˧ɖʐwæ˧ ɖʐwæ˥ze˩ ɖæ˩mi˧qo˧ kɤ˧ʈʂɯ˩ ʝi˩ hõ˩ pi˧kv̩˩tsɯ˩ mv̩˩
Hyp:	ʈʰææ̃˧ ho˩ho˥ mɤ˧tʰɑ˧ ho˧ho˥ mɤ˧tʰɑ˩dʑo˩ tʰi˩˥ əəə... sɯ˧pʰi˧ɳɯ˧ no˧sɯ˩kv̩˩ tʰɑ˧ɖʐwæ˧ ɖʐwæ˥ze˩ ɖæ˩mi˧qo˧ kɤ˧ʈʂe˩ hĩ˩hõ˩ pi˧kv̩˩tsɯ˩ mv̩˩

A.5 Predictions of an unseen Japhug speech file

Ref:	tɕe kɯɕɯŋgɯ tɕe iɕqha mingchao ɯraŋ nɯtɕu pjɤŋu tɕendɤre iɕqha nɤki yanguo kɤti rɟɤlkhɤβ ɣɯ nɯrɟɤlpu nɯ kɯ iɕqha nɯ
Hyp:	tɕe kɯɕɯŋgɯ tɕe iɕqha minchaouɯraŋg nɯ tɕu pjɤŋu tɕendɤre iɕqha nɤki yanguo kɤti rɟɤlkhɤβ ɣɯ nɯrɟɤlpu nɯ kɯ iɕqha nɯ
Ref:	iɕqha nɯ ɯftsa nɯnɯ rɟɤlpu lusɯndɤm pjɤsɯso
Hyp:	iɕqha nɯ ɯftsa nɯnɯ rɟɤlpu lusɯndɤm pjɤsɯso
Ref:	tɕe nɯ rɟɤlpu lusɯndɤm pjɤsɯso tɕe tɕendɤre nɤkinɯ sɤtɕha ra tosɤtʂoʁloʁnɯ ʑo ɕti tɕe tɕendɤre iɕqha nɯ
Hyp:	tɕe nɯ rɟɤlpu lusɯndɤm pjɤsɯso tɕe tɕendɤre nɤkinɯ sɤtɕha ra tosɤtʂoʁloʁnɯ ʑo ɕti tɕe tɕendɤre iɕqha nɯ
Ref:	shandong nɯtɕu ɯrmi zhangxiaobing kɯrmi ci tɯtsɣe ɯkɯβzu ci pjɤtu tɤtɕɯ
Hyp:	shandong nɯtɕu ɯrmi zhangxiaobing kɯrmi ci tɯtsɣe ɯkɯβzu ci pjɤtu tɤtɕɯ

Ref: tɕendɤre ɯrʑaβ nɯ ɯskhrɯ mɯɲɤβdi χsɯsla ma mɯtoβzu ri tɕendɤre ɯpɕi joɬoʁndʑi ɲɤphɣondʑi pjɤra matɕi
Hyp: tɕendɤre ɯrʑaβ nɯ ɯskhrɯ mɯɲɤβdi χsɯsla ma mɯtoβzu ri tɕendɤre ɯpɕi joɬoʁndʑi ɲɤphɣondʑi pjɤra matɕi

---

Ref: sɤtɕha ra pjɤkɤtʂoʁloʁci qhe tɕe nɯra tɕetha kɯsɤɣʑi ra pɯme ma ɲɤsɯsondʑi qhe tɕe nɯ jophɣondʑi
Hyp: sɤtɕha ra pjɤkɤtʂoʁloʁci qhe tɕe nɯra tɕetha kɯsɤɣʑi ra pɯme ma ɲɤsɯsondʑi qhe tɕe nɯ jophɣondʑi

---

Ref: tɕeri tʂu jɤrindʑi tɕe tɕendɤre kɯdɤn mɯtotsundʑi ma tɕendɤre ʁmaʁ ra pjɤdɤn qhe nɯra tɯrme kɯnɤphɯphyo nɯra qhe tówɣsɤtʂoʁloʁnɯ ʑo ɕti qhe ɯrʑaβ ɲɤ́wɣsɯβde
Hyp: tɕeri tʂu jɤrindʑi tɕe tɕendɤre kɯdɤn mɯtotsundʑi ma tɕendɤre ʁmaʁ ra pjɤdɤn qhe nɯra tɯrme kɯnɤphɯphyo nɯra qhe tówɣsɤtʂoʁloʁnɯ ʑo ɕti qhe ɯrʑaβ ɲɤ́wɣsɯβde

---

Ref: qhe ʁzɤmi ni ʑaka jonɯɕendʑi tɕe tɤtɕɯ nɯ kɯ ɯrʑaβ ɯskhrɯm mɤkɯβdi nɯ pjɤnɯzdɯɣ tɕe aʁɤndɯndɤt ʑo ɲɤɕar ri maka mɯpjɤmto
Hyp: qhe ʁzɤmi ni ʑaka jonɯɕendʑi tɕe tɤtɕɯ nɯ kɯ ɯrʑaβ ɯskhrɯm mɤkɯβdi nɯ pjɤnɯzdɯɣ tɕe aʁɤndɯndɤt ʑo ɲɤɕar ri maka mɯpjɤmto

---

Ref: mɯpjɤmto qhe tɕendɤre nɤkinɯ iɕqha nɯ ɯrʑaβ nɯ ma kɯ ɯʁjɯβ cinɤ kɤmto mɯpjɤcha qhe wuma ʑo pjɤnɯzdɯɣ
Hyp: mɯpjɤmto qhe tɕendɤre nɤkinɯ iɕqha nɯ ɯrʑaβ nɯ ma kɯ ɯʁjɯβ ci nɤ kɤmto mɯpjɤcha qhe wuma ʑo pjɤnɯzdɯɣ

Ref: tɕe pjɤnɯzdɯɣ ri kɤpa pjɤme qhe joɕe qhe tɕendɤre tɯsɲi qhe nɤkinɯ iɕqha nɯ
Hyp: tɕe pjɤnɯzdɯɣ ri kɤpa pjɤme qhe joɕe qhe tɕendɤre tɯsɲi qhe nɤkinɯ iɕqha nɯ

Ref: li iɕqha kɯnɤphɯphyo ra nɯrca nɯtɕu tɕe joɣi tɕe shandong pjɤrɤʑi ɕti ri nɯra kɤɕar ntsɯ kɯ henan jozɣɯt
Hyp: li iɕqha kɯnɤphɯphyo ra nɯrca nɯtɕu tɕe joɣi tɕe shandong pjɤrɤʑi ɕti ri nɯra kɤɕar nts ke huolan ɲozɣɯt

---

Ref: henan nɯnɯ jozɣɯt qhe tɕendɤre nɯtɕu qhe tɕe nɤkinɯ tɯtsɣe ɯkɯβzu tsuku ɲɤkɤtɯɣci qhe nɯ ɕɯŋgɯ ɯkɤnɯfse ci ɲɤkɤtɯɣci
Hyp: huolan nɯnɯ jozɣɯt qhe tɕendɤre nɯtɕu qhe tɕe nɤkinɯ tɯtsɣe ɯkɯβzu tsuku ɲɤkɤtɯɣci qhe nɯ ɕɯŋgɯ ɯkɤnɯfse ci ɲɤkɤtɯɣci

---

Ref: tɕe nɯ kɯ ɲɤ́wɣnɯkhɤda qhe manɯtɯnɯzdɯɣ tɕe toti tɕe iɕqha tɯtsɣe sɤβzu ɣɯ ɯʁdɤnba ra ɲɤ́wɣznɤŋgɯ qhe
Hyp: tɕe nɯ kɯ ɲɤ́wɣnɯkhɤda qhe manɯtɯnɯzdɯɣ tɕe toti tɕe iɕqha tɯtsɣe sɤβzu ɣɯ ʁdɤnba ra ɲɤ́wɣznɤŋgɯ qhe

---

Ref: tɕendɤre nɯra nɤʑo tɯtsɣe tɤβze toti
Hyp: tɕendɤre nɯra nɤʑo tɯtsɣe tɤβze toti

Ref: tɕendɤre nɯ ɯqhu tɕe tɕendɤre kɯki zhangxiaobing nɯnɯ henan nɯtɕu lorɤʑi qhe
Hyp: tɕendɤre nɯ ɯqhu tɕe tɕendɤre kɯki zhangxiaobin nɯnɯ huolan nɯtɕu lorɤʑi qhe

---

Bibliography

Adams, Oliver et al. (2018). “Evaluating phonemic transcription of low-resource tonal languages for language documentation”. In: *LREC 2018 (Language Resources and Evaluation Conference)*, pp. 3356–3365.

Adams, Oliver et al. (2020). “User-friendly automatic transcription of low-resource languages: Plugging ESPnet into Elpis”. In: *arXiv preprint arXiv:2101.03027*.

Akiba, Takuya et al. (2019). “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631.

Amodei, Dario et al. (2016). “Deep speech 2: End-to-end speech recognition in english and mandarin”. In: *International conference on machine learning*. PMLR, pp. 173–182.

Ardila, Rosana et al. (2019). “Common voice: A massively-multilingual speech corpus”. In: *arXiv preprint arXiv:1912.06670*.

Artetxe, Mikel et al. (2017). “Unsupervised neural machine translation”. In: *arXiv preprint arXiv:1710.11041*.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). “Layer normalization”. In: *arXiv preprint arXiv:1607.06450*.

Baevski, Alexei, Michael Auli, and Abdelrahman Mohamed (2019). “Effectiveness of self-supervised pre-training for speech recognition”. In: *arXiv preprint arXiv:1911.03912*.

Baevski, Alexei, Steffen Schneider, and Michael Auli (2019). “vq-wav2vec: Self-supervised learning of discrete speech representations”. In: *arXiv preprint arXiv:1910.05453*.

Baevski, Alexei et al. (2020). “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *arXiv preprint arXiv:2006.11477*.

Baevski, Alexei et al. (2021). “Unsupervised Speech Recognition”. In: *arXiv:2105.11084*.

Berment, Vincent (2004). “Méthodes pour informatiser les langues et les groupes de langues «peu dotées»”. PhD thesis. Université Joseph-Fourier-Grenoble I.

Besacier, Laurent et al. (2014). “Automatic speech recognition for under-resourced languages: A survey”. In: *Speech Communication* 56, pp. 85–100. ISSN: 0167-6393. DOI: `https://doi.org/10.1016/j.specom.2013.07.008`. URL: `https://www.sciencedirect.com/science/article/pii/S0167639313000988`.

Chen, Hanting et al. (2021). “Pre-trained image processing transformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12299–12310.

Chen, Kuan-Yu et al. (2019). “Completely Unsupervised Phoneme Recognition by a Generative Adversarial Network Harmonized with Iteratively Refined Hidden Markov Models.” In: *INTERSPEECH*, pp. 1856–1860.

Chen, Ting et al. (2020). “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR, pp. 1597–1607.

Chung, Yu-An and James Glass (2018). “Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech”. In: *arXiv preprint arXiv:1803.08976*.

Chung, Yu-An et al. (2018). “Unsupervised cross-modal alignment of speech and text embedding spaces”. In: *arXiv preprint arXiv:1805.07467*.

Chung, Yu-An et al. (2019). “An unsupervised autoregressive model for speech representation learning”. In: *arXiv preprint arXiv:1904.03240*.

CNRS, Lacito (2021a). *Collection Pangloss*. https://pangloss.cnrs.fr/.

— (2021b). *Japhug*. https://pangloss.cnrs.fr/corpus/Japhug.

— (2021c). *Na de Yongning*. https://pangloss.cnrs.fr/corpus/Yongning_Na.

Conneau, Alexis et al. (2017). "Word translation without parallel data". In: *arXiv preprint arXiv:1710.04087*.

Conneau, Alexis et al. (2019). "Unsupervised cross-lingual representation learning at scale". In: *arXiv preprint arXiv:1911.02116*.

Conneau, Alexis et al. (2020). "Unsupervised cross-lingual representation learning for speech recognition". In: *arXiv preprint arXiv:2006.13979*.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dieleman, Sander, Aäron van den Oord, and Karen Simonyan (2018). "The challenge of realistic music generation: modelling raw audio at scale". In: *arXiv preprint arXiv:1806.10474*.

Dong, Linhao, Shuang Xu, and Bo Xu (2018). "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5884–5888.

Esch, Daan van, Ben Foley, and Nay San (2019). "Future directions in technological support for language documentation". In: *Proceedings of the Workshop on Computational Methods for Endangered Languages*. Vol. 1.

Foley, Ben et al. (2018). "Building speech recognition systems for language documentation: the CoEDL Endangered Language Pipeline and Inference System (ELPIS)". In:

Gales, Mark JF et al. (2014). "Speech recognition and keyword spotting for low-resource languages: BABEL project research at CUED". In: *Fourth International Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU-2014)*. International Speech Communication Association (ISCA), pp. 16–23.

Ghahremani, Pegah et al. (2017). "Investigation of transfer learning for ASR using LF-MMI trained neural networks". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, pp. 279–286.

Godard, Pierre (2019). "Unsupervised word discovery for computational language documentation". PhD thesis. Université Paris-Saclay (ComUE).

Goodfellow, Ian et al. (2020). "Generative adversarial networks". In: *Communications of the ACM* 63.11, pp. 139–144.

Google (2021a). *Google maps of Barkam Xian, Aba Tibetan and Qiang Autonomous Prefecture, Sichuan, China*. https://www.google.fr/maps.

— (2021b). *Speech-to-text*. https://cloud.google.com/speech-to-text. Accessed: 2021-09-26.

Graves, Alex et al. (2006). "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376.

Hannun, Awni (2017). "Sequence modeling with ctc". In: *Distill* 2.11, e8.

Hany, John and Greg Walters (2019). *Hands-On Generative Adversarial Networks with PyTorch 1. x: Implement next-generation neural networks to build powerful GAN models using Python*. Packt Publishing Ltd.

He, Kaiming et al. (2020). "Momentum contrast for unsupervised visual representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738.

Heafield, Kenneth (July 2011). "KenLM: Faster and Smaller Language Model Queries". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 187–197. URL: https://www.aclweb.org/anthology/W11-2123.

Heafield, Kenneth et al. (2013). “Scalable modified Kneser-Ney language model estimation”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 690–696.

Heigold, Georg et al. (2013). “Multilingual acoustic models using distributed deep neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 8619–8623.

Hendrycks, Dan and Kevin Gimpel (2016). “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415*.

Hrinchuk, Oleksii, Mariya Popova, and Boris Ginsburg (2020). “Correction of automatic speech recognition with transformer sequence-to-sequence model”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7074–7078.

Huang, Jui-Ting et al. (2013). “Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 7304–7308.

Jaadi, Zakaria (2021). *A Step-by-Step Explanation of Principal Component Analysis (PCA)*. https://builtin.com/data-science/step-step-explanation-principal-component-analysis.

Jacques, G. (2021). *A Grammar of Japhug*. Comprehensive Grammar Library. Language Science Press. ISBN: 9783985540013.

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

— (2015). *”Mon travail sur le japhug (1),” Panchronica,* https://panchr.hypotheses.org/264(ISSN2494-775X).

— (2016). *Dictionnaire Japhug-chinois-français Version 1.1.*

Jang, Eric, Shixiang Gu, and Ben Poole (2016). “Categorical reparameterization with gumbel-softmax”. In: *arXiv preprint arXiv:1611.01144*.

Jegou, Herve, Matthijs Douze, and Cordelia Schmid (2010). “Product quantization for nearest neighbor search”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.1, pp. 117–128.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2019). “Billion-scale similarity search with gpus”. In: *IEEE Transactions on Big Data*.

Joshi, Vikas et al. (2020). “Transfer learning approaches for streaming end-to-end speech recognition system”. In: *arXiv preprint arXiv:2008.05086*.

Karita, Shigeki et al. (2019). “A comparative study on transformer vs rnn in speech applications”. In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, pp. 449–456.

Krauwer, Steven (2003). “The basic language resource kit (BLARK) as the first milestone for the language resources roadmap”. In: *Proceedings of SPECOM*. Vol. 2003, pp. 8–15.

Lample, Guillaume and Alexis Conneau (2019). “Cross-lingual language model pretraining”. In: *arXiv preprint arXiv:1901.07291*.

Lample, Guillaume et al. (2017). “Unsupervised machine translation using monolingual corpora only”. In: *arXiv preprint arXiv:1711.00043*.

Lewis M. Paul, Gary F. Simons and Charles D. Fennig (eds.) (2016). *Languages of China: An Ethnologue country report*. Dallas, TX: SIL International. URL: http://www.ethnologue.com/.

Littell, Patrick et al. (2018). “Indigenous language technologies in Canada: Assessment, challenges, and successes”. In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2620–2632.

Liu, Da-Rong et al. (2018). “Completely unsupervised phoneme recognition by adversarially learning mapping relationships from audio embeddings”. In: *arXiv:1804.00316*.

Liu, Yinhan et al. (2019). “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692*.

Liu, Yu, Jianshu Chen, and Li Deng (2017). “Unsupervised sequence classification using sequential output statistics”. In: *arXiv preprint arXiv:1702.07817*.

Michaud, Alexis (2017). *Tone in Yongning Na: Lexical tones and morphotonology*. Language Science Press.

Michaud, Alexis et al. (2019). “Phonetic lessons from automatic phonemic transcription: preliminary reflections on Na (Sino-Tibetan) and Tsuut’ ina (Dene) data”. In: *ICPhS XIX (19th International Congress of Phonetic Sciences)*.

Mikolov, Tomas et al. (2013a). “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781*.

Morris, Andrew, Viktoria Maier, and Phil Green (Jan. 2004). “From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition.” In:

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748*.

Palaz, Dimitri, Ronan Collobert, et al. (2015). *Analysis of cnn-based speech recognition system using raw speech as input*. Tech. rep. Idiap.

Panayotov, Vassil et al. (2015). “Librispeech: an asr corpus based on public domain audio books”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5206–5210.

Park, Daniel S et al. (2020). “Improved noisy student training for automatic speech recognition”. In: *arXiv preprint arXiv:2005.09629*.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Pratap, Vineel et al. (2020). “MLS: A Large-Scale Multilingual Dataset for Speech Research”. In: *arXiv preprint arXiv:2012.03411*.

Riviere, Morgane et al. (2020). “Unsupervised pretraining transfers well across languages”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7414–7418.

Ruder, Sebastian (2021). *Language modeling*. `http://nlpprogress.com/english/language_modeling.html`.

Schneider, Steffen et al. (2019). “wav2vec: Unsupervised pre-training for speech recognition”. In: *arXiv preprint arXiv:1904.05862*.

Szymański, Piotr et al. (2020). “WER we are and WER we think we are”. In: *arXiv preprint arXiv:2010.03432*.

Tan, Zheng-Hua, Najim Dehak, et al. (2020). “rVAD: An unsupervised segment-based robust voice activity detection method”. In: *Computer speech & language* 59, pp. 1–21.

Thieberger, Nick (2017). “LD&C possibilities for the next decade”. In:

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: pp. 5998–6008.

Veselỳ, Karel et al. (2012). “The language-independent bottleneck features”. In: *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 336–341.

Wang, Yu-Hsuan, Cheng-Tao Chung, and Hung-yi Lee (2017). “Gate activation signal analysis for gated recurrent neural networks and its correlation with phoneme boundaries”. In: *arXiv preprint arXiv:1703.07588*.

Watanabe, Shinji et al. (2018). “Espnet: End-to-end speech processing toolkit”. In: *arXiv preprint arXiv:1804.00015*.

Wisniewski, Guillaume, Alexis Michaud, and Séverine Guillaume (2020). “Phonemic transcription of low-resource languages: To what extent can preprocessing be automated?”

In: *1st Joint SLTU (Spoken Language Technologies for Under-resourced languages) and CCURL (Collaboration and Computing for Under-Resourced Languages) Workshop*. European Language Resources Association (ELRA), pp. 306–315.

Xu, Qiantong et al. (2020). “Iterative pseudo-labeling for speech recognition”. In: *arXiv preprint arXiv:2005.09267*.

Yeh, Chih-Kuan et al. (2018). “Unsupervised speech recognition via segmental empirical output distribution matching”. In: *arXiv preprint arXiv:1812.09323*.

Zhang, Zi-Qiang et al. (2021). “XLST: Cross-lingual Self-training to Learn Multilingual Representation for Low Resource Speech Recognition”. In: *arXiv preprint arXiv:2103.08207*.