LABORATOIRE INFORMATIQUE D'AVIGNON

INTERNSHIP

Distributed learning for speech recognition in a context of privacy protection

Author:
Adueni Adjoba Eunice AKANI

Supervisor:
Dr. Yannick ESTEVE

Jury:
Miguel COUCEIRO
Kamel SMAILI
Romain SÉRIZEL

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

in the

Université de Lorraine
Institut des Sciences du Digital Management et Cognition
IDMC

February 24, 2020 to August 16, 2020

Declaration of Authorship

I, Adueni Adjoba Eunice AKANI, declare that this thesis titled, "Distributed learning for speech recognition in a context of privacy protection" 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: 23 / 08 / 2020

LABORATOIRE INFORMATIQUE D'AVIGNON

Abstract

Université de Lorraine
Institut des Sciences du Digital Management et Cognition
IDMC

Masters of Natural Language Processing

Distributed learning for speech recognition in a context of privacy protection

by Adueni Adjoba Eunice AKANI

In view of the frequent use of automatic speech recognition systems in many devices and the fact that the system is trained on a large amount of users data which may contain private information; this internship aims to bridge the gap between the improvement of ASR system and the protection of sensitive users information. Thus, many acoustics models were trained on users data and some information such as weight matrix were extracted from the acoustic models. Using a clustering method, the data of the same speaker were grouped together. Moreover, the clustering revealed a gender grouping. Thus, analysis were done on speaker identification and gender grouping. The results obtained show that at the first layers of the models, it is possible to have meta information from the speech such as gender which is not the case for the higher layers. With regards to speaker identification, the best result was obtained from the first layer. Furthermore, the results depend on the number of epochs on which the model is trained. This work gave first results and opens other lines of research.

Acknowledgements

This work is the conclusive work for the Master's degree in Natural Language Processing at IDMC (Institut des Sciences du Digital Management et Cognition) Université de Lorraine. Writing this thesis was a great challenge and was hugely rewarding at the same time because I learnt a lot about my field of activities.

I thank my Lord who gave me strength to fulfill my internship and those even when it was difficult.

This internship wouldn't have been possible without the full support of my internship tutor Yannick ESTEVE. I am especially indebted to him for the opportunities he gave me to work on this thematic.

I express my deep sense of gratitude to the LIA that welcomed me, to the head of the Master, to my godfather, to the jury of this report and the entire staffs of the IDMC.

I would like to show my appreciations to those who help me during this period; thank all of you for the time you took.

Finally, I wish to thank my parents, my big sister and my big brother for their encouragement and belief in me.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ASR	**A**utomatic **S**peech **R**ecognition
BLRI	**B**rain and **L**anguage **R**esearch **I**nstitute
CERI	**C**entre d'**E**nseignement et de **R**ercherche **I**nformatique
CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
DNN	**D**eep **N**eural **N**etwork
fMLLR	**F**eature space **M**aximum **L**ikelihood **L**inear **R**egression
GMM	**G**aussian **M**ixture **M**odelling
GPU	**G**raphics **P**rocessing **U**nit
GRU	**G**ated **R**ecurrent **U**nit
HMM	**H**idden **M**arkov **P**rogramming
ICASSP	**I**nternational **C**onference on **A**coustics, **S**peech, and **S**ignal **P**rocessing
ICST	**I**nformation and **C**ommunication **S**cience and **T**echnology
LIP	**L**inear **I**nteger **P**rogramming
LDA	**L**atent **D**irichlet **A**llocation
LPL	**L**aboratoire **P**arole et **L**angage
LSA	**L**atent **S**emantic **A**lgorithm
MFC	**M**el **F**requency **C**cepstrum
MFCC	**M**el-**f**requency **C**epstral **C**oefficients
MLP	**M**ulti**l**ayer **P**erceptron
OS	**O**perating **S**ystem
RNN	**R**ecurrent **N**eural **N**etwork
UAPV	**U**niversity of **A**vignon and **P**ays de **V**aucluse
WER	**W**ord **E**rror **R**ate
GPGPU	**G**eneral-purpose **P**rocessing on **G**raphics **P**rocessing **U**nits

Dedicated to Family…

Chapter 1

Introduction

Since the evolution of computer science in particular with the breakthrough of artificial intelligence (AI) and machine learning, a lot of users' devices have been contained advanced technologies. One of the technologies included in daily life is automatic speech recognition system. Indeed, through it, users can ask their device using vocal command to execute tasks such as Google researches, alarm activation, music etc. This system of vocal command is very useful when it quickly recognizes the user's voice. It was trained on a lot of data to improve efficiency and performance. These data have been taken from users devices. The data may contain sensitive information about the user and this could create a big problem of privacy. Indeed, users have no guarantee about the privacy of their data neither how they will be used.
Thus, it is important to make a bridge between improving automatic speech recognition system and protecting the private information of users. This internship worked on this thematic. Being part of the national project that aims to protect sensitive information while enhancing speech recognition, the internship is focused on distributed learning for speech recognition that intend to execute same algorithm on different devices by sharing some information each other. To do so, a distributed learning is simulated to analyze information that will be shared. These analysis will be done by the implementation of different methods of clustering on these information.
This report is divided into several chapters. In the chapter 2, the presentation of the host organization will be made; This presentation will cover up the main themes dealt by the organization and its computing machines. Chapter 3 is focused on the description of the problem. It will present the different components of automatic speech recognition system and explain what is a distributed learning. And, Chapter 4 will talk about the dataset, the toolkits and the methods used. It is also going to present some experiments and the results obtained. Basically, the chapter 4 is devoted to the work done during the internship.

Chapter 2

Description of the host organization

This section is focused on the host organization where the internship was done.

2.1 Presentation of the Laboratoire Informatique d'Avignon (LIA)

Created in 1978 by Henri Méloni, former president of the University of Avignon and Pays de Vaucluse (UAPV), the LIA is a laboratory of computer science located in Avignon. It has several researchers who work on Natural Language Processing (whether written or oral), Operational Research and Networks. Its researchers are about 65 out of 75 workers. Its staff is made of 29 permanent researchers, around 30 PhD students, internship students, engineers, administrative staff... This laboratory belongs to the Centre d'Enseignement et de Rercherche en Informatique (CERI)[1] associated with the Agrosciences and Sciences Doctoral School of the UAPV. Being part of the Laboratory of Excellence (Labex) Brain and Language Research Institute (BLRI), the laboratory participates in the Agorantic[2] research federation and is involved in many national and European projects. Yannick Estève, professor at the UAPV, is the current director of the laboratory.

2.2 Description of LIA's research themes

Language, Networks and Operational Research are the principal topics treated by the LIA. There are 2 other topics which are: "Digital Societies" and "Complex Systems".

2.2.1 Language

Being focused on speech processing since its creation, the LIA works today on other areas of natural language processing such as processing of written language, analysis of the Internet content, modeling of human-machine dialogue, etc. LIA follows the state of the art in its field of activities. Indeed, multiple approaches and technologies are used by its researchers. Among these, there are probabilistic approaches, machine learning, language model, multimedia indexing, GMM (Gaussian Mixture Modelling), HMM (Hidden Markov Model), LSA (Latent Semantic Algorithm), LDA

[1] http://ceri.univ-avignon.fr/
[2] http://agorantic.univ-avignon.fr/

(Latent Dirichlet Allocation), deep learning, syntactic analysis, iVectors, etc. Research on natural language processing carried out by the LIA are deep and wide. It is done on linguistic and paralinguistic objects taken from different media such as phone conversation, audio recordings ...

2.2.2 Networks

The theoretical and practical aspects of computer science, mathematics and problems located at the interface of networks are the main interests of this topic. In this area, the LIA aims to implement innovative solutions in order to deal with the complexity and the change of scale of the studied problems. Its research are focused principally on game theory, multi-level optimization, linear integer programming (LIP), optimal control, stochastic processes, mechanism design and biological systems. These models allow the LIA to study the different resulting trade-offs, to identify the limits of networks performance, and to implement algorithms and mechanisms to manage them. These researches can be applied in different fields such as transports network, optics network, etc.

2.2.3 Operational Research

Through this thematic, the laboratory is integrated into other sectors of activity such as Spatial Planning, Transport, Information Extraction and Exploitation, and Scheduling. This is possible thanks to the development of Polyhedral Methods, to Quadratic Programming in Binary Variables, to the development of Meta-heuristic methods as well as robust optimization. This thematic is mainly about Discrete Optimization, or Programming in Whole Numbers.

2.2.4 Digital Societies and Complex Systems

Digital Societies The spread of e-commerce impacts many sectors, in particular the cultural industries. Thus, LIA carries out projects which are located at the interface between people in society and networks they maintain or use in order to understand and control changes, anticipate them and make the most of them. Indeed, it depends on the ability of ICST to produce the methodological and technological tools capable of making the internet usable and intelligible. Research around these topics involves many internal or external collaborations at the UAPV. In other words, the LIA collaborates with other human and social science laboratories inside and outside UAPV such as LPL.

Complex Systems To develop tools and methods for describing and modelling systems in order to understand how they work, improve their design and control, and/or predict their behaviour is the LIA's work on complex systems.

2.3 LIA's equipment

In terms of equipment, the LIA uses some clusters. There are 3 kinds of machine: CPU-only compute nodes, GPU-equipped compute nodes and "administrative" machines.

CPU-only compute node These machines run the same OS and the same set of software tools. LIA has in total 17 such machines. Table 2.1 shows the computational power of each machine of this category.

RAM	CPU cores	Number of machines
16 GB	8 cores	3
24 GB	8 cores	2
32 GB	8 cores	6
64 GB	8 cores	2
128 GB	8 cores	1
256 GB	24 cores	2
512 GB	24 cores	1
Total number of machines		17

TABLE 2.1: Computational power of the CPU cluster

GPU-equipped compute node These kind of machines allow the laboratory to make GPGPU computation. They are composed of compute nodes that are equipped with Nvidia GPU. Such machines are 10 in total. Table 2.2 presents the computational power of each of them.

RAM	CPU cores	Number of GPUs	Number of machines
64 GB	12 cores	2	3
128 GB	20 cores	2	2
128 GB	20 cores	4	1
256 GB	20 cores	8	2
256 GB	24 cores	2	1
512 GB	24 cores	1	1
Total number of machines			10

TABLE 2.2: Computational power of the GPU-equipped cluster

"Administrative" machines These machines are not supposed to be used as compute nodes; So nobody is allowed to run anything on them. They host important shared file-systems, web servers, demo servers, etc. There are 7 of them.

Chapter 3

Presentation of the problem

This section presents the problem studied during the internship. It is divided into two parts. The first part is focused on defining different topics, and the second part will describe the problem.

3.1 Definition

3.1.1 Automatic speech recognition

ASR aims to recognize and translate spoken language into text. Since the breakthrough of the deep learning, this field has more interest for many big companies like Amazon, Apple...Since its creation, the ASR has got a lot of improvement. This part is focused on the recent ones. The figure below shows a representative schema of current ASR system based on a statistical modelling.

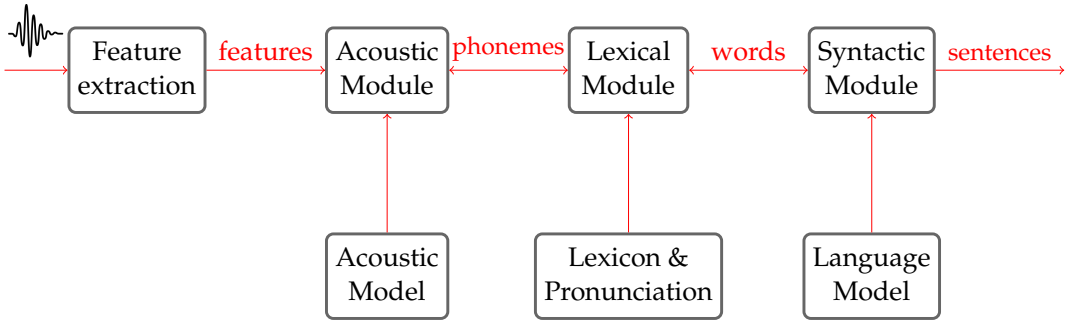


FIGURE 3.1: Automatic Speech Recognition System

Through an ASR, the expectation is to get a text (output) from a raw audio (input). For doing that, the first thing is to extract the features from the raw audio. These features are given to an acoustic model that generate phonemes associated to a probability for each audio segment. This will communicate with a lexical module in order to recognize only words present in a pronunciation dictionary. After that, there is a communication between the generator and the syntactic module to satisfy the language model constraints. The language model is used to estimate the probability of word sequences. It is important to notice that an audio may contains some information (noise, etc.) that have to be process before being given to the ASR system. Thus, a segmentation has to be done. This segmentation aims to group acoustically homogeneous segments in terms of speakers, bandwidth. A segment is a sequence of words spoken without a big pause.

The acoustic modeling improves the accuracy and it is probably the core of any speech recognition system. It is a bridge between an audio signal and the linguistic units (like phonemes) that make up speech. To train the acoustic model, a set of

audio recordings and their corresponding transcripts are given to the model. There are many kinds of acoustic model. Nowadays, traditional approaches such as HMM [Jelinek, 1976; Rabiner, 1989] is combined with GMM or DNN; this leads to hybrid approaches [Bourlard and Morgan, 1993]. Previously, the hybrid approach used in most of the work was HMM-GMM based models for training and decoding. Now, HMM-DNN based models have proved to be the best way for most ASR systems [Hinton et al., 2012].

REF:	***	I want to go	to	LYON
HYP:	well	I want to go	**	LANNION
	I		**D**	**S**

FIGURE 3.2: Alignment of automatic transcription (HYP) with reference transcription (REF). I: insertion, D: deletion, S: substitution

Three types of error faced while making an ASR are: insertion (add word where there is no word), deletion (delete a word) and substitution (replace a word by another). The figure 3.2 shows an example of these errors.
To evaluate an ASR system, Word Error Rates (WER) is used [Pallett, 2003]. This metric is computed by taking into account the errors enumerated above (Equation 3.1). Actually, after the alignment, a WER score is used to estimate the recognized words with the reference transcription.

$$WER = \frac{N_{sub} + N_{ins} + N_{del}}{N_{refwords}} \tag{3.1}$$

where:

N_{sub} = number of substitutions
N_{del} = number of deletions
N_{ins} = number of insertions
$N_{refwords}$ = number of words in the reference

3.1.2 Distributed learning for ASR

Among the fundamental algorithms, There are two types of algorithms with sometimes blurred boundaries between them: These are parallel algorithms and distributed algorithms. In a parallel system, the aim is to separate a big problem into many independent sub-problems. While a distributed system is a system in which independents process interact and cooperate together to solve the same problem. In distributed system, each process starts with the same algorithm; it can share information to the other processes and make some local computations.
The distributed learning has the same principle as distributed algorithm in fundamental algorithm excepted that the distributed learning was mainly used to accelerate the learning in ASR system [Zhang et al., 2019]. A distributed learning can have a centralized architecture or a decentralized one (Figure 3.3). When the centralized architecture is used, the local data are uploaded to the server. Thus, for speech recognition, the computation is made on users devices and the decoding on

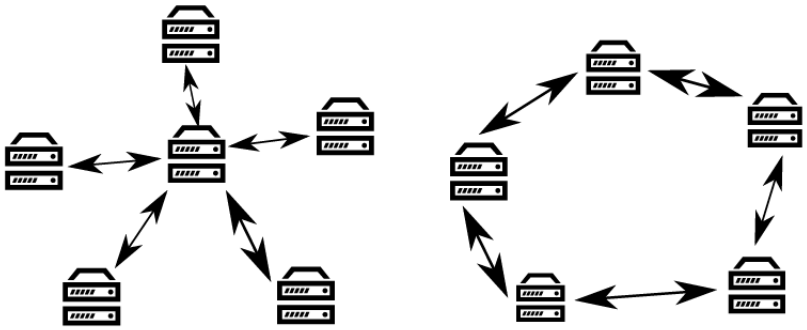


FIGURE 3.3: A centralized distributed learning architecture on the left and a decentralized distributed learning architecture on the right (from Zhang et al., 2019)

the server. While for decentralized distributed learning, the data are identically divided and uploaded to all the machines. So the computation and decoding is made on the devices.

3.2 Description of the problem

There are a lot of devices that use voice technologies in order to interact with humans. These technologies are more powerful thanks to ASR systems that are implemented by the big companies such as Google, ... To improve their performances, user data are taken from the devices and sent to a server. But audio can contain sensitive information and can be used to identify individuals; collecting data through devices and putting them on a central server can be a risky in term of privacy. Indeed, users have few guarantees about how the data are used. Therefore, a question should be asked: how to protect users privacy while improving speech recognition system? To answer this question, a national project named DEEP-PRIVACY is funded by the French national research agency (ANR). The project aims to design and implement privacy-preserving approaches for speech processing. One of these approaches consists on investigating the distributed learning paradigm. Making local computations on the devices of each user, and cross-user computations of non sensitive information through a server is an important way the research on DEEP-PRIVACY will be done. Actually, DEEP-PRIVACY focused on 2 principles objectives: learn privacy-preserving representations of the speech signal, and distributed algorithms and personalization. The idea of studying privacy-preserving representations is to transform the speech signal in order to understand which features extracted can contain sensitive information. Distributed algorithms and personalization will solve the problem of the protection of sensitive data. Indeed, the sensitive information will remain on the users devices. A personalized algorithm will be done and some essentials information to improve the learning will be shared to the server.
The DEEP-PRIVACY project is divided into many parts and involves three Computer Science Labs. A decentralized distributed learning for speech recognition in the context of privacy protection is one part of DEEP-PRIVACY project. The first objective is to implement several approaches for acoustic modeling and machine learning algorithm in the case that the data remain on the users devices. Only the weights computed locally can be updated. The second objective is to study the information that will be shared.

The internship is focused on the distributed learning for ASR. It works on the implementation in PyTorch (4.1.1) of an acoustic model and the analysis of the exchanged contents. To analyse the exchanged content, a distributed learning has to be simulated. Thus, assuming that there is many users who starts the learning with the same model in which the weight is extracted during the learning is the first step. So the data will be divided into two subsets: the first one will be used to train a generic model and the second one will represent the users in their devices.
This work represents a new set of researches which has not been done before.

Chapter 4

Work done

4.1 Experimental Setup

4.1.1 Toolkits

Kaldi [Povey et al., 2011]: Kaldi is an open-source toolkit under the Apache License v2.0[1] that allows to work on signal processing and speech recognition. This toolkit is written in C++ and mainly created to build a recognition system. Figure 4.1 shows a simplified schema of automatic speech recognition with Kaldi toolkit; from the input (the raw audio) to the output (the lattice obtained).

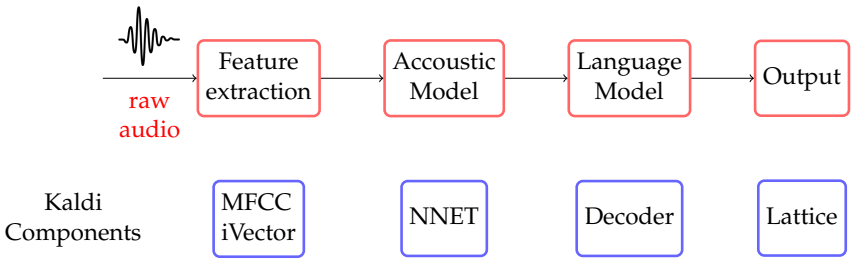


FIGURE 4.1: ASR with Kaldi (simplified schema)

To run kaldi on raw audio, some instructions are given. These instructions are recorded into a file and form a kaldi recipe. A kaldi recipe is constituted of many phases:

- Data, lexicon and Language preparation: The data preparation phase consists of the generation of input file used by kaldi while the lexicon preparation consists of building language knowledge - lexicon and phone dictionaries.
- Feature Extraction: Kaldi generates features that can be fed into some neural models. Some of the features generated by Kaldi are: fMLLR (Feature space Maximum Likelihood Linear Regression) [Gales, 1998] , fbank (filter bank), Mel-frequency cepstral coefficients (MFCC) [Davis and Mermelstein, 1980],... MFCC is the most used feature. MFCC are coefficients that make up a mel-frequency cepstrum (MFC) collectively. Based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency, MFC is a model of the short-term power spectrum of sound.
- Monophone or triphone training: A monophone training is to train the acoustic model without including contextual information (information about the following phone or the preceding phone). While the triphone training produces

[1] https://www.apache.org/licenses/LICENSE-2.0

a phoneme by taking into account phonemes around (left and right). This is the context. Indeed, this is useful because phonemes can vary depending on their context.

- Decoding: Basically, it is the task to catch the most possible sequence of words.

PyTorch-Kaldi [Ravanelli, Parcollet, and Bengio, 2019]: PyTorch-Kaldi is an open-source toolkit for Automatic Speech Recognition (ASR). PyTorch-Kaldi is based on PyTorch[2] developed by Facebook researchers. Indeed, PyTorch is a machine learning framework that allows to build neural network in a simple and flexible way. Pytorch-Kaldi was created with the aim of filling the gap between the PyTorch and the Kaldi toolkit presented in the beginning of section 4.1.1. It consists of developing state-of-the-art Deep Neural Network - Hidden Markov Model (DNN/HMM) speech recognition systems. Each of them (Kaldi and PyTorch) plays a specific role: Kaldi is used to extract features, compute the labels related to the features and make decoding, while Pytorch is used to manage the DNN part (see figure 4.2). One advantage of using PyTorch-Kaldi is that there are many features that users can modify as they wish. Indeed, There are several pre-implemented neural networks in PyTorch-Kaldi that users can use at their convenience by the help of a configuration file. Another advantage of using PyTorch-Kaldi is that you can feed into the neural network created with PyTorch many features generated by Kaldi. These features are concatenated and after given to the neural network.

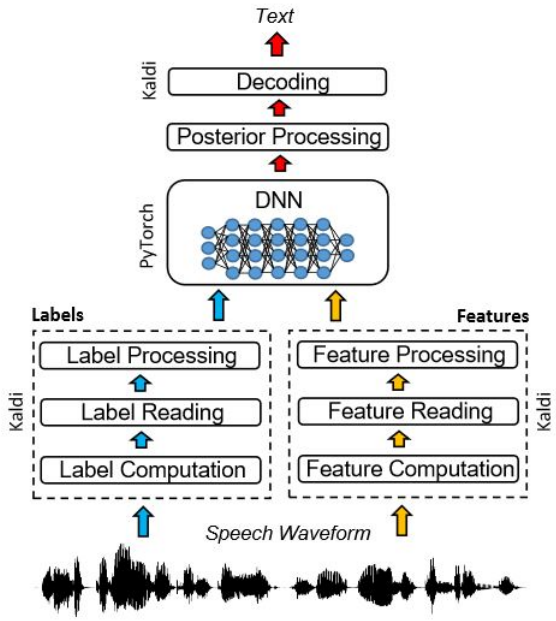


FIGURE 4.2: The architecture of PyTorch-Kaldi *[Ravanelli, Parcollet, and Bengio, 2019]*

4.1.2 Datasets

Description: An Automatic Speech Recognition Corpora, created by the LIUM (Compteur Science Labs of Le Mans' University), was used. This corpus was named

[2] https://pytorch.org/

TEDLIUM. It is a corpus based on TED Talks. The corpus was made by the extraction of videos and captions from the website of TED. There are three releases but this work is focused on the TEDLIUM Release 3 [Hernandez et al., 2018]. This release has two distributions: a legacy version which has the same data sets for development and testing as for the other two releases, and a 'speaker adaptation' version designed for speaker adaptation experiments. In this report, only the speaker adaptation distribution is used. All the statistics about the training set, the development set and the testing set are in table 4.1. It should be noted that some speakers may have more than one talk. Indeed, speakers may have many talks from different years.

Dataset	Train	Dev.	Test
Duration of speech, hours	346.17	3.73	3.76
Number of speakers	1938	16	16
Number of talks	2281	16	16

TABLE 4.1: Details of TEDLIUM 3 (speaker adaptation version). Statistics computed without take into account silence, noise, ...

Dataset partitioning: For experiments, the training set was divided into 2 disjointed subsets (one speech can not belong to both subsets and the speeches of the same author have to be in the same subset). To make sure that the subsets are each other disjointed, the division was made by considering the number of author. The first subset called "partition 1" contains 2/3 of the speakers and the second one called "partition 2" contains the remaining speakers, in other words 1/3 of the total speakers. Table 4.2 shows the statistics obtained after the division.

Partition name	number of speakers	number of talks
Partition 1 (2/3 of speakers)	1288	1514
Partition 2 (1/3 of speakers)	644	765

TABLE 4.2: Dataset's partition

4.1.3 Model

The architecture is composed of recurrent neural network (RNN) [Jain and Medsker, 1999] specially GRU (Gated recurrent unit) [Cho et al., 2014] and MLP (Multilayer perceptron)[Rumelhart, Hinton, and Williams, 1986] layers as softmax classifiers.
The input is an MFCC feature which is generated using Kaldi. Then, the MFCC is fed into the GRU layers architecture followed by two classifiers. The first classifier estimates the standard context-dependent states and the second one, the monophone targets. After that, the weighted sum between the predictions was computed. The weighted sum computed is the final cost function. [Bell and Renals, 2015, Bell, Swietojanski, and Renals, 2017] show that this multitask prediction improved the ASR performance. Figure 4.3 presents the model architecture and how the neural networks are connected.

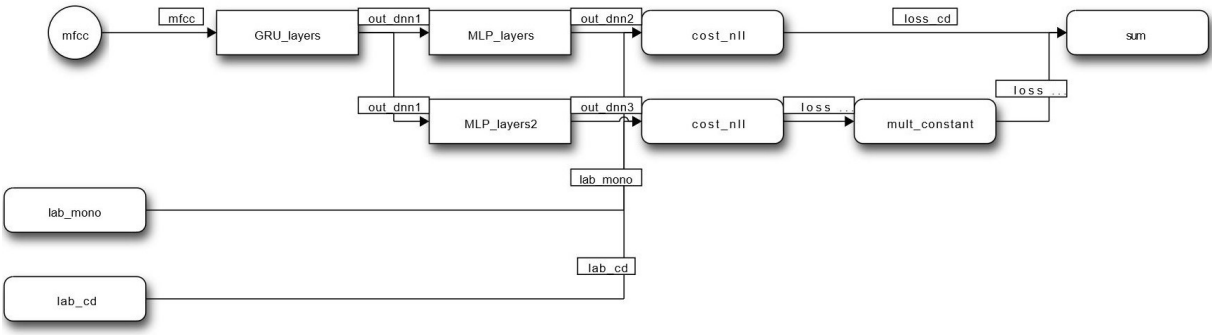
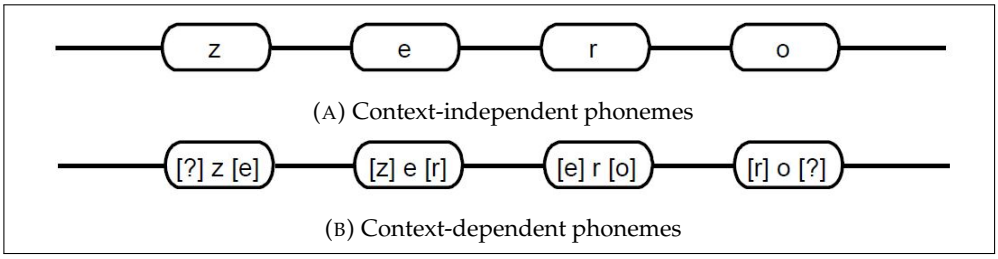


FIGURE 4.3: Neuronal architecture

Context-dependent phone state assumes that a phoneme is dependent of phonemes around, of the context (Figure 4.4b) while context-independent targets assumes that the phonemes are trained independently (Figure 4.4a).



(A) Context-independent phonemes

(B) Context-dependent phonemes

FIGURE 4.4: Context-independent and context-dependent phonemes

Parameters of the GRU architecture: The architecture has 5 bidirectional GRUs; each of them has 550 units, a dropout of 0.2 to avoid overfitting, a batch normalization and tanh as activation function. The learning rate of the architecture is 0.0004. It should be noted that the learning rate will decrease if the relative performance improvement on the dev-set between two consecutive epochs is smaller than a certain threshold. In this case, this threshold is set to 0.001 (Table 4.3).

GRU units	dropout	Learning rate	Activation function
550	0.2	0.0004	tanh

TABLE 4.3: Parameters of each GRU layer

Parameters of the classifiers: The classifiers have the same learning rate, threshold as the GRU layers. In contrast, the activation function is softmax as it was mentioned above. The optimizer used is rmsprop (Root Mean Square Propagation).

All these parameters are set in the configuration file of PyTorch-Kaldi. An example of a configuration file is provided in the Appendix A.

4.2 Generic acoustic model and speaker based model

The problem described in section 3 supposes that all users start with the same model trained on large data. Then, all users trained their own model locally to be powerful on their data. The first model used at the beginning of the experiment is called generic acoustic model and the next trained locally is called speaker based model.

4.2.1 Generic acoustic model

The generic acoustic model was made using the architecture defined in section 4.1.3. This model was trained on the "partition 1" (2/3 of authors in the training set). The training was made for 23 epochs and took 348.85 hours around 14.5 days. After the training, a WER (Equation 3.1) score of 14.45 was obtained on non-training data (the testing set). This result is pretty good and can be improved as well as the architecture will be. All the statistics are consigned in Table 4.4.

Number of epochs	Run time (second)	Run time (day)	Run time (per epoch)	%WER
23	1 255 866	14.5	15 hours	14.45

TABLE 4.4: Generic model result

4.2.2 Speaker based models

The speaker based models were made to simulate an ASR distributed learning. The dataset used for each model is the dataset named "partition 2" composed of 765 talks. The dataset was divided disjointedly to form many subsets of speech for the same speaker. Table 4.5 shows how each talk was divided. Indeed, this division depends on the number of segments of each speech (e.g. if the number of segments for a speech is above 80, the speech is divided into 4 subsets). After division, 2745 subsets of speech were obtained from 765 talks in the beginning. Each speech lasts about 200 seconds (without silence) in average.

	$x \geq 80$	$60 \leq x < 80$	$x < 60$
Number of talks	580	55	130
Subsets created	4	3	2

TABLE 4.5: Number of subsets of talks depending on the number of segments for each 'new' talk

Each subset of talks was considered as an independent talk of a new user on his device. Thus, 2745 models was created; one model for each subset of talk. These models have the same architecture than the one in section 4.1.3. Not having a powerful model on the new data, each model starts with the weights of the generic model that performs well. The speaker based models were finetuned from the generic model. A finetuning is a transfer learning in which a pretrained model was used to perform a new task. In this case, the models were initialized, before starting the training, with the weights of the generic model. Each model was trained for 4 epochs and took around 100 seconds. To evaluate each model, a WER score was computed. After computation, a WER score of 13.94 in average was obtained (Table 4.6).

Number of epochs	Run time (per model)	%WER (in average)
4	100 seconds	13.94

TABLE 4.6: Speaker based model results

4.3 Experiment 1: clustering of weight matrix

4.3.1 Experiment

In order to study the impact of the distributed learning on the data, the weights of each layer were extracted every two epochs during the training of the speaker based models. Then, with these weights, a hierarchical clustering [Müllner, 2011] was made to group up the talks of the same speaker. The principle of the hierarchical clustering is to build a hierarchy of clusters. There are two approaches : a bottom-up approach (Agglomerative) and top-down approach (Divisive). In Agglomerative clustering, the algorithm starts by assuming that each point is his own cluster and next a merge of 2 clusters is made recursively until having one cluster at the end. While Divisive approach is the opposite; the algorithm starts with one cluster that group all the data, and then splits are performed recursively until having one data representing a cluster in each leaf (Figure 4.5).

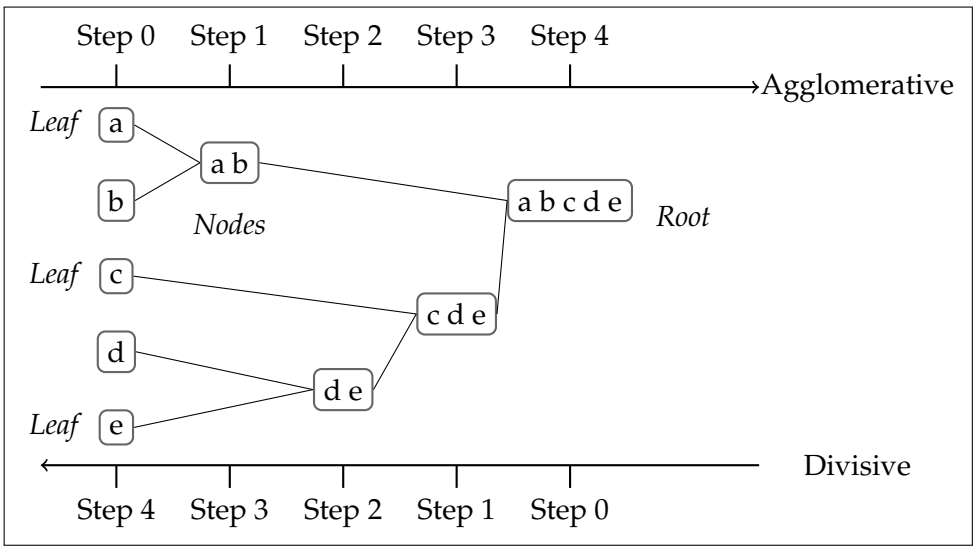


FIGURE 4.5: Illustration of the approaches of hierarchical clustering

Many kinds of distance measuring can be used to make the hierarchical clustering such as cosine distance, euclidean distance, etc. In this work, the euclidean distance is used. This distance is computed as follow:

$$d(x,y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.1}$$

where:

x = point of coordinates x_1 and y_1
y = point of coordinates x_2 and y_2

For agglomerative approach, when two clusters are merged, a new distance between the new cluster merged and the other clusters is computed. There are many linkage methods to compute this distance. Given s, t, v some clusters; Let's assume that s and t will be merged. The new distance between (s,t) and v will be:

- single method: the minimum distance between $d(s,v)$ and $d(t,v)$
- complete method: the maximum distance between $d(s,v)$ and $d(t,v)$
- weighted method: the average distance between $d(s,v)$ and $d(t,v)$
- ward method: it based on Ward variance minimization algorithm [Ward, 1963]. The formula used to compute the Ward's criterion is showed in equation 4.2.

$$d(\{s,t\},v) = \sqrt{\frac{|v|+|s|}{|v|+|s|+|t|}d(v,s)^2 + \frac{|v|+|t|}{|v|+|s|+|t|}d(v,t)^2 - \frac{|v|}{|v|+|s|+|t|}d(s,t)^2} \tag{4.2}$$

where:

s,t = the clusters that will be merged
v = an other cluster
$|x|$ = the cardinality of x
$d(x,y)$ = distance between x and y

The linkage method used in this work is ward method.

4.3.2 Results

The results of hierarchical clustering are usually presented in a dendrogram. A dendrogram is a diagram that shows the hierarchical relationship between objects (talks in our case). Figure 4.6 and 4.7 present the dendograms obtained after the execution of a hierarchical clustering on the weight matrix of each talk extracted from the first GRU layer after the first epoch and three epochs respectively.

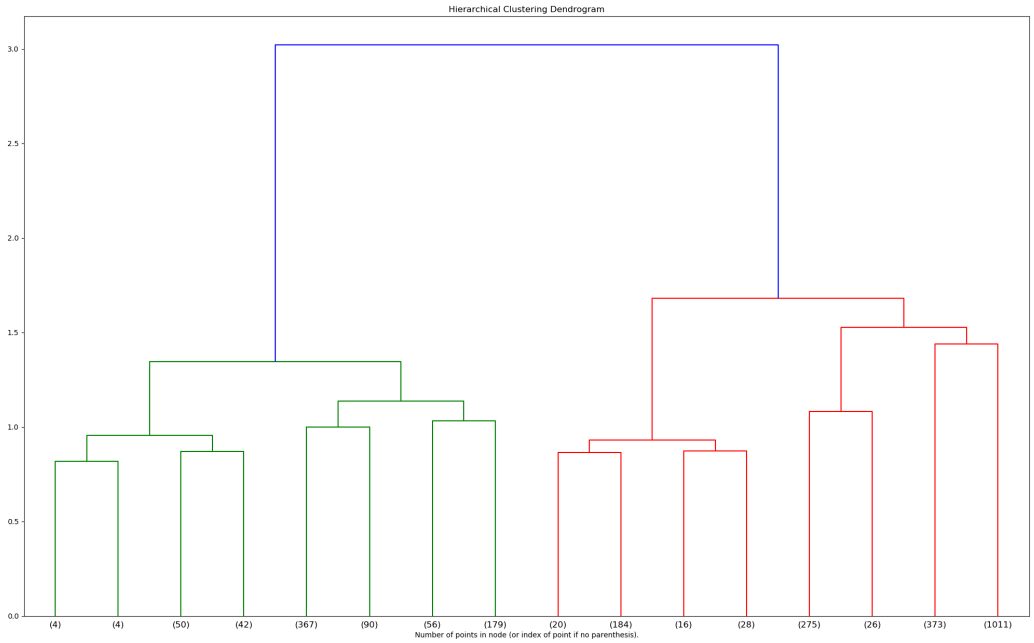


FIGURE 4.6: Hierarchical Clustering Dendrogram (weights of the first GRU layer after one epoch)

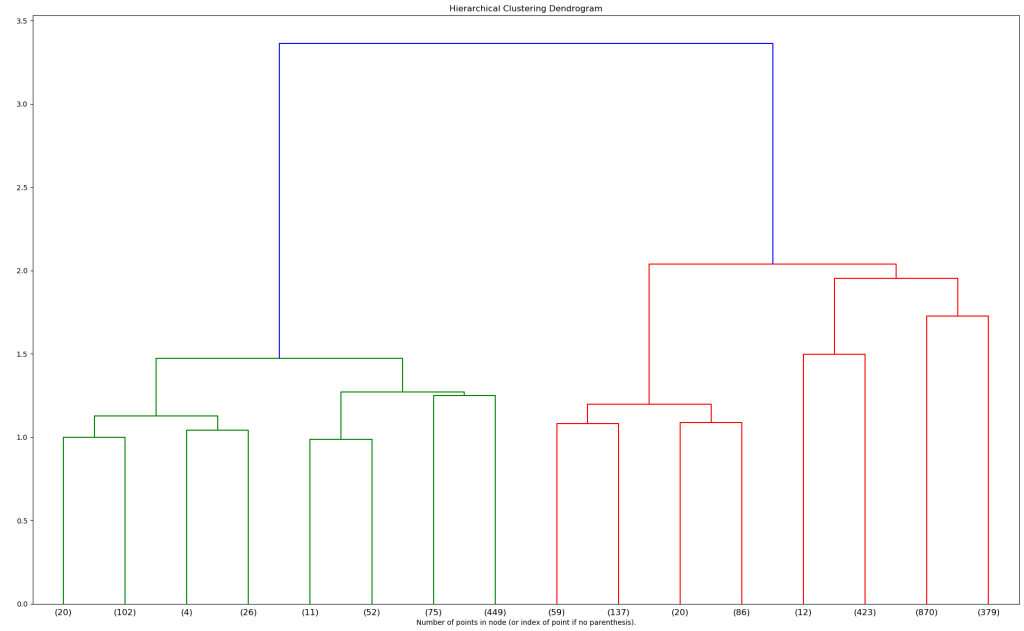
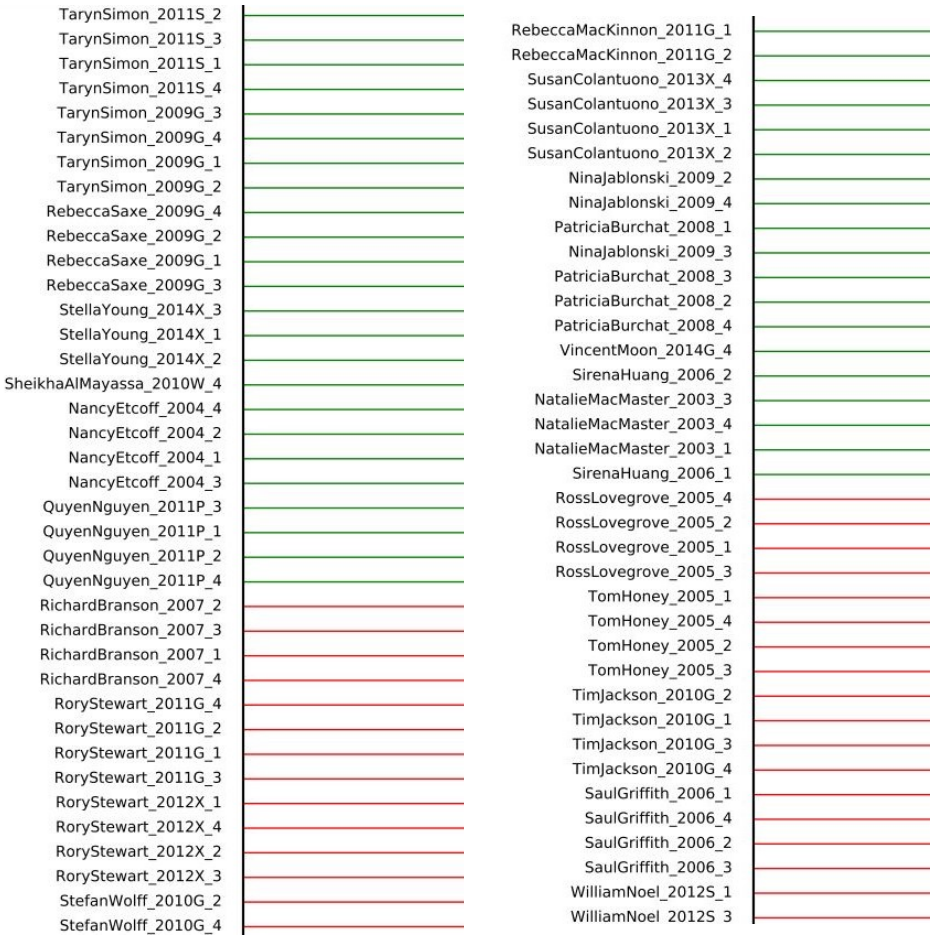


FIGURE 4.7: Hierarchical Clustering Dendrogram (weights of the first GRU layer after 3 epochs)



(A) Epoch 0 layer 0

(B) Epoch 2 layer 0

FIGURE 4.8: Zoom on the dendograms

These dendograms show two big groups: a red one and a green one. Figure 4.8 presents an unwrapped version of the dendogram. By looking at the unwraped dendograms, it seems that the talks of the same speaker are grouped together and the 2 big groups formed are groups of gender (men and women). This leads to make some experiments about speaker clustering (see section 4.4) and gender identification (see section 4.6) . By comparing Figure 4.8a (unwrapped dendogram of epoch 0 layer 0) and Figure 4.8b (unwrapped dendogram of epoch 2 layer 0), the data at the border of the two big groups are not the same. That leads to think that based on the epoch, the grouping is different.

4.4 Experiment 2: speaker clustering

In section 4.3, it seems that the talks of the same user are grouped together. To verify this hypothesis only the speeches having 4 talks were kept. Hence, some techniques to obtained 4 elements per cluster were used. Doing that, the accuracy will be how good the clustering method is. Thus, a random method and two methods based on hierarchical clustering have been used. The entropy measure and the purity are used to measure the quality of the clustering. Indeed, entropy measures how disordered a system is. A greater entropy means that the clustering is not good. To compute global entropy (see Equation 4.4), the entropy of each cluster was computed in the beginning (Equation 4.3). The total entropy of the system depends on the entropy of each cluster. The purity measure is used to know if a cluster contains a single class. When its value is equal to one, it means that the cluster contains elements of the same class. Equation 4.5 shows how to compute the purity measure. The methods used for the clustering and the results obtained are described bellow.

$$H(i) = -\sum_{j \in K} P(i_j) \log_2 P(i_j) \tag{4.3}$$

where:

$H(i)$ = entropy of the cluster i
$P(i_j)$ = probability that a data in the cluster i is classified as class j
K = set of class

$$H = -\sum_{i \in C} H(i) \frac{N_i}{N} \tag{4.4}$$

where:

N_i = the size of the cluster i
N = number of data in total
C = set of clusters

$$Purity = \frac{1}{N} \sum_{i=1}^{k} \max_j |c_i \cap t_j| \tag{4.5}$$

where:

c_i = a cluster
N = total number of data
t_j = a class with max count for c_i

For all the methods, two cases have to be considered for computing purity and entropy:

- Case 1: Each talk forms a class.
- Case 2: The talks of the same speaker form a class. Having some speakers that have talks in different years in the dataset, it will be interesting to make some analysis.

4.4.1 Random Clustering

This method is based on a random choice of talks in the dataset to form a cluster. The random choice allows to know how much better other methods are. The talks being divided into 4 subsets, clusters of 4 elements are formed. The algorithm used is showed as follow:

Repeat

- Choose 4 talks randomly
- Add them to the list of clusters by assigning an unique label.
- Remove the forth from the choice list

While there is a talk in the list of choice

4.4.2 Method 1: based on Divisive hierarchical clustering algorithm

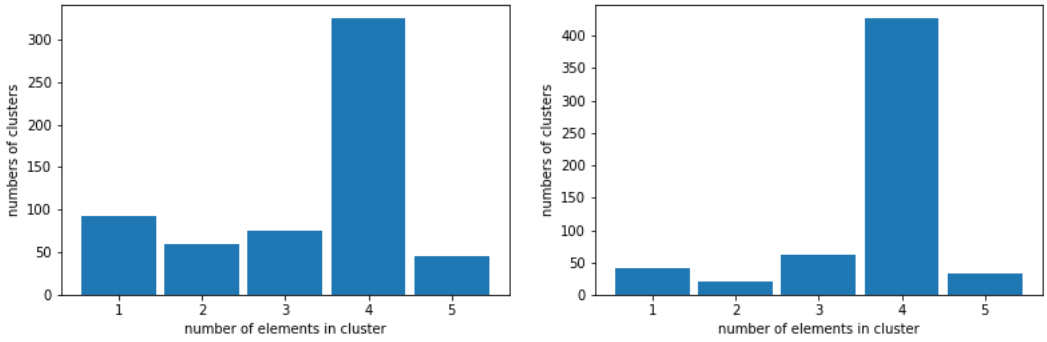
The method 1 is the implementation of the divisive hierarchical clustering. Indeed, there is some implementations of this algorithm but without taking into account the number of elements in each cluster. Thus, for doing that, an implementation that restricting the number of elements at $4 \times \epsilon$ elements was done. The algorithm used is defined as follow:

Given a dataset $X = (x_1, ...x_n)$
Initialize ϵ to 1.5
At the beginning, all the data belong to one cluster
Repeat

- Split the cluster that have the maximum element in two clusters using a clustering method (hierarchical clustering, KMeans, ...)

Until the number of elements in each cluster is less than $4 \times \epsilon$.

Figure 4.9a and 4.9b show the number of elements by cluster using the algorithm described. Most of the cluster have 4 elements; Some have 1, 2, 3 and even 5 elements. Having 5 elements in one cluster can be relevant if these elements are from the same speaker. For epoch 0, there are around 100 clusters that have 1 element and less than 400 clusters that have 4 elements. While for epoch 2, there are around 50 clusters that have 1 element and more than 400 that have 4 elements. This observation tends to mean that after 3 epochs the method is more efficient.



(A) Method 1: epoch 0 layer 0 (B) Method 1: epoch 2 layer 0

FIGURE 4.9: Number of elements per cluster

4.4.3 Method 2: based on Agglomerative hierarchical clustering algorithm

The agglomerative hierarchical clustering algorithm is implemented using python on the libraries such as Scipy[3] and Scikit-learn[4] but there is no limitation about the size of the cluster. The algorithm proposed in this part is based on the functional principle of the agglomerative with fixed size of cluster. The algorithm used is:

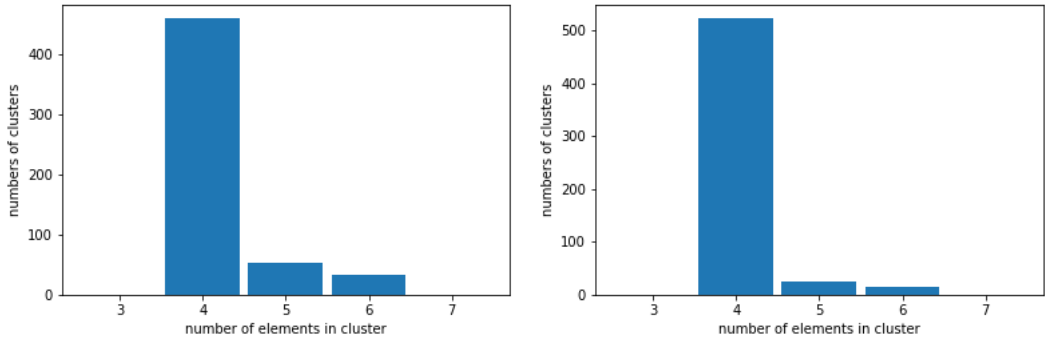
Given a dataset $X = (x_1, ...x_n)$
Compute a distance matrix using euclidean distance (Equation 4.1) for each pair of data
Each data is considered as a cluster
Repeat

- Merge the two clusters that have the minimum distance in the distance matrix
- Compute the new distance between the new cluster formed and the other clusters using the ward's linkage criterion (Equation 4.2) and update the distance matrix.
- Make a verification: if the number of elements in the cluster formed is equal to 4 or greater than 4; Add this cluster to the list of complete clusters.

Until all the cluster are in the list of complete clusters.

Figure 4.10a and 4.10b present histograms of the number of elements per cluster. For this algorithm, there are clusters that have only 4, 5 and 6 elements. This is possible when a cluster of 3 elements merges with a cluster of 2 elements or 3 elements. Like the method 1, the number of cluster that contains 4 elements has increased from epoch 0 to epoch 2.

[3] https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html



(A) Method 2: epoch 0 layer 0 (B) Method 2: epoch 2 layer 0

FIGURE 4.10: Number of elements per cluster

4.4.4 Results and comparison of the different approaches

After computing method 1 and 2 on the dataset, the results obtained is consigned into table 4.7, table 4.8 and table 4.9.

- For epoch 0 (Table 4.7a): The number of cluster is equal to 656 using method 1 which is about 3.5 elements by cluster while the expectation was 4 elements per cluster. In contrast, 548 clusters were obtained for method 2 about 4.22 elements per epoch. Having 4.22 elements per cluster in average is not bad at all considering that some speakers have more than one speech made from different years.
- For epoch 1 (Table 4.7b): Independently of the methods used (method 1 or method 2), the number of elements in average per cluster tends to be 4. For method 1 there are 611 cluster obtained in total while for method 2 there are 564.

At first view, making more epochs tends to bring the subsets of speeches closer together.

	Number of cluster	Number of element in average
Method 1	656	3.5
Method 2	548	4.22

(A) epoch 0

	Number of cluster	Number of element in average
Method 1	611	3.78
Method 2	564	4.099

(B) epoch 2

TABLE 4.7: Statistics obtained for the first layer

The purity measure and the entropy were computed by considering the two cases mentioned above. Table 4.8 has treated the case in which each talk forms a class while table 4.9 considers the case that the talks of the same speaker form a class.

- Each talk forms a class: The random method gives low results. Indeed, the number of pure clusters (clusters that are composed of exactly one class) is null. There is a disorder in each cluster in sense that the entropy is high and

the purity measure is low. The result could not have been better, since it's random.
The number of pure cluster obtained by the execution of method 1 is higher than the one for method 2. This is because there are some clusters that contain only 1 element when you run method 1. Which leads to higher purity and lower entropy. Thus, if the cluster having 1 element is removed from the computation, both methods tend to have the same results. Another interesting point is that the clustering on the weights extracting after the third epoch has good result than the one on the weights extracting after the first epoch, and this using the method 1 and method 2. Using method 2 on the weights of epoch 2, 82.62 % of the clusters are pure (Out of 564 clusters, 466 are pure) opposed to 62.77 % on the weights of epoch 0.

		Number of pure cluster	Purity	Entropy
Random Clustering		0	0.253	1.99
Method 1	epoch 0	494	0.878	0.36
	epoch 2	528	0.942	0.18
Method 2	epoch 0	344	0.822	0.49
	epoch 2	466	0.919	0.22

TABLE 4.8: Purity and entropy in the case 1

- Considering that the talks of the same speaker form a class, from one epoch to another the number of pure cluster increased (from 509 after epoch 0 to 545 after epoch 2) for both methods. The purity measure has increased and the entropy has reduced compared to the first case.

		Number of pure cluster	Purity	Entropy
Random Clustering		0	0.254	1.99
Method 1	epoch 0	509	0.893	0.32
	epoch 2	545	0.955	0.14
Method 2	epoch 0	359	0.84	0.45
	epoch 2	480	0.929	0.19

TABLE 4.9: Purity and entropy in the case 2

The result obtained in case 2 is higher because sometime the subsets talks of the same speaker from different talks are grouped together. In 4.10a, cluster

195 is a pure cluster independently of the case considered; Indeed, it contains the subset of talks of the same talk and speaker. But cluster 498 and 450 are pure only if the second case is considered. This is the same for 4.10b in which the clusters 258, 343 and 461 are not pure unless case 2 is regarded. This observation leads to consider the second case to evaluate the clustering.

Subset of talks	Cluster
NielsDiffrient_2002_1	450
NielsDiffrient_2002_2	450
NielsDiffrient_2002_3	498
NielsDiffrient_2002_4	498
NielsDiffrient_2002a_1	450
NielsDiffrient_2002a_2	450
NielsDiffrient_2002a_3	498
NielsDiffrient_2002a_4	498
NigelMarsh_2010X_1	195
NigelMarsh_2010X_2	195
NigelMarsh_2010X_3	195
NigelMarsh_2010X_4	195

(A)

Subset of talks	Cluster
PaolaAntonelli_2007P_1	258
PaolaAntonelli_2007P_2	258
PaolaAntonelli_2007P_3	258
PaolaAntonelli_2007P_4	343
PaolaAntonelli_2007_1	461
PaolaAntonelli_2007_2	258
PaolaAntonelli_2007_3	461
PaolaAntonelli_2007_4	461
PaolaAntonelli_2013S_1	343
PaolaAntonelli_2013S_2	343
PaolaAntonelli_2013S_3	461
PaolaAntonelli_2013S_4	343

(B)

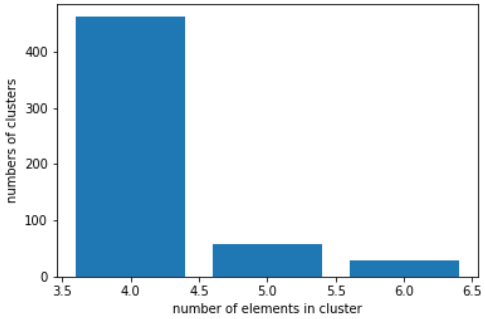
TABLE 4.10: Result of the clustering (method 2 epoch 2)

4.4.5 speaker clustering (weights of the other layers)

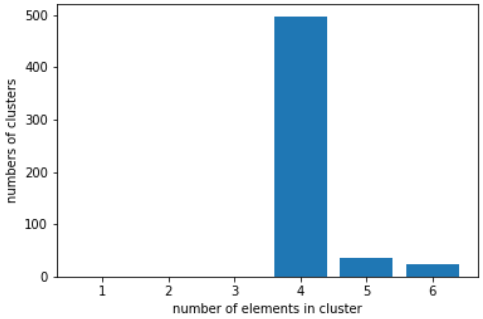
For the other layers, only the results obtained using method 2 is presented in this part. The execution was done on the weights extracting to layer 1, 2, 3 and 4 of the speaker based models (section 4.2.2) after epoch 0 and epoch 2. Figure 4.11 shows some bar charts that contain the number of elements per cluster. In fact, from epoch 0 to epoch 2, the number of cluster that have 4 elements increase. The results of the clustering obtained are consigned in Table 4.11. These results confirmed the observation made in section 4.4.4. Actually, the percentage of pure cluster at epoch 0 for layer 1, 2 and 3 is higher than the one at epoch 2. In term of percentage, layer 2 gives the best result independently of the epochs (epoch 0 or 2).

Layers	Epochs	Number of clusters	Number of pure clusters	Percentage of pure clusters
layer 1	epoch 0	549	343	62.48
	epoch 2	558	361	64.70
layer 2	epoch 0	554	356	64.26
	epoch 2	565	380	67.26
layer 3	epoch 0	554	347	62.64
	epoch 2	561	367	65.42
layer 4	epoch 0	546	344	63.00
	epoch 2	552	347	62.86

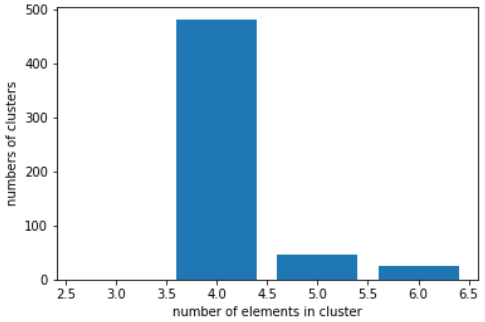
TABLE 4.11: Results obtained when method 2 was used on the weights of the other layers



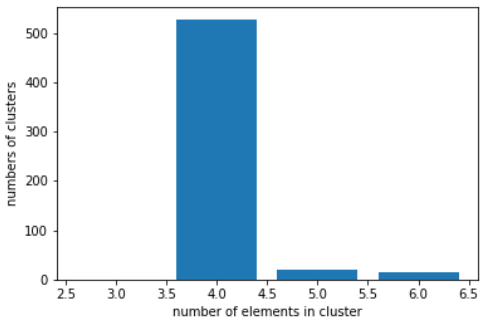
(A) Epoch 0 layer 1



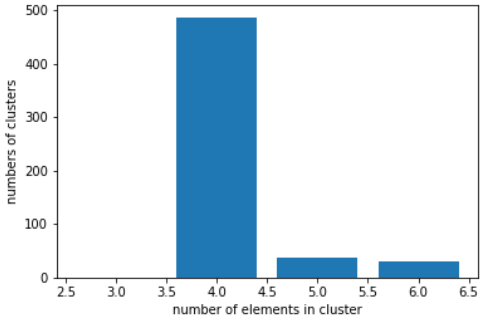
(B) Epoch 2 layer 1



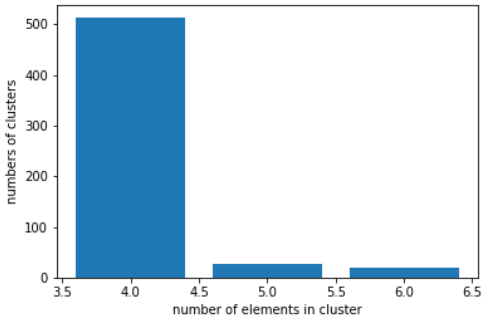
(C) Epoch 0 layer 2



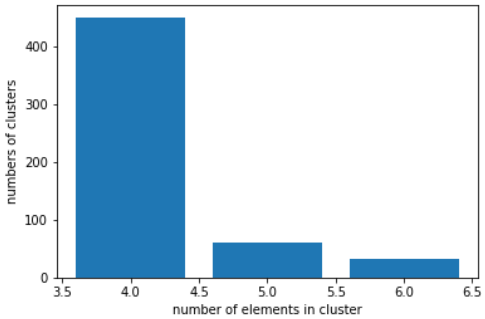
(D) Epoch 2 layer 2



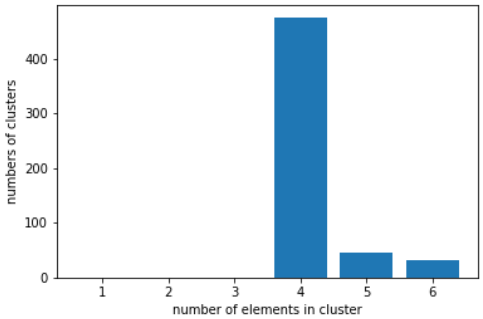
(E) Epoch 0 layer 3



(F) Epoch 2 layer 3



(G) Epoch 0 layer 4



(H) Epoch 2 layer 4

FIGURE 4.11: Number of elements per cluster using method 2

4.5 Experiment 3: speaker identification

4.5.1 Experiment

The objective of this experiment is to know how closed in term of euclidean distance (Equation 4.1), two subsets of the same speaker are when they are put with subsets of others speakers. To do so, an experiment was defined as follow.

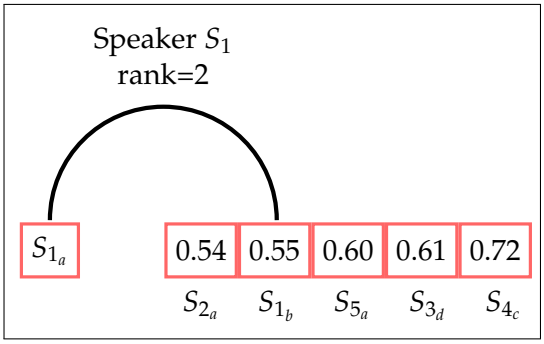


FIGURE 4.12: Random experiment to identify speaker

In Figure 4.12, 2 sub-talks of the same speaker was taken and 4 sub-talks from other speaker was chosen randomly. The subset a of speaker S_1 is closer to the subset b of speaker S_2 than its own subset. Its subset is in rank 2 in term of euclidean distance. Thus, the experiment will consist of obtaining the average rank if the experiment is repeated k times.
In this paper 49 subsets of other subsets were taken and the experiment was repeated 50 times. The algorithm is given as follow:

Given a dataset $X = (x_1, ...x_n)$ of size n; a class $C = (c_1, ...c_m)$ of size m; a distance matrix $d = (d_{x_1}, ...d_{x_n})$ such that d_{x_l} is the distance vector between x_l and $X \setminus \{x_l\}$; $k = 50$
For each class c_t

- Choose two data x_i and $x_j \in c_t$
- For k iterations:
  - Choose randomly 49 others data in the dataset that belong to $C \setminus \{c_t\}$
  - Get and sort the distance vector d_{x_i} between x_i and the 50 others data
  - Select the rank of the data x_j in the sorted distance vector
  - Add the rank to the rank list of speaker c_t
- Compute the average rank of class c_t

4.5.2 Results

The algorithm defined in section 4.5 was implied on the weights extracted to all the layers and for epoch 0 and 2. The results are consigned in Table 4.12. These results are the average rank of all the speakers: Let's suppose $R = \{r_i, i \in [1, n]\}$ the set of rank such that r_i is the average rank of speaker i. For each epoch and each layer, the average rank will be $\frac{r_1+...+r_n}{n}$.

Looking table 4.12, layer 0 has the best result. On average, a sub-talk of a speaker is at the 17th position in term of euclidean distance.

Layers	Epoch	Average rank
layer 0	epoch 0	19.47
	epoch 2	17.69
layer 1	epoch 0	23.33
	epoch 2	22.03
layer 2	epoch 0	22.55
	epoch 2	20.53
layer 3	epoch 0	22.14
	epoch 2	21.36
layer 4	epoch 0	24.04
	epoch 2	23.96

TABLE 4.12: Average of speaker rank (per layer and epoch)

4.6 Gender identification

The hierarchical clustering made in section 4.3 showed at first glance that the speaker of same gender are grouped together. Thus, it is important to verify the hypothesis whereby the gender matter for the system. For this part also, the dataset used is the "partition 2".

4.6.1 Corpus labeling

The dataset is provided with no information about the gender of each speaker. Thus, using the web of TED conference, the labeling of the corpus "partition 2" by defining the gender was done manually. The labelled corpus, Figure 4.13 shows a bar chart

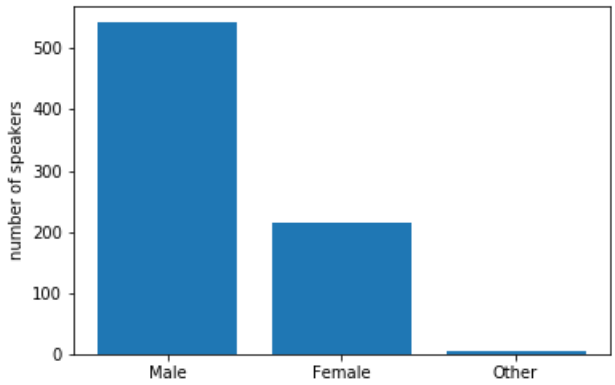


FIGURE 4.13: Distribution of the label of gender

that presents the different labels of the corpus and the distribution of these labels. Most of the speakers are males (more than 500). Around 220 speakers are females and a very small part is constituted of a group of persons. As the data with label

"other" were not in large quantities, they have been removed from the dataset. Thus, only two classes (Female and Male) were taken.

4.6.2 Gender grouping (for the first layer)

Using unsupervised method Knowing the number of cluster, KMeans algorithms and Agglomerative cluster were used. Indeed, KMeans is an unsupervised method that aims to divide into k clusters n observations. This was implemented using Python on scikit-learn[5] library. The results obtained by executing KMeans and agglomerative clustering on the weight matrix extracting after epoch 1 and 3 are consigned in Table 4.13.

Cluster	Male	Female
0	1827	88
1	114	684

(A) Epoch 0: KMeans

Cluster	Male	Female
0	1905	277
1	36	495

(B) Epoch 0: Agglomerative clustering

Cluster	Male	Female
0	195	707
1	1746	65

(C) Epoch 2: KMeans

Cluster	Male	Female
0	1868	91
1	73	681

(D) Epoch 2: Agglomerative clustering

TABLE 4.13: Distribution of the data into each cluster (layer 0)

For epoch 0 (Tables 4.13a and 4.13b), whatever the method used (KMeans or agglomerative clustering), the cluster 0 is equivalent to the class Male because most of the data in this cluster are from male class. The KMeans method is more powerful than the agglomerative clustering that groups more data from female class into male class. And this can be verified by the fact that the purity obtained from Kmeans is bigger than the one computed from agglomerative clustering (Table 4.14).
For epoch 2 (Tables 4.13c and 4.13d), KMeans confuses a lot more male and female in the sense that 195 data from the male class were put into cluster 0 that contained female class. The agglomerative clustering is better than Kmeans in term of purity measure when it is applied on epoch 2; But Kmeans is better than agglomerative clustering when it is on epoch 0 (Table 4.14).

		Purity	Entropy
Kmeans	Epoch 0	0.93	0.36
	Epoch 2	0.904	0.40
Agglomerative clustering	Epoch 0	0.88	0.51
	Epoch 2	0.939	0.32

TABLE 4.14: Entropy and purity for gender clustering (layer 0)

4.6.3 Gender grouping using the weights of other layers

In the section 4.6.2, the result obtained by the execution of KMeans and Agglomerative cluster gave a good result on the weight of the first layer (layer 0). This part

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

presents the results obtained by making the same experiment of the other layers (layer 1, layer 2, layer 3 and layer 4). The table 4.15 shows the value of entropy and purity obtained. For layer 1, the purity is high independently of the epoch considered and the method used. That is because the cluster 0 contains most data of male speakers (only 67 females were confused using KMeans) and the cluster 1 contains most of the female speakers (see table 4.16).

Layers	Epochs	Kmeans		Agglomerative clustering	
		Purity	Entropy	Purity	Entropy
layer 1	epoch 0	0.94	0.31	0.94	0.32
	epoch 2	0.92	0.33	0.94	0.32
layer 2	epoch 0	0.72	0.84	0.72	0.85
	epoch 2	0.72	0.85	0.72	0.83
layer 3	epoch 0	0.72	0.85	0.72	0.85
	epoch 2	0.72	0.85	0.72	0.85
layer 4	epoch 0	0.72	0.85	0.72	0.86
	epoch 2	0.72	0.85	0.72	0.85

TABLE 4.15: Gender grouping: entropy and purity of KMeans and Agglomerative clustering computed on the other layers

The entropy and the purity obtained for the other layers (2,3 and 4) are almost the same. The purity measure is 0.72 for all and independently of the method used and the entropy is high (between 0.83 and 0.85). Looking table 4.16, there is a lot of confusions between male and female for layer 2. It explains why the entropy is high. Moreover, layer 1 has good result than layer 0. These observations lead to say that in layer 2, 3 and 4, it is difficult to make difference between male and female, but this is possible for layer 0 and 1.

		Cluster	Male	Female
Layer 1	Kmeans	0	1845	67
		1	36	705
	Agglomerative clustering	Cluster	Male	Female
		0	1860	81
		1	81	691
Layer 2	Kmeans	Cluster	Male	Female
		0	822	206
		1	1119	566
	Agglomerative clustering	Cluster	Male	Female
		0	1062	517
		1	879	255

TABLE 4.16: Distribution of each data into each cluster for layer 1 and 2 (epoch 2)

4.7 Discussion

Speaker clustering and speaker identification By making clustering of speakers, the best results obtained is for layer 0 and epoch 2. And for speaker identification the best result is obtained for layer 0 and epoch 2. These results can be explained by the fact that the more the data are trained, the more the system can learn. Thus, the question is: for how many epochs each speaker based model have to be trained (without overfitting). In speaker identification experiment, the two sub-talks of speaker are chosen randomly, thus it is possible that the sub-talks chosen may influence the final outcome.

Gender identification The results obtained by making KMeans and Agglomerative clustering on weight matrix can be explained by the fact that at the high level in the learning, the system drops meta information about the speech and is focused on the prediction of phonemes. Indeed, the meta information like speakers gender are not useful to predict phonemes. Thus, they can be obtained only at the low level and especially after the first layer.
The paper [Doukhan et al., 2018] presents an open-source framework for gender identification. It used frame-level gender detection and obtained a F-measure of 96.52 using CNN system. In contrast, in this work the gender identification was made on the weight matrix. An idea will be to use the framework implemented in [Doukhan et al., 2018] into the distributed learning with the aim to see how good the framework is for the task.

Chapter 5

Conclusion

The internship was carried out to obtain the master's degree in Natural Language Processing. During the internship some concepts related to deep neural network, speech processing,... have been used. Thus, the internship's subject fits perfectly with the master's program. The work assigned was to implement an acoustic model based on deep neural network in Pytorch, to measure the performance of the model and to analyze the exchanged content in the case of distributed learning. This is what has been done. The work presented in this report have been done by myself under the supervision of Yannick Esteve.
Due to the health crisis linked to Covid-19, the internship was teleworked from March until its end (in August). Thus, it was made without social contact with other members of the LIA. The integration of this new work context into daily life has been done gradually. It developed autonomy in the way of working.

The internship was focused on different points. The first one was to train an acoustic model with Pytorch-Kaldi and the second one was emphasised on the analysis of the exchanged information during a distributed learning. Thus, a model created on Pytorch-Kaldi was trained and a WER score of about 14% was obtained. This model has allowed to simulate a distributed learning by making some speaker based models in which the weights were extracted. By analyzing these weights, observations have resulted. The observation which has been made is that the weights matrix contain information about speakers. Indeed, the clustering method revealed that the speeches of the same speaker were grouped together. Speaker identification experiment and clustering have showed that layer 0 is the layer on which the best results were obtained. In fact, making the clustering shows that 82.62% of the clusters were pure. For speaker identification, an average rank of 17.69 was obtained for layer 0 and epoch 2. Epoch 2 has proved to be the one that gave the best result. Hence, the clustering and the speaker identification depend on the number of epochs on which the model was trained. Regarding gender identification, the results were good for the two first layers and confirm the point that meta information can be obtained only at the low level in the training. The poor result obtained for speaker identification may also means that the weight matrix contain speaker-specific information, such as gender, but are not relevant for identifying the person.

All the results lead to make some deep analysis. Actually, it will be interesting to train the model on more epochs and see how the speakers will be grouped. Another point that lead to make more epochs will be to study the training limits of the speaker based model. Moreover, looking at the importance of the speech content (if there is a lot of pure speech segments) can enhance speaker identification. Furthermore, regards to gender identification, a supervised learning can be done on the data

in order to compare the results with the one obtained for the framework presented in [Doukhan et al., 2018].

The results obtained in this internship was the first start for the LIA in regards to the DEEP-PRIVACY project. The speaker identification and the clustering results open many lines of research that will be pursued by the LIA. One of them concerns the fact that some speakers may have the same manner to speak. Making some analysis on that will allow to reduce the run-time. Indeed, if two speakers have the same way to speak and if there are only data for one of them, a study of the trajectory will yield them to use the same model.

This work will be the subject of a submission to the ICASSP international conference in collaboration with LIA's researchers.

Appendix A

PyTorch-Kaldi Configuration file

```
[cfg_proto]
cfg_proto = proto/global.proto
cfg_proto_chunk = proto/global_chunk.proto

[exp]
cmd =
run_nn_script = run_nn
out_folder = exp/TEDLIUM_GRU_mfcc
seed = 2234
use_cuda = True
multi_gpu = True
save_gpumem = False
n_epochs_tr = 23

[dataset1]
data_name = TEDLIUM_tr
fea = fea_name=mfcc
        fea_lst=../kaldi/egs/tedlium/s5_r3/data/train/feats.scp
        fea_opts=apply-cmvn --
utt2spk=ark:../kaldi/egs/tedlium/s5_r3/data/train/utt2spk
ark:../kaldi/egs/tedlium/s5_r3/data/train/data/cmvn_train.ark ark:-
ark:- | add-deltas --delta-order=2 ark:- ark:- |
        cw_left=0
        cw_right=0

lab = lab_name=lab_cd
        lab_folder=../kaldi/egs/tedlium/s5_r3/exp/tri3_ali
        lab_opts=ali-to-pdf
        lab_count_file=auto
        lab_data_folder=../kaldi/egs/tedlium/s5_r3/data/train/
        lab_graph=../kaldi/egs/tedlium/s5_r3/exp/tri3/graph

        lab_name=lab_mono
        lab_folder=../kaldi/egs/tedlium/s5_r3/exp/tri3_ali
        lab_opts=ali-to-phones --per-frame=true
        lab_count_file=none
        lab_data_folder=../kaldi/egs/tedlium/s5_r3/data/train/
        lab_graph=../kaldi/egs/tedlium/s5_r3/exp/tri3/graph
n_chunks = 5

[dataset2]
data_name = TEDLIUM_dev
fea = fea_name=mfcc
        fea_lst=../kaldi/egs/tedlium/s5_r3/data/dev/feats.scp
        fea_opts=apply-cmvn --
utt2spk=ark:../kaldi/egs/tedlium/s5_r3/data/dev/utt2spk
ark:../kaldi/egs/tedlium/s5_r3/data/dev/data/cmvn_dev.ark ark:-
ark:- | add-deltas --delta-order=2 ark:- ark:- |
        cw_left=0
        cw_right=0

lab = lab_name=lab_cd
        lab_folder=../kaldi/egs/tedlium/s5_r3/exp/tri3_ali_dev
        lab_opts=ali-to-pdf
        lab_count_file=auto
        lab_data_folder=../kaldi/egs/tedlium/s5_r3/data/dev/
        lab_graph=../kaldi/egs/tedlium/s5_r3/exp/tri3/graph

        lab_name=lab_mono
        lab_folder=../kaldi/egs/tedlium/s5_r3/exp/tri3_ali_dev
        lab_opts=ali-to-phones --per-frame=true
        lab_count_file=none
        lab_data_folder=../kaldi/egs/tedlium/s5_r3/data/dev/
        lab_graph=../kaldi/egs/tedlium/s5_r3/exp/tri3/graph
n_chunks = 1

[dataset3]
data_name = TEDLIUM_test
fea = fea_name=mfcc
        fea_lst=../kaldi/egs/tedlium/s5_r3/data/test/feats.scp
        fea_opts=apply-cmvn --
utt2spk=ark:../kaldi/egs/tedlium/s5_r3/data/test/utt2spk
ark:../kaldi/egs/tedlium/s5_r3/data/test/data/cmvn_test.ark ark:-
ark:- | add-deltas --delta-order=2 ark:- ark:- |
        cw_left=0
        cw_right=0


lab = lab_name=lab_cd
        lab_folder=../kaldi/egs/tedlium/s5_r3/exp/tri3_ali_test
        lab_opts=ali-to-pdf
        lab_count_file=auto
        lab_data_folder=../kaldi/egs/tedlium/s5_r3/data/test/
        lab_graph=../kaldi/egs/tedlium/s5_r3/exp/tri3/graph

        lab_name=lab_mono
        lab_folder=../kaldi/egs/tedlium/s5_r3/exp/tri3_ali_test
        lab_opts=ali-to-phones --per-frame=true
        lab_count_file=none
        lab_data_folder=../kaldi/egs/tedlium/s5_r3/data/test/
        lab_graph=../kaldi/egs/tedlium/s5_r3/exp/tri3/graph
n_chunks = 1

[data_use]
train_with = TEDLIUM_tr
valid_with = TEDLIUM_dev
forward_with = TEDLIUM_test

[batches]
batch_size_train = 8
max_seq_length_train = 1000
increase_seq_length_train = True
start_seq_len_train = 100
multply_factor_seq_len_train = 2
batch_size_valid = 8
max_seq_length_valid = 1000

[architecture1]
arch_name = GRU_layers
arch_proto = proto/GRU.proto
arch_library = neural_networks
arch_class = GRU
arch_pretrain_file = none
arch_freeze = False
arch_seq_model = True
gru_lay = 550,550,550,550,550
gru_drop = 0.2,0.2,0.2,0.2,0.2
gru_use_laynorm_inp = False
gru_use_batchnorm_inp = False
gru_use_laynorm = False,False,False,False,False
gru_use_batchnorm = True,True,True,True,True
gru_bidir = True
```

```
gru_act = tanh,tanh,tanh,tanh,tanh
gru_orthinit = True
arch_lr = 0.0004
arch_halving_factor = 0.5
arch_improvement_threshold = 0.001
arch_opt = rmsprop
opt_momentum = 0.0
opt_alpha = 0.95
opt_eps = 1e-8
opt_centered = False
opt_weight_decay = 0.0

[architecture2]
arch_name = MLP_layers
arch_proto = proto/MLP.proto
arch_library = neural_networks
arch_class = MLP
arch_pretrain_file = none
arch_freeze = False
arch_seq_model = False
dnn_lay = N_out_lab_cd
dnn_drop = 0.0
dnn_use_laynorm_inp = False
dnn_use_batchnorm_inp = False
dnn_use_batchnorm = False
dnn_use_laynorm = False
dnn_act = softmax
arch_lr = 0.0004
arch_halving_factor = 0.5
arch_improvement_threshold = 0.001
arch_opt = rmsprop
opt_momentum = 0.0
opt_alpha = 0.95
opt_eps = 1e-8
opt_centered = False
opt_weight_decay = 0.0

[architecture3]
arch_name = MLP_layers2
arch_proto = proto/MLP.proto
arch_library = neural_networks
arch_class = MLP
arch_pretrain_file = none
arch_freeze = False
arch_seq_model = False
dnn_lay = N_out_lab_mono
dnn_drop = 0.0
dnn_use_laynorm_inp = False
dnn_use_batchnorm_inp = False
dnn_use_batchnorm = False
dnn_use_laynorm = False
dnn_act = softmax
arch_lr = 0.0004
arch_halving_factor = 0.5
arch_improvement_threshold = 0.001
arch_opt = rmsprop
opt_momentum = 0.0
opt_alpha = 0.95
opt_eps = 1e-8
opt_centered = False
opt_weight_decay = 0.0

[model]
model_proto = proto/model.proto
model = out_dnn1=compute(GRU_layers,mfcc)
        out_dnn2=compute(MLP_layers,out_dnn1)
        out_dnn3=compute(MLP_layers2,out_dnn1)
        loss_mono=cost_nll(out_dnn3,lab_mono)
        loss_mono_w=mult_constant(loss_mono,1.0)
        loss_cd=cost_nll(out_dnn2,lab_cd)
        loss_final=sum(loss_cd,loss_mono_w)
        err_final=cost_err(out_dnn2,lab_cd)

[forward]
forward_out = out_dnn2
normalize_posteriors = True
normalize_with_counts_from = lab_cd
save_out_file = False
require_decoding = True

[decoding]
decoding_script_folder = kaldi_decoding_scripts/
decoding_script = decode_dnn.sh
decoding_proto = proto/decoding.proto
min_active = 200
max_active = 7000
max_mem = 50000000
beam = 13.0
latbeam = 8.0
acwt = 0.2
max_arcs = -1
skip_scoring = false
scoring_script = local/score.sh
scoring_opts = "--min-lmwt 1 --max-lmwt 10"
norm_vars = False
```

Appendix B

Config file for fintening model

```
[cfg_proto]
cfg_proto = proto/global.proto
cfg_proto_chunk = proto/global_chunk.proto

[exp]
cmd =
run_nn_script = run_nn
out_folder = exp/Test_GRU_fine
seed = 2234
use_cuda = True
multi_gpu = True
save_gpumem = False
n_epochs_tr = 4

[dataset1]
data_name = TEDLIUM_tr
fea = fea_name=mfcc
        fea_lst=../kaldi/egs/tedlium/s5_r3_speaker/data/train/fea
ts.scp
        fea_opts=apply-cmvn --
utt2spk=ark:../kaldi/egs/tedlium/s5_r3_speaker/data/train/utt2spk
ark:../kaldi/egs/tedlium/s5_r3_speaker/data/train/data/cmvn_trai
n.ark ark:- ark:- | add-deltas --delta-order=2 ark:- ark:- |
        cw_left=0
        cw_right=0

lab = lab_name=lab_cd
        lab_folder=../kaldi/egs/tedlium/s5_r3_speaker/exp/tri3_al
i
        lab_opts=ali-to-pdf
        lab_count_file=auto
        lab_data_folder=../kaldi/egs/tedlium/s5_r3_speaker/data/
train/
        lab_graph=../kaldi/egs/tedlium/s5_r3_speaker/exp/tri3/gr
aph


        lab_name=lab_mono
        lab_folder=../kaldi/egs/tedlium/s5_r3_speaker/exp/tri3_al
i
        lab_opts=ali-to-phones --per-frame=true
        lab_count_file=none
        lab_data_folder=../kaldi/egs/tedlium/s5_r3_speaker/data/
train/
        lab_graph=../kaldi/egs/tedlium/s5_r3_speaker/exp/tri3/gr
aph
n_chunks = 1

[data_use]
train_with = TEDLIUM_tr
valid_with = TEDLIUM_tr
forward_with = TEDLIUM_tr

[batches]
batch_size_train = 8
max_seq_length_train = 1000
increase_seq_length_train = True
start_seq_len_train = 100
multply_factor_seq_len_train = 2
batch_size_valid = 8
max_seq_length_valid = 1000

[architecture1]
arch_name = GRU_layers
arch_proto = proto/GRU.proto
arch_library = neural_networks
arch_class = GRU
arch_pretrain_file =
exp/TEDLIUM_GRU_mfcc/exp_files/final_architecture1.pkl
arch_freeze = False
arch_seq_model = True
gru_lay = 550,550,550,550,550
gru_drop = 0.2,0.2,0.2,0.2,0.2
gru_use_laynorm_inp = False
gru_use_batchnorm_inp = False
gru_use_laynorm = False,False,False,False,False
gru_use_batchnorm = True,True,True,True,True
gru_bidir = True
gru_act = tanh,tanh,tanh,tanh,tanh
gru_orthinit = True
arch_lr = 0.0004
arch_halving_factor = 0.5
arch_improvement_threshold = 0.001
arch_opt = rmsprop
opt_momentum = 0.0
opt_alpha = 0.95
opt_eps = 1e-8
opt_centered = False
opt_weight_decay = 0.0

[architecture2]
arch_name = MLP_layers
arch_proto = proto/MLP.proto
arch_library = neural_networks
arch_class = MLP
arch_pretrain_file =
exp/TEDLIUM_GRU_mfcc/exp_files/final_architecture2.pkl
arch_freeze = False
arch_seq_model = False
dnn_lay = N_out_lab_cd
dnn_drop = 0.0
dnn_use_laynorm_inp = False
dnn_use_batchnorm_inp = False
dnn_use_batchnorm = False
dnn_use_laynorm = False
dnn_act = softmax
arch_lr = 0.0004
arch_halving_factor = 0.5
arch_improvement_threshold = 0.001
arch_opt = rmsprop
opt_momentum = 0.0
opt_alpha = 0.95
opt_eps = 1e-8
opt_centered = False
opt_weight_decay = 0.0

[architecture3]
arch_name = MLP_layers2
arch_proto = proto/MLP.proto
arch_library = neural_networks
arch_class = MLP
arch_pretrain_file =
exp/TEDLIUM_GRU_mfcc/exp_files/final_architecture3.pkl
```

```
arch_freeze = False
arch_seq_model = False
dnn_lay = N_out_lab_mono
dnn_drop = 0.0
dnn_use_laynorm_inp = False
dnn_use_batchnorm_inp = False
dnn_use_batchnorm = False
dnn_use_laynorm = False
dnn_act = softmax
arch_lr = 0.0004
arch_halving_factor = 0.5
arch_improvement_threshold = 0.001
arch_opt = rmsprop
opt_momentum = 0.0
opt_alpha = 0.95
opt_eps = 1e-8
opt_centered = False
opt_weight_decay = 0.0

[model]
model_proto = proto/model.proto
model = out_dnn1=compute(GRU_layers,mfcc)
        out_dnn2=compute(MLP_layers,out_dnn1)
        out_dnn3=compute(MLP_layers2,out_dnn1)
        loss_mono=cost_nll(out_dnn3,lab_mono)
        loss_mono_w=mult_constant(loss_mono,1.0)
        loss_cd=cost_nll(out_dnn2,lab_cd)
        loss_final=sum(loss_cd,loss_mono_w)
        err_final=cost_err(out_dnn2,lab_cd)

[forward]
forward_out = out_dnn2
normalize_posteriors = True
normalize_with_counts_from = lab_cd
save_out_file = False
require_decoding = True

[decoding]
decoding_script_folder = kaldi_decoding_scripts/
decoding_script = decode_dnn.sh
decoding_proto = proto/decoding.proto
min_active = 200
max_active = 7000
max_mem = 50000000
beam = 13.0
latbeam = 8.0
acwt = 0.2
max_arcs = -1
skip_scoring = false
scoring_script = local/score.sh
scoring_opts = "--min-lmwt 1 --max-lmwt 10"
norm_vars = False
```

Appendix C

Gender identification: distribution of each data into each cluster (layers 0, 1, 2, 3 and 4)

Layer 0	Kmeans	Cluster	Male	Female
		0	1827	88
		1	114	684
	Agglomerative clustering	Cluster	Male	Female
		0	1905	277
		1	36	495
Layer 1	Kmeans	Cluster	Male	Female
		0	1845	67
		1	96	705
	Agglomerative clustering	Cluster	Male	Female
		0	1860	81
		1	81	691
Layer 2	Kmeans	Cluster	Male	Female
		0	822	206
		1	1119	566
	Agglomerative clustering	Cluster	Male	Female
		0	1062	517
		1	879	255
Layer 3	Kmeans	Cluster	Male	Female
		0	835	231
		1	1106	541
	Agglomerative clustering	Cluster	Male	Female
		0	904	237
		1	1037	535
Layer 4	Kmeans	Cluster	Male	Female
		0	735	192
		1	1206	580
	Agglomerative clustering	Cluster	Male	Female
		0	1236	539
		1	705	233

TABLE C.1: Epoch 0

Layer 0	Kmeans	Cluster	Male	Female
		0	195	707
		1	1746	65
	Agglomerative clustering	Cluster	Male	Female
		0	1868	91
		1	73	681
Layer 1	Kmeans	Cluster	Male	Female
		0	169	732
		1	1772	40
	Agglomerative clustering	Cluster	Male	Female
		0	1862	86
		1	79	686
Layer 2	Kmeans	Cluster	Male	Female
		0	822	209
		1	119	563
	Agglomerative clustering	Cluster	Male	Female
		0	596	193
		1	1045	579
Layer 3	Kmeans	Cluster	Male	Female
		0	838	193
		1	1103	533
	Agglomerative clustering	Cluster	Male	Female
		0	800	217
		1	1141	555
Layer 4	Kmeans	Cluster	Male	Female
		0	845	237
		1	1096	535
	Agglomerative clustering	Cluster	Male	Female
		0	770	206
		1	1096	566

TABLE C.2: Epoch 2

Bibliography

Bell, Peter and Steve Renals (Apr. 2015). "Regularization of context-dependent deep neural networks with context-independent multi-task training". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4290–4294. DOI: 10.1109/ICASSP.2015.7178780.

Bell, Peter, P. Swietojanski, and Steve Renals (2017). "Multitask learning of context-dependent Targets in Deep Neural Network Acoustic Models". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.2, pp. 238–247. DOI: 10.1109/TASLP.2016.2630305.

Bourlard, Herve and Nelson Morgan (1993). *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer.

Cho, Kyunghyun et al. (June 2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: DOI: 10.3115/v1/D14-1179.

Davis, S. P. and P. Mermelstein (1980). *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*.

Doukhan, D. et al. (2018). "An Open-Source Speaker Gender Detection Framework for Monitoring Gender Equality". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5214–5218.

Gales, M.J.F. (1998). "Maximum likelihood linear transformations for HMM-based speech recognition". In: *Computer Speech & Language* 12.2, pp. 75 –98. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1998.0043.

Hernandez, François et al. (2018). "TED-LIUM 3: twice as much data and corpus repartition for experiments on speaker adaptation". In: *CoRR* abs/1805.04699. arXiv: 1805.04699. URL: http://arxiv.org/abs/1805.04699.

Hinton, G. et al. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97. ISSN: 1558-0792. DOI: 10.1109/MSP.2012.2205597.

Jain, L. C. and L. R. Medsker (1999). *Recurrent Neural Networks: Design and Applications*.

Jelinek, F. (1976). "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4, pp. 532–556.

Müllner, Daniel (2011). "Modern hierarchical, agglomerative clustering algorithms". In: *CoRR* abs/1109.2378. URL: http://dblp.uni-trier.de/db/journals/corr/corr1109.html#abs-1109-2378.

Pallett, D. S. (2003). "A look at NIST'S benchmark ASR tests: past, present, and future". In: *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*, pp. 483–488.

Povey, Daniel et al. (Dec. 2011). "The Kaldi Speech Recognition Toolkit". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village Big Island, Hawaii, US: IEEE Signal Processing Society.

Rabiner, L. R. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286.

Ravanelli, M., T. Parcollet, and Y. Bengio (2019). "The PyTorch-Kaldi Speech Recognition Toolkit". In: *In Proc. of ICASSP*. Brisbane, QLD, Australia.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-propagating Errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

Ward, Joe H. (1963). "Hierarchical Grouping to Optimize an Objective Function". In: *Journal of the American Statistical Association* 58.301, pp. 236–244. URL: http://www.jstor.org/stable/2282967.

Zhang, W. et al. (May 2019). "Distributed Deep Learning Strategies for Automatic Speech Recognition". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5706–5710. DOI: 10.1109/ICASSP.2019.8682888.