UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Computer Systems and Computation

Acoustic adaptation of automatic speech recognition systems in educational environments

Master's Thesis

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

AUTHOR: Mas Mollà, Gerard

Tutor: Juan Císcar, Alfonso

Cotutor: Sanchis Navarro, José Alberto

ACADEMIC YEAR: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MASTER'S THESIS

Acoustic Adaptation of Automatic Speech Recognition Systems in educational environments

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging
Academic Course 2022/2023

Gerard Mas Mollà

Advisers:
Dr. Alfons Juan Císcar
Dr. Albert Sanchis Navarro

Abstract / Resum / Resumen

Abstract

Acoustic adaptation of Automatic Speech Recognition (ASR) systems is a field of great interest in different ASR domains and, particularly, in educational environments such as the UPV. In general, the main goal of this task is that of improving the general purpose ASR systems taking into account the specific acoustic conditions of the application domain. In this work, we will review the state-of-the-art in terms of acoustic adaptation of ASR systems and apply the ones that we find more interesting for educational environments and, more precisely, for the UPV media repository.

Resum

L'adaptació acústica de sistemes de reconeixement automàtic de la parla (ASR) és una tasca de gran interés en diversos dominis d'aplicació de l'ASR i, en particular, en entorns educatius com ara el de la pròpia UPV. En general, l'objectiu principal d'aquesta tasca és la millora de sistemes d'ASR de propòsit general tenint en compte particularitats acústiques específiques del domini d'aplicació. En aquest treball es proposa fer una revisió de l'estat de l'art en adaptació acústica de sistemes d'ASR i aplicar les tècniques que es consideren més adequades per a entorns educatius i, en particular, per al repositori UPV mèdia.

Resumen

La adaptación acústica de sistemas de reconocimiento del habla (ASR) es una tarea de gran interés en distintos dominios de aplicación del ASR y, en particular, en entornos educativos como el de la propia UPV. En general, el objetivo principal de esta tarea es el de mejorar sistemas de ASR de proposito general teniendo en cuenta particularidades acústicas específicas del dominio de aplicación. En este trabajo se propone hacer una revisión del estado del arte en adaptación acústica de sistemas de ASR y aplicar las técnicas que se consideren más adecuadas para entornos educativos y, en particular, para el repositorio UPV media.

Contents

Chapter 1
Introduction

This work explores the state-of-the-art techniques for acoustic adaptation of Automatic Speech Recognition (ASR) systems, with the aim of applying the techniques that suit the best the use case of the Universitat Politècnica de València (UPV) media repository in the Spanish language.

In this chapter, we describe the motivation for this work, as well as the context in which it is carried out. In addition, an outline of the document can be found at the end of the chapter.

1.1 Motivation

Automatic Speech Recognition (ASR) applications are well established in the everyday of today's society in the form of personal assistants (Amazon's Alexa[1], Google's Assistant[2], Samsung's Bixby[3], Apple's Siri[4]...), answering machines, medical applications [75, 79, 58, 25] and automatic transcription of media repositories. They usually provide very accurate transcriptions of a given speech audio, although they are highly sensitive to noisy environments and speech peculiarities, as well as very dependant on the tasks they were trained on. There has been a lot of efforts put on mitigating the noise problem and improving environment generalisation of ASR systems [36, 74, 26], but as with most tasks, specific domain-adapted systems provide considerably better results than general-purpose systems.

This Master's Thesis is carried out while working with the *Machine Learning and Language Processing* (MLLP) research group[5]. Since 2014, the MLLP has been providing the Universitat Politècnica de València (UPV) with automatic transcription, translation and dubbing for the UPV Mèdia platform, producing more than 2.7K hours of unsupervised labeled audio only taking into account the poliMèdia videos

[1]https://developer.amazon.com/en-GB/alexa. Last accessed: 11-07-2023
[2]https://assistant.google.com/. Last accessed: 11-07-2023
[3]https://www.samsung.com/us/apps/bixby/. Last accessed: 11-07-2023
[4]https://www.apple.com/siri/. Last accessed: 11-07-2023
[5]https://www.mllp.upv.es. Last accessed: 11-07-2023

(knowledge pills recorded under very specific conditions at the UPV). However, the systems that provide with these services are general-purpose systems built to perform well in all kind of tasks.

Considering this situation, and taking advantage of the fact that all of the poliMèdia videos are recorded under the same conditions (exactly equal rooms with the same professional equipment in each one of them), this Master's Thesis aims to apply state-of-the-art techniques for acoustic adaptation to adapt the acoustic model of the ASR system to the poliMèdia task using unsupervised labeled acoustic data generated with the same model. In addition, we will use the adapted systems to re-adapt them to the speakers in the test set using the same principle. Finally, the experimentation performed on this work aims to open the door to adapting systems under streaming conditions, using the self-generated data to adapt the acoustic model to a task in real time.

1.2 Main objectives

The main objectives of this work are the following:

- To explore and gather the most promising acoustic adaptation techniques and apply them.
- To work with big amounts of unsupervised data and use it as training data.
- To improve the results of the baseline system on the poliMèdia corpus by adapting it with unsupervised data.
- To perform speaker-adaptation with the baseline system to improve the results for each speaker on the poliMèdia corpus.

1.3 Document structure

This document is structured as follows: First, Chapter 2 will introduce the reader to the general concepts of ASR that are necessary before delving deeper into this work, as well as introducing state-of-the-art techniques currently used in ASR applications and acoustic adaptation. Then, Chapter 3 will present the poliMèdia corpus and the unsupervised data, the processing performed and the tasks considered, as well as the tools used to carry out this work. Thereupon, Chapter 4 and Chapter 5 will describe in detail the work developed to adapt the acoustic models to the poliMèdia task and to the test speakers, respectively, as well as their experimental results. Finally, Chapter 6 summarizes the work done, give some concluding remarks and introduce some future lines of work and investigation.

The reader is recommended to read the document sequentially, as it has been written to do so. Each chapter introduces new concepts that will be used in the chapters that succeed. However, if the reader is already experienced with ASR technologies, Chapter 2 can be skipped.

Chapter 2

Automatic Speech Recognition Background

This chapter gives an introduction into the Acoustic Speech Recognition (ASR) field, going through the basics of the technologies that underpin it. It is structured as follows: First, Section 2.1 introduces the reader to the ASR research field, briefly summarizing its history. Second, Section 2.2 goes through the basics of machine learning needed to understand the field of ASR. Then, Section 2.3 introduces neural networks, going from their origins and evolution to state-of-the-art architectures relevant to this work. After that, Section 2.4 puts the concepts explained in the previous sections together to expose the different components of an ASR system. Next, Section 2.5 explores adaptation techniques and studies their applicability to the context of the acoustic adaptation in ASR. Finally, Section 2.6 introduces the evaluation methods of the different components of an ASR system, as well as the evaluation of the whole ASR system put together.

2.1 Introduction

ASR is a pattern recognition task in which, given an acoustic signal as input, extracts the most probable sequence of words corresponding to its transcription. In contrast to human transcription, producing automatic transcripts is cheap and efficient. This opens up a scenario where large technology companies devote large amounts of resources to improving their ASR techniques, making ASR one of the most popular fields within machine learning today.

The first ASR systems, completely analogical, were focused on simple tasks with a small amount of words. An example of these systems can be IBM's Shoebox[1], an ASR system developed in 1962 that was able to recognize numbers from 0 to 9 and math operators such as "plus", "minus" or "total" in order to provide with an speech

[1] `https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html`. Last accessed: 20-06-2023.

interface for an adding machine. Later on, with the introduction of the Hidden Markov Models (HMMs), the ASR systems started to work with probability, allowing them to cover considerably bigger vocabularies. Years after that, with the introduction of new methods and models to estimate the probabilities of the HMMs, as well as the introduction of context-aware models that fitted better to the task, the quality of the automatic transcriptions improved drastically, reaching the level of quality of today's ASR systems, where the automatic transcriptions performed with good conditions are often difficult to distinguish from the human transcriptions [33, 65, 68].

Nowadays, ASR is applied to all kind of environments with acceptable results in terms of quality, allowing us to have automatic transcription services even in a smartphone or tablet. It is applied to automatically transcribe videos in media repositories, to build answering machines, to build easier methods for interacting with machines and even in very delicate environments like medical applications and health care. Here, ASR systems are in charge of automatically transcribe the conversations inside surgery rooms, medical recipes or telematic appointments. However, ASR is also used to assist in speech recognition of people with diseases like dysarthria[2] or Alzheimer [75, 79].

Aside from medical-specific applications, ASR can be used to help deaf people or people with hearing diseases, allowing them to access to media resources by automatically transcribing their content. This case is specially appreciated in educational environments, where automatic transcription allows these people to access to quality education, ensuring everyone's right to education, comprised in the article 26 of the universal declaration of human rights[3]. Furthermore, if we combine ASR applications with machine translation (MT) and text-to-speech (TTS), we can make media resources even more accessible, breaching the language barriers [50, 51].

The field of ASR has a large active community that put a lot of effort on building open toolkits for developing ASR systems such as Kaldi [52] or HTK [77], as well as big open audio datasets like Mozilla's Common Voice [6], LibriSpeech [48], TED-LIUM [55], Europarl-ASR [20] or VoxPopuli [72]. Thanks to these efforts, ASR applications are nowadays easily accessible for everyone.

2.2 Machine learning

Machine learning is a research field of science that uses computer algorithms to automatically discover regularities in data by applying pattern recognition techniques and use these regularities to classify them into different categories [10]. More specifically, it focuses on analyzing the statistical properties of patterns, usually expressed as probability densities [17].

The main goal of a machine learning system is to assign a label c to a given sample. However, in order to be able to correctly analyze real life objects, first they need to be represented with a set of representative features, extracted by the means of an ad

[2]Dysarthia is a disease that affects the motor system, difficulting the patient's ability to pronounce and causing their speech to become more difficult to understand.

[3]https://www.un.org/en/about-us/universal-declaration-of-human-rights. Last accessed: 11-07-2023

hoc feature extraction process and stored in a feature vector $\boldsymbol{x}$. Then, the machine learning system has to estimate the probability function $p(c|\boldsymbol{x})$, that measures the probability of an object represented by the feature vector $\boldsymbol{x}$ belonging to a class c. Usually, and specially when dealing with linear classifiers, the classification function can be computed by using the Bayes' theorem. This theorem relates the probability of an object $\boldsymbol{x}$ belonging to a class c to the probability of c yielding $\boldsymbol{x}$. Thus, applying this theorem, the classification function can be expressed as:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c|\boldsymbol{x}) = \underset{c \in C}{\operatorname{argmax}}\, \frac{p(\boldsymbol{x}|c)p(c)}{p(\boldsymbol{x})} \tag{2.1}$$

Note that $\hat{c}$ refers to the estimated class that yields the maximum value of $p(c|\boldsymbol{x})$.

However, since $p(\boldsymbol{x})$ is constant to c, it can be safely removed from the denominator. It is important to note that, with this modification, we are not estimating $p(c|\boldsymbol{x})$, but a score that keeps the relation with it. This can be done since the aim of a machine learning system is to minimize the classification error and, therefore, its success does not depend on the magnitude of the output scores, but on the ratio between them. With this modification, the final equation for a machine learning classifier can be expressed as:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(\boldsymbol{x}|c)p(c) \tag{2.2}$$

The statistical models used are the result of fitting their parameters through a training process. This training process can be either supervised, if the training samples are labeled (this is, they are associated to a ground truth class), or unsupervised, if they are not, even though machine learning is mostly associated with supervised training processes. In both cases, the goal of the system is to classify unlabeled samples as accurately as possible in the inference process. Figure 2.1 shows a schematic of a machine learning system.

Following the statistical approaches of machine learning, there are many typical binary classifiers for classifying linear data, such as probability distributions [23] and mixtures of probability distributions [22, 31, 70, 38], support vector machines (SVM) [14, 47] or, as used in most modern solutions, neural networks.

2.3 Neural networks

Neural networks is the most popular concept when it comes to artificial intelligence and machine learning, even for non-expert people. This is mostly because of their ability of tackling difficult problems by learning from very big amounts of data. In the current days, there are many variations of neural networks, but they are all based on the original perceptron model.

2.3.1 Perceptron and multi-layer perceptron

The perceptron [54] is a statistical supervised learning linear classifier model, capable of classifying between two classes. It is referred to as a function that maps an input

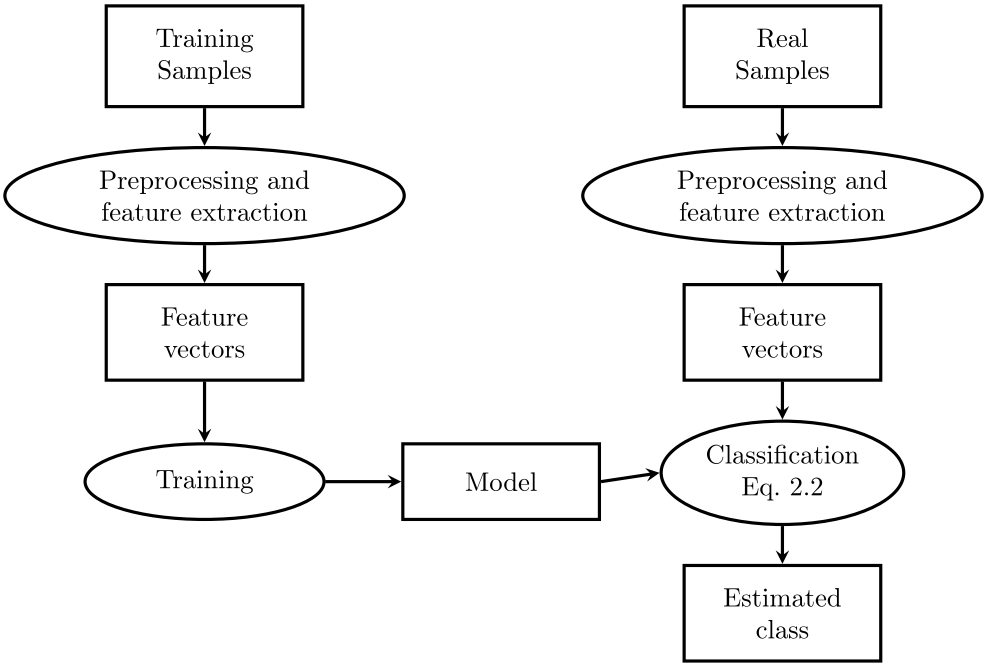


Figure 2.1: Generic pattern recognition system.

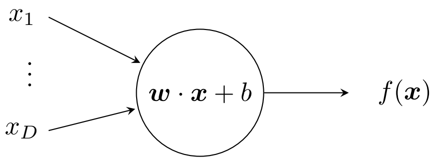


Figure 2.2: Representation of a perceptron as a neuron.

vector $\boldsymbol{x}$ to a binary output value $f(\boldsymbol{x})$ that can be expressed as:

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Where $\boldsymbol{w}$ is a trainable vector of weights, $\boldsymbol{x}$ is the input feature vector and b is a trainable bias parameter.

This model is also often represented relating it with natural neurons, as it can be seen as a cell that retrieves an array of connexions as an input through its dendrites, and outputs the result of some internal processing through its axon terminals. Figure 2.2 shows the representation of the perceptron as a neuron for D dimensional vector classification.

However, the problem of the perceptron is that, as we already mentioned, it is a linear classifier for two-class problems. This is why these models started to grow first in parallel, to add more classes, and then in layers, to generate more complex decision boundaries. The result of this evolution is the multilayer perceptron (MLP)[24], which laid the foundation for modern neural networks. As a result of stacking more units and layers, the equation for a MLP can be expressed as:

$$s_j^l = \sum_{i=0}^{M_{l-1}} \theta_{ji}^l \cdot s_i^{l-1}, 1 \leq j \leq M_l, 1 \leq l \leq L \tag{2.4}$$

Where s_j^l is the output of the j^{th} neuron in layer l, θ_{ji}^l is the weight of the connexion between neurons s_i^{l-1} and s_j^l, M_l is the number of neurons in layer l and L is the total number of layers. Figure 2.3 shows the structure of a generic MLP.

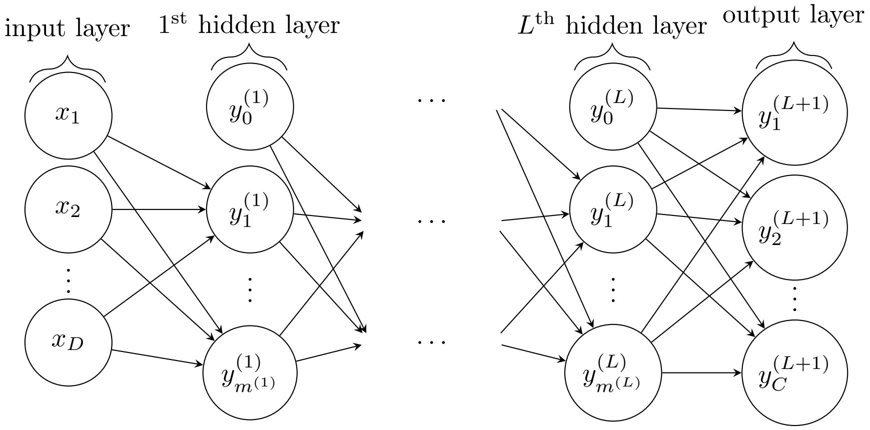


Figure 2.3: Representation of a multilayer perceptron. Layers are connected between them, being the input of the n-th layer the output of the $n-1$-th.

2.3.2 Feed forward neural networks

Even though the MLP allowed the perceptron classifier to tackle multi-class tasks with complex decision boundaries, it is still a linear classifier, as it is a combination of several linear subunits. To solve this problem, the equation of the MLP was modified, adding a non-linear function called "activation function" to the output of each neuron. The resulting equation can be expressed as:

$$s_j^l = g(\sum_{i=0}^{M_{l-1}} \theta_{ji}^l \cdot s_i^{l-1}), 1 \leq j \leq M_l, 1 \leq l \leq L \tag{2.5}$$

Where $g(\cdot)$ is a non-linear function. Examples of non-linear activation functions can be ReLU (Equation 2.7), Sigmoid (Equation 2.8), hyperbolic tangent (Equation 2.9) or Softmax [35]. This modification, with the introduction of the backpropagation training algorithm [56], led to the modern neural networks, allowing them to deal with non-linear decision boundaries and giving them the ability of training with big amounts of data. A neural network without loops, where the information can only flow in one direction, is called Feed-forward network [8] (FFN). This type of network can be used both for regression, where the raw values of the output neurons are taken, and classification, where there is a softmax function that takes the output values and converts them into a unary probability mass. The softmax function is expressed as:

$$\sigma_M(z_n) = \frac{\exp(z_n)}{\sum_{m=0}^{N} \exp(z_m)} \tag{2.6}$$

$$\text{ReLU}(z_i) = max(0, z_i) \tag{2.7}$$

$$\sigma(z_i) = \frac{1}{1 + \exp(-z_i)} \tag{2.8}$$

$$\tanh(z_i) = \frac{\exp(z_i) - \exp(-z_i)}{\exp(z_i) + \exp(-z_i)} \tag{2.9}$$

The FFNs usually are build with several hidden layers (usually more than two). At this point, we can start talking about Deep Learning and Deep Neural Networks (DNN), even though it is not clear where to put the line. A bigger number of hidden layers allows the network to do a deeper feature analysis, since the layers that are closer to the input extract more low-level features, whereas the latter layers can represent more high-level features [78].

2.3.3 Recurrent neural networks

FFNs are great models for classifying independent samples, but they are not designed for using temporal information, even though there are works that use them by directly appending the temporal context to the input [40]. To attack this problem, the recurrent neural networks (RNN) [60] were introduced. In essence, they are FNNs where the input information of every layer in the i^{th} timestep of the sequence is represented by appending the output of the same layer in the $i-1^{th}$ timestep. By this mean, RNNs are able to cope with temporal sequences as input with better results than regular FNNs in temporal-dependent tasks [59]. In the field of ASR, they can be both used for language and acoustic modeling [41, 57]. Figure 2.4 shows the mechanism of an RNN unit.

However, the RNN have a practical limitation when it comes to take into account a very long history of context. Since the memory information is updated in every iteration of the net, it tends to fade out. This phenomenon is known as gradient vanishing [49].

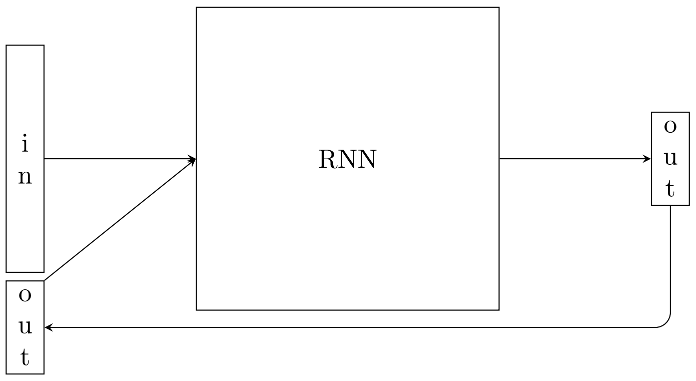


Figure 2.4: Representation on how the mechanism of an RNN cell works.

2.3.4 Long-short term memory models

The Long-Short Term Memory (LSTM) models [27] are a kind of RNN with a more complex memory system. They include three control gates that act as throttling factors to regulate the usage of the context data. First, the input gate controls how important it is to consider the current sample to predict the output. Then, the forget gate is applied to determine the importance of remembering the current sample for future iterations. Finally, there is an output gate that decides how important it is to propagate the cell's information to the following units. These three gates allow the LSTM cell to handle the memory information in a better way than how the RNNs do it, allowing these networks to consider potentially infinite context lengths. Figure 2.5 represents an schematic of the memory mechanism of an LSTM cell.

However, the downside of the LSTMs is that they are sequential models, which forces them to process a signal in a sequential way. This means that dependencies can only be considered in one direction, in addition to having to process the entire signal to predict a single sample. To solve the first problem, the LSTM networks evolved, giving raise to the Bidirectional LSTM (BLSTM) [53] models. The BLSTM models are, in essence, two LSTM networks that analyze the signal one in each direction, solving the problem of the direction of the dependencies.

Figure 2.6 shows the idea of BLSTM bidirectionality. In it, the frames represent time instant and nodes f_i and b_i are the states of the nets in such instant. The f_i nodes are responsible of processing the input data forwards, while b_i nodes process it backwards.

With the introduction of BLSTMs came improvements in many tasks such as MT or ASR, since they allowed to detect references to "future" words. Still, while its bidirectionality brings improvements in temporal-dependent tasks, it also introduces an improvement in the computational cost, since they are two LSTMs in parallel. Aside from that, although these models were introduced to overcome the problem of gradient vanishing, they still struggled to successfully consider large contexts. In order to solve it, the attention-based models where introduced [13].

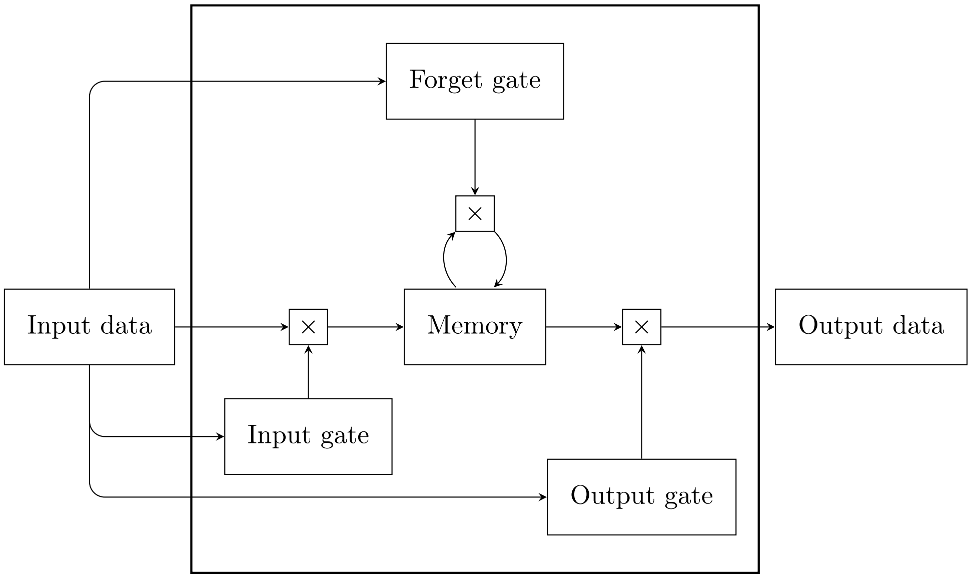
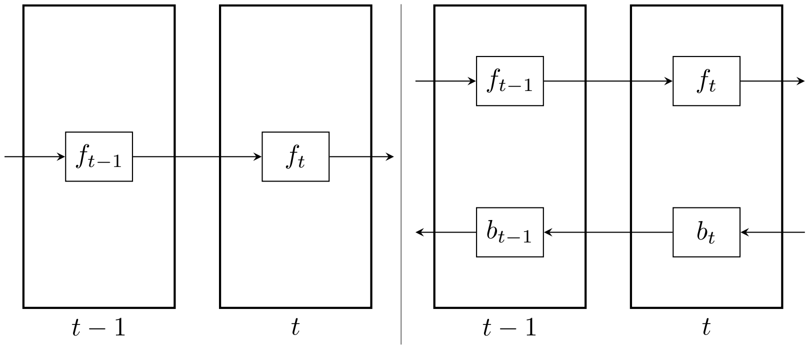


Figure 2.5: Schema of the memory mechanism of an LSTM cell. Nodes labeled with × denote the multiplication of their inputs.



(a) Long short term memory network.

(b) Bidirectional long-short term memory network.

Figure 2.6: Long-short term memory network (left) vs bidirectional long-short term memory network (right).

Attention mechanisms allow to process an entire signal in parallel to capture the most important content for each element in it while keeping a time complexity of $O(n)$ with the length of the signal. When combined with recurrent models, the attention models allowed to capture dependencies in very large contexts, resulting in an improvement in the results without drastically improving the time complexity.

2.3.5 Transformer models

The Transformer architecture [71] was introduced to solve the time-complexity problem of the sequential RNNs by relying on the attention mechanisms. To get rid of sequentially processing the full context, the Transformer architecture introduces a new self-attention mechanism that can relate two elements in the same sequence, computing the dependences between them, in a similar way to how a syntactic analysis is done. This way, the time complexity of the model is drastically reduced, since the self-attention can be computed in $O(1)$.

This architecture is based on a encoder-decoder structure inherited from the MT models, where both parts are formed by blocks consisting on stacked attention modules and FFNs. Here, the encoder maps an input sequence of symbols $\boldsymbol{X}$ to a sequence of continuous representations $\boldsymbol{Z}$ that will be used as input by the decoder. Given $\boldsymbol{Z}$, the decoder can map the sequence to an output symbol sequence $\boldsymbol{Y}$.

Aside from the reduction in time complexity compared to the RNNs, the combination of attention mechanisms granted the Transformers the ability of handling potentially infinite histories, which resulted in even better results, especially with very complex time-dependent tasks [32]. This quickly positioned the Transformers as the state-of-the-art models to analyze sequences, being quickly adopted in a large number of tasks. Nowadays, Transformers are considered the SOTA in ASR applications, being applied in LMs, AMs and even in end-to-end systems [73, 7].

2.4 Automatic speech recognition

Having gone through the basic knowledge to understand the contents of this work, the ASR aims to correctly transcribe an input speech sample by modeling the probability $p(w|\boldsymbol{x})$ of a word sequence w given and input audio feature vector $\boldsymbol{x}$. This probability can be directly estimated using a sequence-to-sequence (seq-to-seq) model, aiming to reduce the complexity of the task by using only one model. However, this approach is hard to use since it requires a significant amount of hours of labelled speech for training the model, restricting its usage only to big companies or economically well-supported research groups. On the other hand, this probability $p(w|\boldsymbol{x})$ can be decomposed by using Bayes' theorem. With this, the probability can be simplified into to main components:

$$\hat{w} = \underset{w \in L^*}{\operatorname{argmax}}\, p(w|\boldsymbol{x}) = \underset{w \in L^*}{\operatorname{argmax}}\, p(\boldsymbol{x}|w)p(w) \tag{2.10}$$

In this new equation, we can distinguish two different probabilities: the probability $p(\boldsymbol{x}|w)$ of an acoustic feature vector $\boldsymbol{x}$ given a word sequence w, which can be thought of as the probability of w being represented with $\boldsymbol{x}$, and the probability $p(w)$ of a word sequence w in a language. The first one can be modeled with an acoustic model (AM), whereas the latter can be estimated with a language model (LM), combining both of them in a decoding step. Since there are two different models that are combined to decode the best sequence of words w, this approach is called hybrid approach.

This approach allows to exploit more abundant data sources since, although the acoustic model is still limited to training with labelled acoustic data, the language model can be trained with monolingual text data. Hence, it allows to build robust and powerful LMs that can significantly expedite raw ASR performance. However, since the nature of each of the components is different, this approach presents some problems that need to be fixed. On one side, each of the components work in a different probabilistic scale, which causes that, sometimes, the contribution of the LM becomes irrelevant for the decoding process. This is why an scaling factor α that scales the probability yielded by the LM is introduced, as represented in equation 2.11. On the other side, and since both models have been trained separately, it is difficult to achieve a final system as robust as an end-to-end model. This problem makes the decoding step very important and, thus, an exhaustive exploration of the decoding parameters is needed in order to find an optimal combination.

$$\hat{w} = \underset{w \in L^*}{\operatorname{argmax}}\, p(\boldsymbol{x}|w)p(w)^{\alpha} \tag{2.11}$$

2.4.1 Language model

AS mentioned, the LM is in charge of modelling the probability $p(w)$ of a sequence of words w in a language L. To do so, it aims to represent the structure of the language L by computing the probability of every word sequence w in the language L, which is usually approximated with the probability $p(w_i|w_j...w_{i-1})$ of a word w_i given its context $w_j...w_{i-1}$ where $j < i$ as follows:

$$p(w) \approx \prod_{i=1}^{I+1} p(w_i|w_{max(i-n+1,0)}^{i-1}) \tag{2.12}$$

Where I is the length of w, n is the maximum history length considered, w_j^{j+n} is the sequence of words $(w_j...w_{j+n}) \in w$ and w_0 and w_{I+1} represent the special tokens for the start and end of the sentence, respectively. Since the LM aims to build a probability distribution on every word sequence $w \in L^*$, all probabilities are subject to:

$$\sum_{w \in L^*} P(w) = 1 \tag{2.13}$$

One of the most common approaches to language modeling are n-gram models. In ASR, n-grams are contiguous sequences of n words, usually referred to by its length (unigrams, bigrams...). N-grams approximate $p(w)$ as the probability $P(w_i|w_{i-n+1}^{i-1})$ of a word w_i given an n-sized word sequence w_{i-n+1}^{i-1} as context. They are easy-to-build LMs, since the probabilities are empirically estimated by counting the number of occurrences of each n-gram in a training corpus and normalizing it to the number of occurrences of its history [21]. This can be formalized as:

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{N(w_{i-n+1}^{i})}{N(w_{i-n+1}^{i-1})} \tag{2.14}$$

Where $N(w)$ is the amount of times a word sequence w appears in the training data. This method is very useful for estimating the probability of sequences seen in the training corpus, but it is unable to assign a non-zero probability to a sequence that was not seen in the past. To cope with these kind of sequences, there are smoothing techniques consisting on interpolating different models [18, 34] or re-distributing the probabilities assigned to lesser seen sequences [19, 11], among others [37].

In more recent works, neural approaches to language modeling have been introduced [9]. They were mostly based on RNNs [41], more precisely LSTMs with attention mechanisms [12, 39], since they are able to consider sequential data dependencies providing good results. Nowadays, the Transformer-based models are capable of building powerful LMs, as they are able to process long sequences in parallel, aside from their generalisation capabilities [73, 7]. As a product of this, they are the state-of-the-art and the go-to architecture when building LMs.

In terms of the data processing for language modeling, there are different approaches, mainly depending on the expected output of the system. Essentially, no characters other than those belonging to words of the language must be left on the corpus. In the case of Spanish, for example, all characters not contained in the Latin alphabet should be removed. It is also interesting to re-write all the characters using the same encoding format (i.e. UTF-8).

2.4.2 Acoustic model

The acoustic model, in charge of modeling $P(\boldsymbol{x}|w)$, is the component that relates the acoustic signal with the phonetic units in the hybrid approach. It is trained with labelled acoustic data, either by humans or previous ASR systems, and is usually modeled with Hidden Markov models (HMMs), since these are specially good at modelling time-dependant tasks, in combination with some other classification system that models the transition probabilities between the hidden states. Formally, we can define an HMM M as follows:

$$M = (Q, \Sigma, \pi, A, B) \tag{2.15}$$

Where Q is a set of finite states, Σ is a finite set of symbols or alphabet, π is an initial probability vector, A is a matrix with the probabilities of transitioning from a certain state q_i to another state q_{i+1} with $q_i, q_{i+1} \in Q$, and B is a matrix with the probabilities of emitting a symbol $x_t \in \Sigma$ from a certain state $q_t \in Q$ in a timestep t. Figure 2.7 contains a representation of a three-state hidden Markov model. These kind of models are used to model Markovian processes, which are random time-dependant processes, like the human speech. More concretely, in the ASR context, the HMMs are used to model acoustic phonemes or triphonemes (phonemes with past and future context). With this idea in mind, the probability of an HMM that models a phoneme f generates an acoustic sequence $\boldsymbol{x}$ can be expressed as:

$$P_f(\boldsymbol{x}) = \sum_{\boldsymbol{q} \in Q^*} \prod_{t=1}^{|\boldsymbol{q}|} P(q_t|q_{t-1}) P(x_t|q_t) \tag{2.16}$$

Where $\boldsymbol{q}$ is a state sequence with t timesteps, x_t is the vector in position t at the acoustic vector sequence $\boldsymbol{x}$, $P(q_t|q_{t-1})$ is the transition probability $A_{(q_{t-1},q_t)}$ between states q_{t-1} and q_t, and $P(x_t|q_t)$ is the emission probability $B_{(q_t,x_t)}$ of emitting x_t in state q_t.

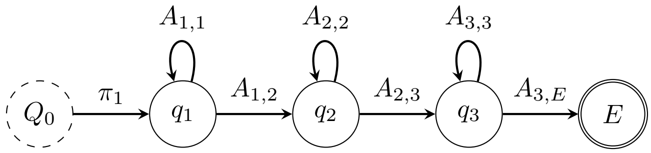


Figure 2.7: Representation of an HMM. Q_0 represents the initial state, with an initial transition probability contained in π. q_1, q_2 and q_3 are states in Q, with transitions between them whose probabilities are contained in A. E is the final state of the model.

As mentioned, the emission probabilities are usually modelled with external models, traditionally being these Gaussian mixtures (GMMs) trained with an Expectation-Maximization (EM) algorithm. This combination is usually referred to as GMM-HMMs. Then, these model transitioned to using FFNs, convolutional NNs or other types of neural networks, since they are more powerful in generalisation than GMM. However, more modern approaches have used RNNs (more concretely, BLSTMs) for estimating the transition probabilities, since human speech has temporal dependencies to be considered. Nowadays, there are already acoustic models based on the Transformer architecture but, unlike in other tasks, they are not yet established as a better alternative to the BLSTMs.

In terms of the treatment process of the data, the traditional process for obtaining the feature vectors consists on applying a set of filters to the acoustic signal through its frequency spectrum. This set of filters can vary depending on the interests of the task, but it is usually either a set of Mel frequency cepstral coeficients (MFCC) or a set of filterbanks. Figure 2.8 represents how the filters are applied through the frequency domain of the input signal.

2.4.3 Hybrid decoding

In the hybrid approach to ASR, the decoding step is the process where all the components are put together to extract a sequence of words given a sequence of input acoustic vectors. There are different approacches to hybrid decoding, being Weighted Finite State Transducers decoders [43] (WFST) and History-Conditioned Search based decoders [46] (HCS) two of the most popular ones. In this work, we are going to focus on the HCS decoders.

In a decoding process, a graph containing all the HMM states is unfolded over time, updating the transition probabilities between them by using both the AM and the LM. The transition in the decoder graph can be both inner transitions in the HMMs or transitions between different HMMs, called across-word transitions. With this idea in mind, the AM is queried to update the probabilities between the inner transitions inside a single HMM, while the LM is queried to update the probabilities

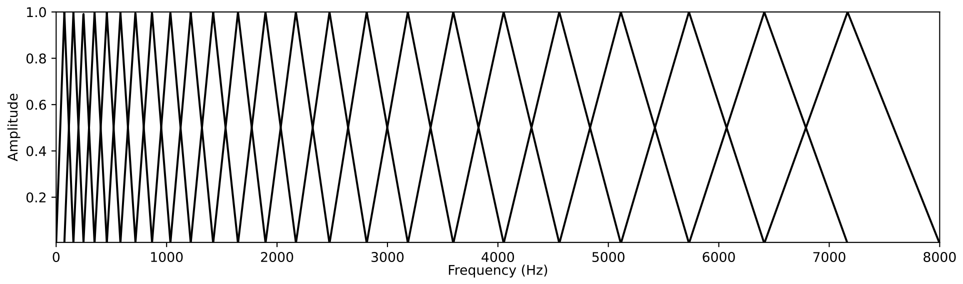


Figure 2.8: Representation of the structure of a bank of filters that can be used for extracting acoustic feature vectors. The filters are applied through the frequency domain, obtaining a feature vector of n components where n is the number of filters in the bank.

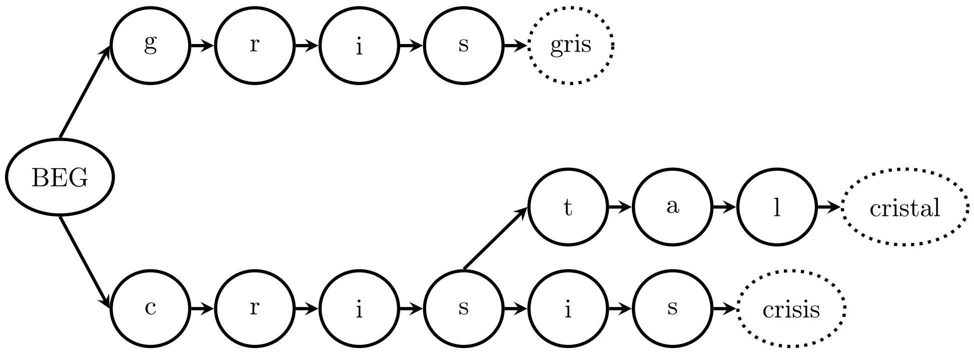


Figure 2.9: Simplified decoder for single word recognition. A simplified language {"gris", "crisis", "cristal"} is considered.

of the across-word transitions. If we simplify this problem to single word recognition, we can see the HMM graph as a prefix tree where each state represent a phoneme, as seen in Figure 2.9. In a same manner, if we abstract the decoder to word-nodes, the whole decoding graph can be seen as a concatenation of prefix trees via across-word connections. Figure 2.10 zooms out the representation of Figure 2.9 to show this idea adapted to multi-word recognition.

Since this process has to deal with a large graph concatenating the whole vocabulary every time a word node is reached, the latency of the system can be compromised, denying the streaming applications of hybrid systems. Lots of efforts were made to overcome this problem, which led to the conclusion that unnecessary queries to the LM should be avoided, at it is the slowest component of the system. Following this solution path, and to avoid unnecessary queries when performing look-ahead operations [45], the usage of look-ahead tables were introduced [29]. These look-ahead tables are usually build from a simplified version of the big LM, usually via a super-aggressive pruning process or a different m-gram model where $m < n$. Since their

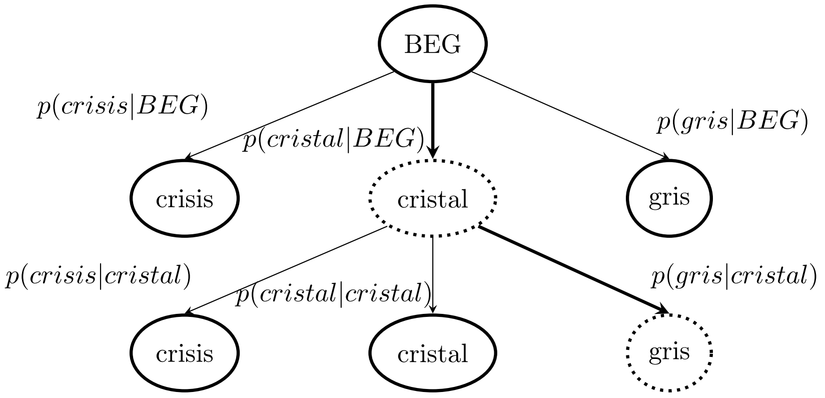


Figure 2.10: Hybrid decoder for multiple word recognition. The language considered is the same as in Figure 2.9 and the sequence recognized was "cristal gris". Probabilities consider a bigram language model and denote the language model factor for the words in each node

size is significantly smaller than the big LM, their speed is also drastically higher, although they are not as precise. This modification allows to drastically reduce the amount of queries to the big LM, being the latter only queries when a word-node is reached. With this, aside from other upgrades like variance regularisation [61], lazy evaluation and language model history recombination [29], the decoding complexity is reduced, allowing it to perform even in streaming environments.

2.5 Acoustic adaptation

The acoustic adaptation is a technique that aims to obtain better results overall in an ASR task by slightly modifying the acoustic model according to the tasks peculiarities. However, domain adaptation is a general concept in machine learning, that also aims to do so. It is defined as the process of adapting a model trained with data in a certain domain $\mathcal{X}_s$ and a certain set of labels $\mathcal{Y}$ to a different domain $\mathcal{X}_t$ that shares the same set of labels $\mathcal{Y}$ [44]. In this definition, $\mathcal{X}_s$ can be a domain formed with real images of cats and dogs, $\mathcal{X}_t$ is a domain conformed by artistic paintings of the same animals, and $\mathcal{Y}$ is a set of labels {*"cat", "dog"*}.

Domain adaptation is comprised in the field of transfer learning, which aims to take advantage of the similarity between data of different tasks to obtain better results in tasks with fewer labeled data. This situation opens the door to tackling the domain adaptation problem with different methods, such as fine-tuning the model, adding adapter layers or using self-supervised learning.

The fine-tuning approach is usually carried out by performing a second training process on the model, albeit with a significantly lower learning rate. This allows the model to learn from the new data and "move" its knowledge to perform better on the new task, although usually it loses performance on its original domain as a

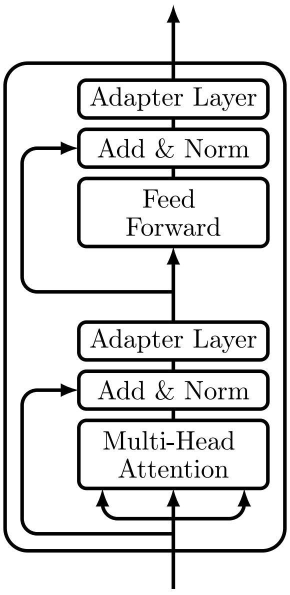


Figure 2.11: Schematic on the aplication of the adapter layers to a transformer encoder block.

consequence. The key factor on this process is the adjustment of the leaning rate, since a higher learning rate will cause the model to forget its original task. This approach is vastly used when using big pre-trained models to tackle a task with fewer data, such as using a large CNN model like a DenseNet that was trained with the ImageNet corpus to analyze images of paving and classify them in whether or not it is damaged.

Another popular technique used for taking advantage of pre-trained models is the addition of adapter layers to the model. This approach is mostly used when both the source domain and the target domain are somehow similar. In this approach, we can modify a model with parameters $\boldsymbol{\Theta} = (\boldsymbol{\theta_1}, \boldsymbol{\theta_2})$ by adding a third operation in the middle with a set of parameters $\boldsymbol{\theta_3}$ so that $\boldsymbol{\Theta'} = (\boldsymbol{\theta_1}, \boldsymbol{\theta_3}, \boldsymbol{\theta_2})$. Now, we can fine-tune the original model, but freezing the original parameters $(\boldsymbol{\theta_1}, \boldsymbol{\theta_2})$, so that the only set of parameters that will be modified is $\boldsymbol{\theta_3}$. The aim on this technique is to change the behaviour of the original model, but allowing the original set of parameters to remain untouched. Usually the added parameters are linear, assuming that the difference between the two domains can be linearly computed and allowing a faster convergence. However, and since this approach is usually utilized with big transformer-based models with a large number of parameters, it usually yields a bigger and slower model, which can be harmful for application domains where the system latency is important.

Finally, the other major adaptation technique is the self-training or pseudo-labeling. The basic idea of this technique is “to use the model itself to infer predictions on unlabeled data, and then treat these predictions as labels for subsequent training" [44]. Self-training is commonly applied in to manners. In the first way, the pseudo-labels are predicted for the entire collection of data to, then, train the model in combination with the labelled data. In the second way, the model is used to label a subset of the unlabeled data that is used to immediately train the model. The first method is mostly useful when dealing with enormous collections of data [76], whereas the second is often used in more sophisticated contexts [64].

In the field of ASR, where it is very easy to obtain large amounts of unlabeled data, this last technique is specially interesting. Furthermore, if we consider that, in hybrid systems, the process of obtaining the output transcription (and, therefore, the pseudo-labels) involves both the AM (that is to be adapted) and a powerful LM that has a lot of knowledge on the structure of the language, we can consider that this process takes advantage of the knowledge of the LM to correct the behaviour of the AM.

However, and since it will be used later in this work, we are going to devote a paragraph to L2 regularisation and a method called empirical Bayes. Empirical Bayes is a set of procedures widely used in variational methods in Deep Learning to estimate the weights of a model where latent variable spaces are high-dimensional. However, in the specific case of this work, where we need to adapt an existing model to a new domain, it can be interesting to apply this technique, as it allows to perform a training process were the new model receives a penalty based on the distance between its weights and the previous model's that can be computed using the L2 regularisation, among other distance-based regularisations. The L2 is a measure of distance that can be easily expressed as $d(p,q) = \sqrt{\sum_{n=0}^{N}(p_n - q_n)^2}$, where p and q are points in a N-dimensional euclidean space. Putting everything together, the loss function that would be applied using this technique is:

$$\text{L2 Loss}(w^i) = \text{Loss}(w^i) + \text{L2 penalty}(w^i) = \text{Loss}(w^i) + \sum_{n=0}^{N}(w_n^i - w_n^{i-1})^2 \quad (2.17)$$

Where w are the weights of the model, i is the iteration of the training process and N is the dimension of the model. This technique allows to train the base model with a lower risk of overfitting it, which is the usual problem of the finetuning methods.

2.6 ASR Evaluation

There are several metrics available to evaluate the performance of an ASR system and its components. Trained ASR systems can be evaluated using the word error rate (WER) on supervised labelled data. The WER is defined as the minimum edition distance between the reference and the infered transcription. It is similar to the Levenshtein distance of edition, allowing insertion, deletion and substitution of words. It is computed with the following formula:

$$WER = \frac{I + D + S}{|R|} \tag{2.18}$$

Where I, D and S are the total count of insertions, deletions and substitutions, respectively, and $|R|$ is the total length of the reference transcription. However, aside from the WER, the individual components of an ASR can be evaluated individually by using the FER, for the AM, and the PPL, for the LM.

On one side, the perplexity (PPL) is the metric to minimize in the training process of a language model. The PPL of a given sequence of words w can be formally expressed as:

$$PPL(w) = 2^{-\frac{1}{n}\log p(w)} \tag{2.19}$$

Where $-\frac{1}{n}\log p(w)$ is an estimation of the cross-entropy for w when it is sufficiently long. It can also been intuitively expressed as the average number of path that can follow a given word in a sequence.

On the other side, the acoustic model can be evaluated by using the frame error ratio (FER), which can be easily defined as:

$$FER = \frac{F_{wrong}}{F_{total}} \tag{2.20}$$

Where F_{wrong} is the number of uncorrectly classified frames, while F_{total} is the total number of frames in the acoustic sequence.

Chapter 3

MediaUPV dataset, tasks and tools

In this chapter, the poliMèdia dataset is described, exposing the MediaUPV repository, the original poliMèdia dataset and the automatic transcriptions used in this work as unsupervised training data. The two tasks considered for this work are explained, as well as the data processing steps performed. Aside from this, the tools used in this work are exposed.

The chapter is structured as follows: Section 3.1 exposes the MediaUPV repository, the poliMèdia dataset and the unsupervised data that was automatically generated by the baseline system. After that, Section 3.2 lists the tools used to develop this work. Then, Section 3.3 details the processing steps performed to clean the data and prepare it for its usage. Finally, Section 3.4 explains the two tasks that were considered for this work.

3.1 PoliMèdia dataset description

The MediaUPV repository is a professional UPV service for creation, storage and diffusion of educational videos [67, 2]. It was launched in 2007, firstly thought of as a platform where UPV lecturers could produce high-quality short videos at dedicated UPV studios, with the aim of supporting blended learning through prerecorded "knowledge pills", usually referred to as *poliMedias*. With the time, the platform also became the main video service for the UPV to provide MOOCs [4, 3], helping the university to become one of the most renowned MOOC providers in Spanish, with more than 85 MOOCs and 290 editions already completed, more than 2.3 enrolments, and two of the 100 most popular online courses of all time [1]. Apart from poliMedias, MediaUPV has been expanded to include self-produced videos by students and lecturers, known as *poliTubes*. Finally, since joining the Opencast consortium in 2011, UPV has deployed lecture capture technology to 84 locations, retrieving more than 600h per year and uploading them to MediaUPV for their distribution to students only [66].

Although MediaUPV comprises diverse kinds of educational material, this work will focus only on PoliMedias due to their simplicity in terms of duration, speakers and audio quality. PoliMedias are produced at dedicated and standarized video-recording studios at the UPV. They are 4x4 metre rooms equipped with a white backdrop, video camera, capture station, pocket microphone, lighting and AV equipment including a video mixer and an audio noise gate. To record a poliMèdia, the lecturers come to a poliMèdia studio with their slides prepared after making an appointment via a web service. Once in the studio, the system automatically embeds the video of the lecturer with their slides into the video output. Then, with review and approval of the lecturer, the resulting video is uploaded as a poliMèdia to MediaUPV.

The number of videos uploaded to MediaUPV has been increasing steadily since 2007, with more than 97000 videos in June 2023. As with normal face-to-face lectures, most of the poliMedias are produced in Spanish, followed, although not very closely, by Catalan and English as shown in Table 3.1. However, there are already initiatives that focus on matching these numbers, as Catalan is a co-official language at the UPV and English is important for its global competitiveness, aside from the fact that promoting multilingualism means that all supported languages must be treated equally with regard to available resources.

Table 3.1: PoliMedia lecturers for Spanish (Es), Catalan (Ca) and English (En).

	Es	Ca	En
Number	23192	623	2869
Duration (h)	4167	74	359
Total (%)	86.9	2.3	10.8

The MediaUPV repository is constantly evolving in terms of size and complexity, being this the reason why the sub-set constituted by the poliMedias has been chosen as a case study in many EU projects. Aside from that, since the transLectures project [62], there has been poliMèdia-adapted ASR/MT systems that transcribed and translated content to make it available in all the supported languages and enrich the platform with raw multilingual subtitles. At first, since the systems were not as accurated as modern systems, the subtitles required post-editing. With this idea in mind, a user-friendly tool for reviewing was integrated into the production workflow, allowing both the author and non-author users to easily review and correct the captions and translations [69, 42, 63, 15]. However, with more modern systems, there is no longer the need of reviewing the automatic subtitles, being them often directly fit for non-supervised publication.

This initiative of automatically transcribing and translating poliMedias has generated a large corpus of unsupervised transcribed audio that can be used for training or adapting systems. More precisely, since the publication of the poliMèdia corpus (114.46 hours in Spanish), around 2.6K hours of unsupervised audio were created only counting the poliMedias.

For this work, we focused in the poliMèdia corpus, being this a subset of the videos in the MediaUPV repository constituted only by poliMedias, as they are all produced in a standarized manner, opening the possibility for task-adapt experiments. More precisely, and since MLLP's ASR systems are mostly monolingual, we decided to work on the Spanish partition of the corpus, since it has the most hours among the different language partitions. With this, we are left with 2.732 hours of unsupervised audio across 17K videos to be processed.

3.2 Tools used

This section details the main tools used in order to develop the final acoustic models: the transLectures-UPV toolkit (TLK) and TensorFlow.

3.2.1 TLK

TLK [16] is a state-of-the-art toolkit developed by the *MLLP-VRAIN* research group at the Universitat Politècnica de València (UPV), under the context of the transLectures European project [62]. It is a toolkit that aims to perform every action in the acoustic pipeline, aside from implementing a one-pass decoder capable of transcribing speech in streaming. It is the toolkit used for building the winner ASR system for the Albayzin-RTVE Speech-To-Text Challenge on their 2018 and 2020 editions [30, 28]. In this work, TLK was used for preprocessing the acoustic data, aligning the transcriptions with the audio, training the model and recognising. The following tools were used:

- `tLtask-preprocess`: it is a feature extraction tool. It takes wav files as an input, preprocesses them and outputs their feature vectors, that can be either MFCC or filter bank vectors.

- `tLtask-train`: it takes the feature vectors generated with `tLtask-preprocess` to train acoustic models. It is able to work with monophoneme and triphone models, as well as convert triphoneme models into tied phone models. It uses the Baum-Welch and the Viterbi algorithms to estimate parameters. It can be used to train a new model or to adapt an existing one.

- `tLtask-recognise`: it is the tools that implements the hybrid decoder. It takes acoustic feature vectors and obtains the most probable hypothesis using the previously trained AM and LMs.

3.2.2 TensorFlow

TensorFlow [5] is an open-source toolkit developed by the Google Research team. It aims to provide a friendly environment for creating and training machine learning models. It uses dataflow graphs to represent the states of the models and the computation flow. In this work, TensorFlow is internally used by `tLtask-train` to build and train the BLSTM used in the BLSTM-HMM acoustic model. In order to be able to work with TensorFlow with acoustic feature vectors, they need to be converted to a representation that the toolkit can use. For this, the feature vectors were converted to TFRecords, a TensorFlow format used to represent sequential samples that can be easily processed by the BLSTM.

3.3 Data processing

As mentioned in Section 2.4, there are two main treatments in the traditional ASR data preprocessing: the acoustic data preprocess and the text data preprocess. In this work, since it is based on adapting an existing model, the data needed to be preprocessed following the same directions as in the training of the baseline model.

On the one hand, the text data was processed to be left with only lower-cased unicode characters from the Latin alphabet following the following steps. First, it is needed to tokenize the text in order to obtain minimum text units to work with. This process can be done by sepparating the text by literal words, treating every word in the original corpus as a token. However, this method comes with a catch, as the words that do not appear in the training process won't exist in the final vocabulary. To overcom this problem, there are techniques like byte-pair encoding (BPE) or Sentencepiece, which break down words by their stem and modifications, allowing to obtain even smaller units that may help to understand words that do not appear in the training set. However, since the original model was trained with the traditional tokenization, the tokenization in this work used the first method. After this, the text needed to be normalized by expanding the abbreviations (i.e. sr. → señor, sra. → señora...) and rewriting the numbers with their corresponding words (i.e. 3 → three). Finally, a cleaning step was performed to remove every non-alphabetical word left in the corpus, trim the text and remove empty lines to obtain a clean dataset. To perform these steps, we used the `awk` language, the `sed` utility and the `num2words` Python library with a few modifications, which already provides support for several languages in its base version, including Spanish, among other less used Unix utilities.

On the other hand, the acoustic data was obtained by first converting the poliMèdia videos to 16 bit little endian `wav` files in mono mode and a sampling frequency of 16kHz. After that, the acoustic feature vectors were generated by sampling the audio signal in windows of 50 ms with a 10 ms overlapping between them. Aside from that, the windows are shaped with a Hamming process, consisting on aplying the next formula:

$$w(n) = \begin{cases} 0.54 - 0.46\cos\frac{2\pi n}{N}, & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Where N is the length of the window (50 ms, in this case). This treatment allows to smooth both ends of the window, as taking the raw shape of the window cuts the audio too abruptly. Next, the windows are processed using a fast Fourier Transform (FFT) algorithm to compute the discrete Fourier transform (DFT), that is defined as:

$$X_n = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi i}{N}kn} \quad \forall k \in 0, ..., N-1 \tag{3.2}$$

Where i is the unitary imaginary number. Following this, we applied a set of filterbanks (Section 2.4) with 85 components, obtaining the final set of acoustic vectors that will be fed to the BLSTM. All this steps were performed by using the `tLtask-preprocess` tool.

3.4 Tasks

From this dataset we extracted two tasks: adaptation to the poliMèdia environment, where we adapt the baseline model to perform better in the poliMèdia task, and speaker-adaptation, where we adapt the model to the particularities of each speaker of the test set. Considering this, and in order to avoid interferences between both tasks, we extracted from the first task's dataset every video with a speaker that appears in the test set.

3.4.1 Task-adaptation

The first task considered for this work was to adapt the model to the poliMèdia environment. For this, we took all the non-supervised poliMedias automatically transcribed by the baseline system in a production setup to train the same model. As mentioned in section (self-learning), this idea aims to take advantage of the robustness of the LM to adapt the AM to adverse conditions. This is because in the decoding process, the LM uses its knowledge on the structure of the language to correct the words that were misheard by the AM, resulting in a sort of semi-supervised labels that can be re-used for training. After having removed the videos from the test speakers and removing problematic videos (without audio, text or with any kind of anomally), the training set for this task was left with 2634 hours of unlabeled audio data.

For the development and test sets, we used the actual development and test partitions of the original poliMèdia corpus. By doing this, we can measure the impact of the training process over unsupervised data over the real poliMèdia task.

3.4.2 Speaker-adaptation

The second task for this work is to adapt the model to the speakers in the test set of the original poliMèdia corpus. The approach for this task was to adapt the baseline model, generating a new model for each speaker, where each model is adapted to the particularities of every speaker's speech.

Initially, most of the poliMedias created by the speakers of the original test set had their transcriptions supervised by humans, which left us with roughly 1.3 hours of unsupervised audio data and leaving 4 out of the 5 speakers without unsupervised data to adapt. However, we were able to acces the unsupervised version of these videos after a few weeks, which left us with 4 hours shared between 4 out of the 5 test speakers, with a representative amount of hours for 3 of them. Table 3.2 shows the distribution of these videos among the 5 speakers in the test split with and without the supervised version of their videos.

Table 3.2: Distribution of the unsupervised data among the test speakers.

Speaker	Original data (h)	Unlocked non-sup. (h)
Sp. 1	0.13	1.14
Sp. 2	1.17	1.30
Sp. 3	0.00	1.63
Sp. 4	0.00	0.18
Sp. 5	0.00	0.00
Total	2.30	4.25

In this case, we devoted a video from each speaker for development purposes. Since this task could be used over the time, adapting a model with every inference process, we decided to select the last video from each speaker, simulating a case where the model used the previous videos from every speaker to improve the results on the following one. On the other hand, we used the poliMèdia test partition for testing, evaluating each speaker with their own videos from the original test.

Chapter 4
Self-supervised acoustic adaptation

This chapter exposes the work developed over the first task proposed in Section 3.4. It is structured as follows: First, Section 4.1 exposes the baseline system and its results on the poliMèdia task. Then, Section 4.2 describes in detail all the steps performed to adapt the acoustic models for, in Section 4.3 integrating them into the final system and evaluate their performance. Finally, Section 4.4 presents the results obtained with our proposed systems and compares them with the baseline results.

4.1 Baseline system

As mentioned in 3.4, this task has as a goal to adapt MLLP's current production ASR system in Spanish to improve its results on the poliMèdia task by using self-generated unsupervised data. This baseline system consists on a one-pass decoder [29] with a BLSTM-HMM acoustic model and an interpolation of a Transformer LM (TLM) and an n-gram model as a language model. This system features various tweaks presented in [7] that allows it to recognise in streaming environments without losing performance compared to its offline version [28]. More concretely, the LM interpolation combines a 4-gram LM with 55K entries and a TLM based on a transformer decoder with 24 transformer layers, 12 attention heads, 4096 unit-FFNs and a model dimension of 768. In the case of the AM, which is the model that is going to be adapted in this work, it is a BLSTM with 8 layers and 1024 units per layer, an output layer of 10041 units with a previous bottleneck layer of 200 units. It was initially trained with 85 component filterbank acoustic vectors and a total amount of arround 3900 hours of audio retrieved from different datasets. With all this, Table 4.1 contains the results obtained by this system in the poliMèdia task.

Apart of being used to transcribe educational videos in poliMèdia, this system is currently providing automatic transcription services to ÀPunt and TV3, the autonomic television providers for the Valencian Country and Catalonia, respectively.

Table 4.1: Results of the baseline system on the poliMèdia dataset.

	Dev	Test
Baseline	7.6	8.5

4.2 Acoustic adaptation

With the baseline results defined, we started this task by preparing the data. As mentioned in Section 3.4, after selecting the data for each task, we were left with around 2634 hours of audio. From this point, we performed an initial study to check if there were videos that, even if they had audio and transcriptions, could present a problem. With this, we found 31 videos with white noise and around 70 videos with very bad audio quality, which caused the system to return an intelligible transcription. We opted for removing those videos, as they roughly represented the 0.6% of the total amount of training data. However, it opened the possibility that a bigger percentage of the data could present similar issues. This phenomenon could be explained by the fact that, although the poliMedias are recorded in very standardized conditions, there can be technical problems that may not be detected if the video is not supervised afterwards. Also, due to the big volume of video data stored in the UPVMèdia repository, there may be some non-poliMèdia videos downloaded in our data dump.

Once in this point, we started processing our data. For this, we replicated the preprocessing steps performed by the baseline system: For the text part, we removed the punctuation marks and all the non-Latin characters, expanded the abbreviations, tokenized the corpus into words, transliterated the numbers and performed a cleaning step. At the end, we were left only with plain text without punctuation marks, numbers or abbreviations, only with characters from the Latin alphabet encoded with Unicode. In the side of the audio data, we converted the video files into wav files and extracted the feature vectors using the `tLtask-preprocess` utility. From this point, in order to be able to train the model with our data, we need to align it. The alignment process is performed to locate the text data inside the acoustic data. For this, we performed a recognition with the baseline's AM to obtain the position of each phoneme of the transcription, inside the audio data. The next step was to normalize the audio vectors. It is possible to normalize them at video-level by taking into account every sample in the video before normalizing. However, as we mentioned in the previous section, our baseline system was built to be able to recognise in streaming environments, where you can't see the future and, thus, normalize the samples at video-level. For this, we performed a normalization process that simulates an on-the-fly normalization by using a sliding context window, similar to the normalization process that the decoder performs in inference time. The final data processing step was to convert the feature vectors into *tFrecords,* tensorFlow's sample format, to be ready to train the net.

Next, once the processing step had finished, we prepared the training environment. From all the adaptation techniques mentioned in Section 2.5, we decided to explore the one based on empirical Bayes and L2 regularisation, combining it with self-learning.

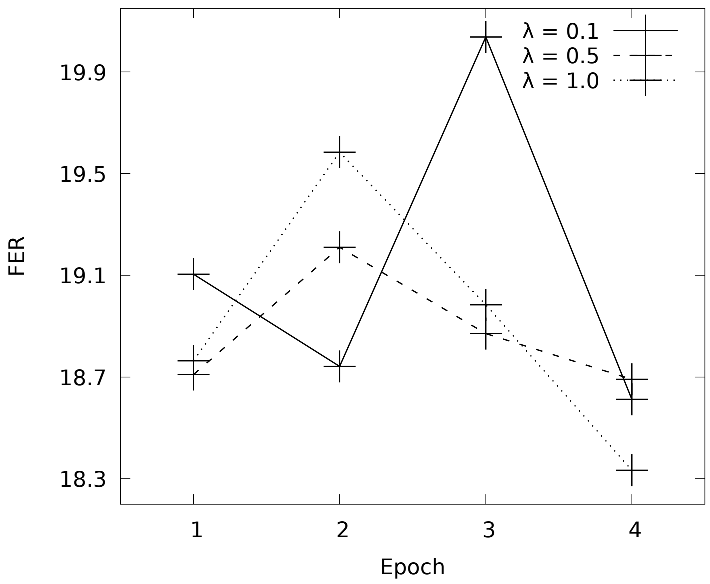


Figure 4.1: Evolution of the FER over the training process for $\lambda = \{0.1, 0.5, 1.0\}$ through 4 epochs

The reason for this decision was that self-learning can be used to train with large amounts of self-generated unsupervised data, which is the main idea of this work, but it is usually applied by performing some fine-tuning process. However, from previous experiences of the *MLLP* group, performing a fine-tuning is a delicate process, as it demands a very specific learning rate that allows the net to learn the new domain, but without overfitting to it. With this idea in mind, and to try to overcome the problem of setting a correct learning rate for the fine tuning process, we can introduce the regularisation term to the training loss function as in Equation 2.17, allowing us to use a more aggressive learning rate while reducing the risk of overfitting by inducing a bigger penalty the more the new model differs from the previous one.

Once the training process was defined and the data was preprocessed, the next step was to adapt the models. In this case, we proposed 3 regularisation parameters $\lambda = \{0.1, 0.5, 1\}$ to explore its effect on the training evolution. All the training processes where performed with an initial learning rate $lr = 2.225e-5$. As mentioned in Section 2.6, the evaluation metric used in the training process of acoustic models is the frame error rate (FER), that measures the rate between correct and wrong frame classifications. The evolution of the training process can be seen in Figure 4.1.

Every epoch took around 2 days running on a NVIDIA RTX 2080ti card. As it can be seen, the evolution is not constant, as it varies from improving the result to worsening it. This phenomenon could be explained due to the fact that we did not explore different learning rate values and may be mitigated by using lower values.

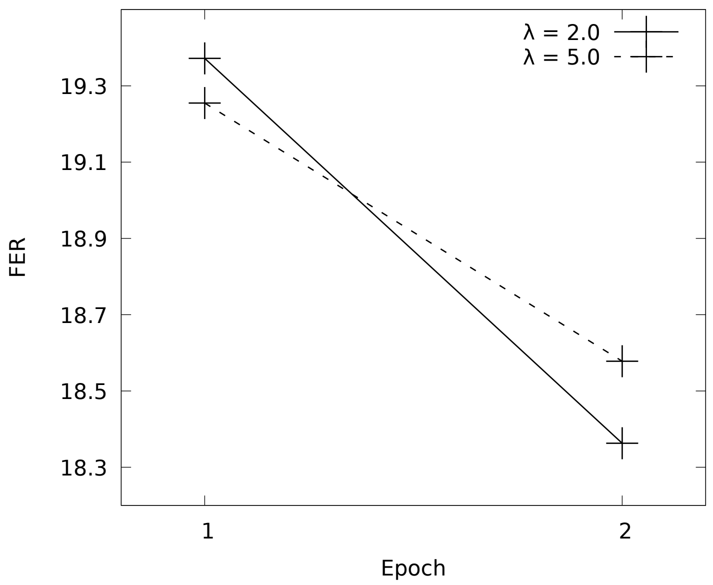


Figure 4.2: Evolution of the FER over the training process for $\lambda = \{2.0, 5.0\}$ through 2 epochs

However, it can be seen that there is a tendency to smooth this behaviour with greater values of λ. Considering this tendency, we decided to launch to more training processes, now with $\lambda = \{2.0, 5.0\}$. However, due to time constraints, it was only possible to complete two epochs on each process. Figure 4.2 shows the evolution of the fer during both trainings.

Although 2 epochs are not enough for extracting conclusions, we can see that the behaviour experienced with lower values for λ seems to have been even more smoothed. At the absence of further exploration, and attending to the absolute best result obtained by every model, we could hypothesise that the best values for λ are $\lambda = \{1.0, 2.0\}$, with a best FER score of 18.333 and 18.363, respectively.

4.3 Integration of the new AMs into the hybrid decoder

Once all the models were trained, the next step was to integrate them into the decoder, considering the same setup as with the baseline system. For this, we designed an experimentation environment arround the tool `tLtask-recognise`. This tool, which implements TLK's decoder, receives both the AM and the LM and performs the decoding step, leaving some adjustable hyperparameters to tune this process. From those hyperparameters, and since this work adapted the acoustic part of the system,

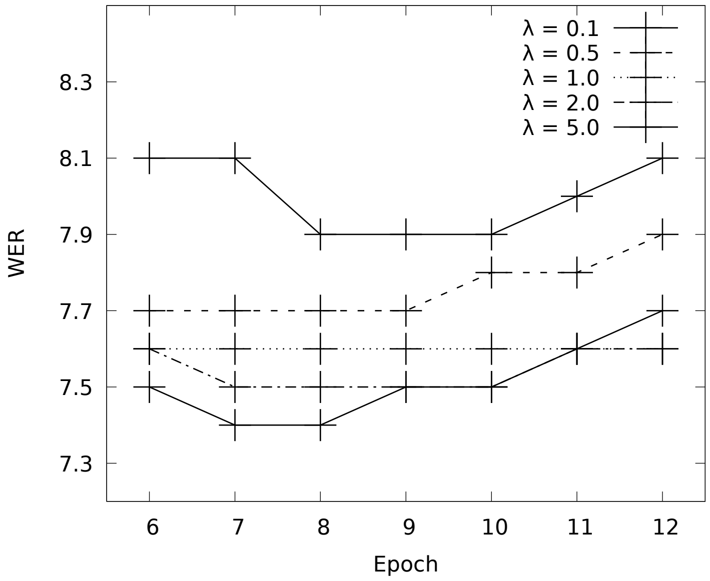


Figure 4.3: WER scores plotted against the GSF values explored.

the exploration will focus on:

- **GSF** or *Grammar Scale Factor*, used to scale the weight of the LM with respect to the AM in the decoding step. It matches the α factor in Equation 2.11.
- **PSF** or *Prior Scale Factor*, used to scale the prior probabilities $p(s_t)$ of the HMM states learned by the AM.

The rest of the hyperparameters (LMHR, HP, LMHP, BEAM...) were left with the values that were considered optimal for the baseline system, since they are not directly related to the AM model. `tLtask-recognise` also allows to tune the size of the "future" context in frames given to the AM via the **LA** (Look-Ahead) parameter. However, we focused on comparing our system with the offline version of the baseline system (check Table 4.1), so we fixed this parameter to $LA = 99$ frames, which is the standard Look-ahead value used by *MLLP*'s offline systems. Having set the experimental environment and the parameters to explore, we decided to explore the *GSF* parameter with values $GSF = \{6, 7, 8, 9, \mathbf{10}, 11, 12\}$, and *PSF* with the values $PSF = \{0.5, 0.6, 0.7, \mathbf{0.8}, 0.9, 1.0\}$, being $GSF = 10$ and $PSF = 0.8$ the optimal values for the baseline system. To perform this exploration, we decided to do a two step exploration to reduce the number of test cases. With this plan, we first fixed the *PSF* parameter and explored the *GSF* values and, then, we fixed *GSF* to the best value we found to explore the values of *PSF*. For this exploration, we used the model yielded by the last epoch of training for each value of λ, this is, the model

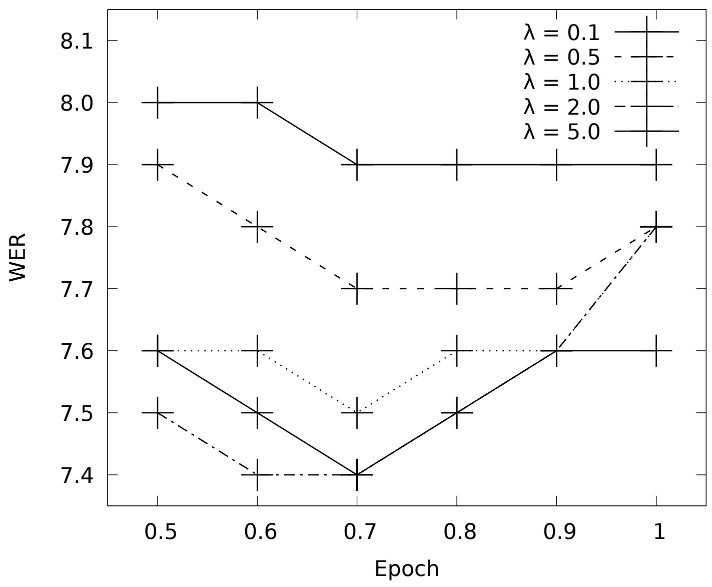


Figure 4.4: Results of the PSF exploration in WER scores.

trained after 4 epochs for $\lambda = \{0.1, 0.5, 1.0\}$, and 2 epochs for $\lambda = \{2.0, 5.0\}$. The first exploration was performed with a fixed $PSF = 0.8$. Figure 4.3 plots the WER results obtained against the GSF values explored.

In the light of the results, we can see that the GSF value that provided the best results overall is $GSF = 8$. This result, 2 points lower than the optimal value of the baseline system, could be explained by taking into account that the GSF parameter gives more importance to the AM (and less to the LM) with higher values. Considering that we adapted the AM to the poliMèdia task, the AM has more expert knowledge and, thus, we can give it more weight and rely less on the language structure knowledge of the LM. After exploring GSF and having set it to the best value, $GSF = 8$, we explored the proposed values for PSF. Figure 4.4 shows the results in WER plotted against the considered PSF values.

Based on the outcome, we can see that the value that provides the best results is $PSF = 0.7$. With both experiments finished and analyzed, we can conclude that the best combination of hyperparameters is $GSF = 8, PSF = 0.7$, which yields best result of 7.5 WER points with the model trained with $\lambda = 1.0$.

4.4 Results and comparison

Once we finished the hyperparameter exploration and obtained an optimal combination, we compared our results with the baseline results. The recognition process was performed in all cases using the `tLtask-recognise` command-line tool from TLK

Table 4.2: WER % results of our ASR systems compared to the baseline system. Our systems are represented by their regularisation value λ. Results on the development partition of poliMèdia have been added to complete the comparison.

	Dev	Test
Baseline	7.6	8.5
$\lambda = 0.1$	7.9	8.3
$\lambda = 0.5$	7.7	8.3
$\lambda = 1.0$	7.5	8.3
$\lambda = 2.0$	7.4	**8.2**
$\lambda = 5.0$	7.4	8.3

using the test partition from the poliMèdia corpus. Table 4.2 summarizes the WER figures obtained for all 6 systems considered.

In the light of the results, the system that provided the best WER score is the one adapted with $\lambda = 2.0$, obtaining a 4% relative improvement over the baseline results. However, this comparison is not complete, as the models trained with $\lambda = \{2.0, 5.0\}$ were trained only during 2 epochs, whereas the rest of the models were trained during 4 epochs. Although the system that provided the best results is the one with the $\lambda = 0.2$ AM, a clear future work line is to evaluate all the models with an equal training time and compare them in equal circumstances.

Chapter 5

Speaker-aware self-supervised acoustic adaptation

This chapter compiles the work done to adapt the baseline system to the speakers, as explained in Section 3.4. Its structure follows this outline: Section 5.1 describes the baseline system and defines the baseline results. Then, Section 5.2 exposes the work done to adapt the acoustic model to the speakers in the test set of poliMèdia. Next, Section 5.3 integrates the adapted AMs to the decoder to build our proposed speaker-adapted systems. Finally, Section 5.4 exposes the results of our proposed systems and compares them to the baseline results.

5.1 Baseline system

As explained in Section 3.4, this task aims to adapt the baseline system previously considered in this work to each speaker on the original poliMèdia test split using unlabelled data. The baseline system, is already explained with more detail in Section 4.1, although we will give a brief description here. It is a system based on a one-pass decoder with the ability of performing on streaming environments without losing performance. Its language model consists on an interpolation of a 4-gram LM with a Transformer-based LM, whereas the acoustic model is a BLSTM-HMM with 8 BLSTM layers of 1024 units each and an output layer of 10041 units. The AM was trained with around 3900 hours of audio retrieved from different tasks. Table 5.2 exposes the WER scores of the baseline system over each speaker on the test split of the poliMèdia corpus.

Table 5.1: Baseline results on each speaker of the poliMèdia test speaker.

	Baseline results
Test speaker 1	6.6
Test speaker 2	9.9
Test speaker 3	7.4

5.2 Acoustic adaptation

As mentioned in Section 3.4, we were left with a total amount of 4 hours shared among 4 of the 5 speakers in the test split. However, as it can be seen from Table 3.2, only three of them had a significative amount of hours to allow speaker adaptation. For this, we decided to discard speakers 4 and 5 and focus only on speakers $1, 2$ and 3, with $1.14, 1.30$ and 1.63 hours, respectively.

Once we decided the speakers and the task was defined, we started to preprocess the unsupervised data. At this point, we performed the same preprocess steps that were used to train the baseline AM. Since these steps are already described in the previous chapter, the reader is redirected to Section 4.2 if it has not been red yet. However, a brief description is given in the following lines: to process the text data, we removed every non-Latin character and punctuation marks, expanded the abbreviations, transliterated the numbers and encoded the text in Unicode format. On the side of the acoustic data, we extracted filterbank vectors with 85 components using the `tLtask-preprocess` tool and aligned them with their transcriptions using the baseline model. Then, we normalized the data by simulating an on-the-fly normalizing step and converted the feature vectors and their labels to a tensorFlow representation of sequential samples named *tFrecords*.

Once all the data was processed, we were ready to train the new models. For the training process, we decided to follow the empirical Bayes approach that was introduced in the previous section. However, since the amount of training data was very low, specially when compared to the previous task, we directly thought of using higher values for the regularisation parameter to avoid overfitting. Based on this reasoning, we decided to train with $\lambda = \{1.0, 2.0, 5.0\}$. We also used the same learning rate as in the previous task ($lr = 2.225e - 5$) to allow us to compare the impact of the empirical Bayes technique in both tasks. With the training environment already defined, Figure 5.1 shows the evolution of the training process across the epochs in terms of FER.

Each epoch took around 5 minutes running on a NVIDIA RTX 2080ti. As it can be seen, even if the training parameters are the same, the particularities of each speaker's speech have a clear effect on the results. In the case of Speaker 3, the training process yielded an improvement of the previous results, evolving slowly but constantly towards lower FER scores. However, on another note, Speaker 2 presented a very different behaviour. In this case, the FER evolved in a way more random manner, alternating between increasing and decreasing its value. In the middle of both cases, Speaker 1 presents an stable FER value throughout the whole training

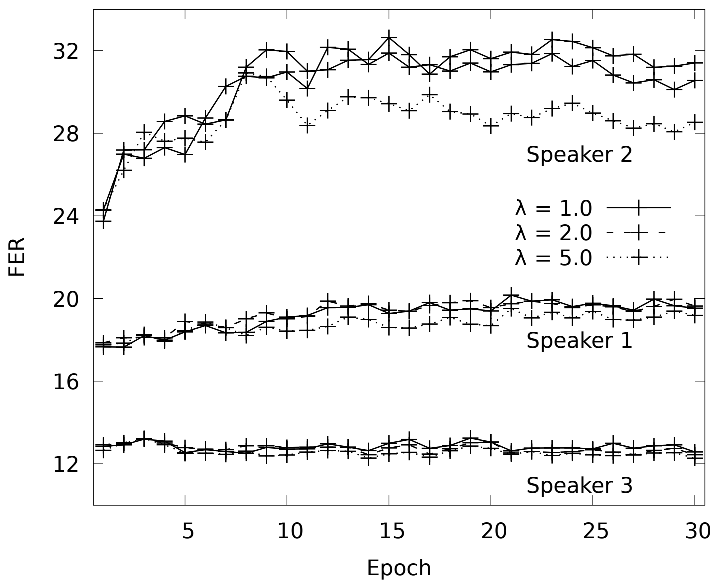


Figure 5.1: Evolution of the FER over the training process for $\lambda = \{1.0, 2.0, 5.0\}$ for each speaker.

process, improving slightly in comparison to the first stages. With this variety of cases, we can clearly see the effect of the regularisation parameter: In cases where the model performance evolves smoothly, it does not make a big difference. However, in cases where the training process loses control over the model evolution, the regularisation term allows to smooth this behaviour.

If we focus on the independent results of every speaker, it can be seen that they are very disparate, with speaker 3 being the only one to have obtained improvements after the training. After analyzing the results, we concluded that it was not interesting to explore other values of regularisation, as the cases in which the model lost performance couldn't be attributed to a bad selection of parameters, but to a poor amount of training data. However, we could conclude that, in all the cases, the models trained with $\lambda = 5.0$ yielded the best results. Still, and since the hybrid approach of ASR considers the AM and LM separately, an increase or decrease on the training metrics of one of them does not mean that the final system will see the same tendency applied on the WER results.

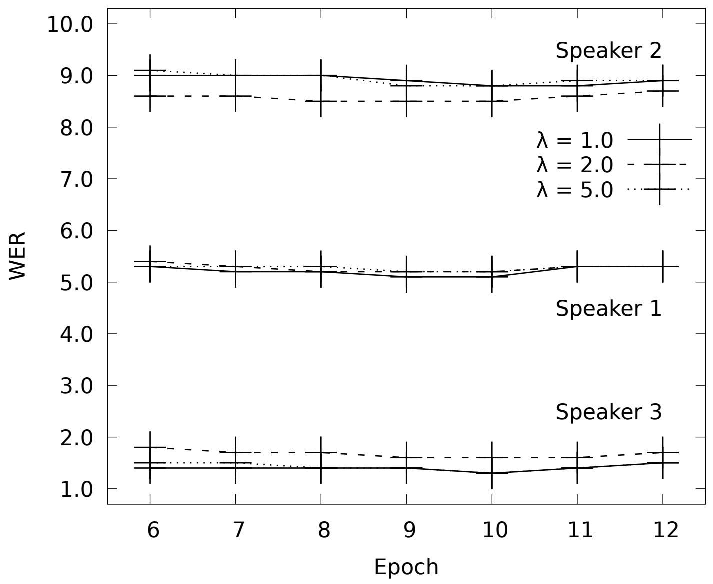


Figure 5.2: Results in WER of the exploration of the GSF parameter.

5.3 Integration of the new AMs into the hybrid decoder

The next step after training the models was to integrate them into the decoder and test them in terms of WER. As in the previous Chapter (in Section 4.3), we designed an experimentation environment around the `tLtask-recognise` tool and proceeded in a similar way: We explored the GSF and PSF hyperparameters over the development set in a two-step exploration. First, we fixed the PSF value to $PSF = 0.8$ and explored the values for GSF. Then, we fixed the GSF parameter to the best value we obtained in the exploration and repeated the process for PSF. The considered parameters were the same as in the previous Chapter: $GSF = \{6, 7, 8, 9, 10, 11, 12\}$ and $PSF = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Figure 5.2 relates the WER values obtained with the explored GSF values.

Looking at the results, we can say that, although the results differ by small intervals, the value for GSF that provided the best result overall is $GSF = 10$. It may be important to note that, since the development partition of this task is also generated with pseudo-labels, results that are closer to 0 may represent models that are closer to the baseline. In addition, note that the models that suffered bigger changes with respect to the baseline model present a bigger impact of the regularisation parameter on the final WER. Once optimal value for the GSF parameter was fixed to $GSF = 10$, we explored the values for the PSF parameter, with the results plotted in Figure 5.3.

Looking at the results, it is important to note that the baseline system yielded a

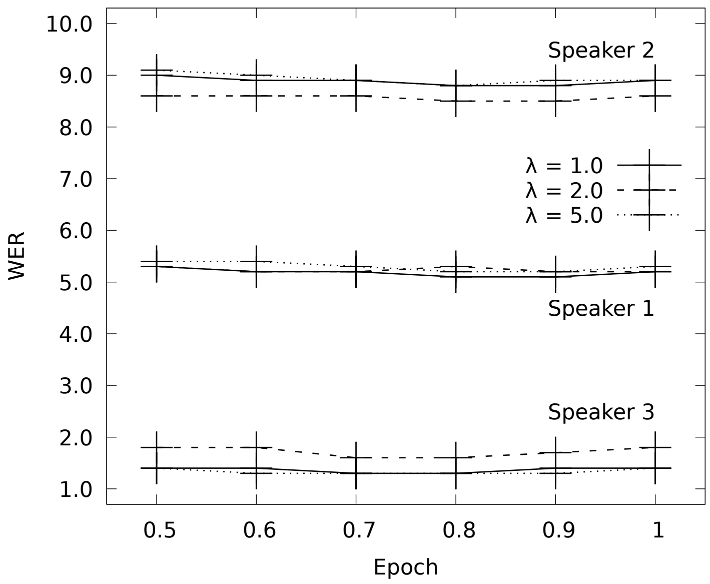


Figure 5.3: WER results against the PSF values explored.

Table 5.2: Results in terms of WER of our proposed speaker-adapted systems in comparison with the baseline system. Our systems are represented by their regularisation value λ. Results on the development have been added to complete the comparison.

Sp 1	Dev	Test
$\lambda = 1.0$	5.2	**6.0**
$\lambda = 2.0$	5.1	6.1
$\lambda = 5.0$	5.2	6.2
Baseline	0.0	6.6

Sp 2	Dev	Test
$\lambda = 1.0$	8.8	10.7
$\lambda = 2.0$	8.5	10.6
$\lambda = 5.0$	8.8	**10.5**
Baseline	0.0	9.9

Sp 3	Dev	Test
$\lambda = 1.0$	1.3	**8.3**
$\lambda = 2.0$	1.6	8.4
$\lambda = 5.0$	1.3	8.5
Baseline	0.0	7.4

transcription with a WER score of 0.0. This is due to the fact that this transcription was generated in the past by the same model. As we can see from the results, the results are also separated by small intervals, but the value that performs the best is $PSF = 0.8$. This triggered our curiosity, as these are the same optimal values as the baseline system. This may be explained by considering that the different acoustic models that we trained did not differ much from the baseline system and, thus, the difference may not be enough to provoke a change in these hyperparameters. With this, we can state that the best hyperparameters for this systems are $GSF = 10, PSF = 0.8$. However, in this case we can not identify a value for λ that performs better than the others.

5.4 Results and comparison

Once the systems are ready and we explored the optimal hyperparameters, we can compare them against the baseline system with the test split. As with the previous Chapter, the recognition process was performed in all cases using the `tLtask-recognise` command-line tool from TLK using the test partition of the poliMèdia corpus.

In the light of the results, we can see that Speakers 2 and 3 did not improve their baseline result, having obtained scores that are 6% and 10.8% worse, respectively. However, in the case of Speaker 1, all three systems improved the baseline results, with a 9% improvement in the case of the best result (the one with $\lambda = 1.0$). With this results, we can conclude that, with this amount of training data, we can not make sure that speaker adaptation will always improve the results of an all-purpose system. However, in the cases where it achieves an improvement on the result, it can be considered a significative improvement. However, further exploration of this technique with more training data is left to be done.

Chapter 6

Conclusions and future work

This work has exposed the process of adapting the acoustic part of a hybrid ASR system to a certain task. To do this, we explored the state-of-the-art techniques for acoustic adaptation of ASR systems and selected the ones that suited the best our case. Then, we applied them to adapt an acoustic model using more than 2.6K hours of unsupervised audio data, pseudo-labeled with the same system that is going to be adapted. Finally we combined our adapted acoustic models with the rest of the components of the baseline system to ensemble our proposed adapted systems, and compared them with the results of the baseline system.

With respect to the tasks considered, we proposed to different tasks. The first one was adaptation to the poliMèdia task, which has very specific audio characteristics, using all the media labelled with the baseline system since it was put in production, back in 2020. The second task was to perform speaker-adaptation, also using unsupervised self-labeled data. For this second task, we considered 3 out of the 5 speakers with videos on the test split of the poliMèdia corpus and adapted our systems by using their videos recorded afterwards.

For the first task, following the results from the comparison between our best ASR system proposed and the baseline system, it can be said that the goal of improving the existing results was achieved, as our best system achieved a WER improvement of 4% w.r.t. the baseline model. In the case of the second task, we were only able to improve the results for one of the speakers we considered, but we achieved an impressive 9% improvement w.r.t. the baseline system.

6.1 Future Work

This work leaves many aspects that can be further explored and are open for improvement. Some of them are:

- To perform an exhaustive exploration for the combination of learning rate and regularisation parameter for the training process. In this work, we used $lr = 2.225e{-}5$ in every training process we performed, varying only the regularisation parameter λ. However, a further exploration is needed to find a balance between these two parameters.
- To study our poliMèdia dump and check for strange data. In this work, we found various cases of videos without audio or transcription, videos that only had white noise in the audio channel, or videos that are not classified as poliMèdia.
- To explore the speaker adaptation task with a bigger amount of data. For this work, we only had access to 4 hours of data, leaving us with between 1 and 2 hours for each speaker we considered. If suspect that, with more data, the results will improve.
- To repeat the comparison between the systems of the first task with equal conditions for all the models. Due to time constraints, two out of the five proposed AMs were trained during two epochs, whereas the other tree models where trained during four epochs.

Bibliography

[1] ClassCentral, 2023. The 100 most popular online courses of all time (2023). https://www.classcentral.com/report/most-popular-online-courses. Retrieved on June 2023.

[2] MediaUPV, 2020. The MediaUPV repository. https://media.upv.es. Retrieved on June 2023.

[3] UPValenciaX, 2020. UPValenciaX: UPV as an edX member. https://www.edX.org/school/upvalenciax. Retrieved on June 2023.

[4] UPVX, 2020. UPVX: The MOOC initiative at the UPV. https://www.upvx.es. Retrieved on June 2023.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Łukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[6] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common Voice: A massively-multilingual speech corpus. *CoRR*, abs/1912.06670, 2019.

[7] Pau Baquero-Arnal, Javier Jorge, Adrià Giménez, Joan Albert Silvestre-Cerdà, Javier Iranzo-Sánchez, Albert Sanchis, Jorge Civera, and Alfons Juan. Improved Hybrid Streaming ASR with Transformer Language Models. In *Proc. Interspeech 2020*, pages 2127–2131, 2020.

[8] G. Bebis and M. Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[11] Germán Bordel, M Inés Torres, and Enrique Vidal. Back-off smoothing in a syntactic approach to language modelling. In *ICSLP*, volume 94, pages 851–854, 1994.

[12] Thomas Cherian, Akshay Badola, and Vineet Padmanabhan. Multi-cell LSTM Based Neural Language Model, 2018.

[13] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. In *NIPS*, 2015.

[14] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. In *Machine Learning*, pages 273–297, 1995.

[15] A Pérez González de Martos, Joan Albert Silvestre-Cerda, JD Valor Miró, J Civera, and A Juan. Mllp transcription and translation platform. In *Proceedings of 10th European Conference on Technology Enhanced Learning. EC-TEL*, 2015.

[16] M. A. del Agua, A. Giménez, N. Serrano, J. Andrés-Ferrer, J. Civera, A. Sanchis, and A. Juan. The Translectures-UPV Toolkit. In Juan Luis Navarro Mesa, Alfonso Ortega, António Teixeira, Eduardo Hernández Pérez, Pedro Quintana Morales, Antonio Ravelo García, Iván Guerra Moreno, and Doroteo T. Toledano, editors, *Advances in Speech and Language Technologies for Iberian Languages*, pages 269–278, Cham, 2014. Springer International Publishing.

[17] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley, New York, 2 edition, 2001.

[18] U. Essen and H. Ney. On smoothing techniques for bigram-based natural language modelling. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, pages 825–828, Los Alamitos, CA, USA, apr 1991. IEEE Computer Society.

[19] William A Gale and Geoffrey Sampson. Good-turing frequency estimation without tears. *Journal of quantitative linguistics*, 2(3):217–237, 1995.

[20] Gonçal V. Garcés Díaz-Munío, Joan Albert Silvestre-Cerdà, Javier Jorge, Adrià Giménez, Javier Iranzo-Sánchez, Pau Baquero-Arnal, Nahuel Roselló, Alejandro Pérez-González de Martos, Jorge Civera, Albert Sanchis, and Alfons Juan. Europarl-ASR: A large corpus of parliamentary debates for streaming asr benchmarking and speech data filtering/verbatimization. In *Proc. Interspeech 2021*, Brno (Czech Republic), 2021.

[21] Ismael García Varea. *Traducción automática estadística: modelos de traducción basados en máxima entropía y algoritmos de búsqueda.* PhD thesis, Universitat Politècnica de València, 2003. Advisor: Francisco Casacuberta Nolla.

[22] Adrià Giménez Pastor. *Bernoulli HMMs for Handwritten Text Recognition.* PhD thesis, Universitat Politècnica de València, 2014. Advisors: Alfons Juan Ciscar and Jesús Andrés Ferrer.

[23] Mokhtar M Hasan and Pramod K Mishra. Robust gesture recognition using gaussian distribution for features fitting. *International Journal of Machine Learning and Computing*, 2(3):266, 2012.

[24] Simon Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.

[25] Caroline Henton. Bitter pills to swallow. asr and tts have drug problems. *International Journal of Speech Technology*, 8:247–257, 2005.

[26] Takuya Higuchi, Nobutaka Ito, Takuya Yoshioka, and Tomohiro Nakatani. Robust mvdr beamforming using time-frequency masks for online/offline asr in noise. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5210–5214. IEEE, 2016.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[28] Javier Jorge, Adrià Giménez, Pau Baquero-Arnal, Javier Iranzo-Sánchez, Alejandro Pérez, Gonçal V. Garcés Díaz-Munío, Joan Albert Silvestre-Cerdà, Jorge Civera, Albert Sanchis, and Alfons Juan. MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge. In *Proc. IberSPEECH 2021*, pages 118–122, 2021.

[29] Javier Jorge, Adrià Giménez, Javier Iranzo-Sánchez, Jorge Civera, Albert Sanchis, and Alfons Juan. Real-Time One-Pass Decoder for Speech Recognition Using LSTM Language Models. In *Proc. Interspeech 2019*, pages 3820–3824, 2019.

[30] Javier Jorge, Adrià Martínez-Villaronga, Pavel Golik, Adrià Giménez, Joan Albert Silvestre-Cerdà, Patrick Doetsch, Vicent Andreu Císcar, Hermann Ney, Alfons Juan, and Albert Sanchis. MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge. In *Proc. IberSPEECH 2018*, pages 257–261, 2018.

[31] A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 367–370 Vol.3, 2004.

[32] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456, 2019.

[33] Joshua Y. Kim, Chunfeng Liu, Rafael A. Calvo, Kathryn McCabe, Silas C. R. Taylor, Björn W. Schuller, and Kaihang Wu. A comparison of online automatic speech recognition systems and the nonverbal responses to unintelligible speech, 2019.

[34] Woosung Kim, Sanjeev Khudanpur, and Jun Wu. Smoothing issues in the structured language model. In *INTERSPEECH*, pages 717–720. Citeseer, 2001.

[35] Johannes Lederer. Activation Functions in Artificial Neural Networks: A Systematic Overview, 2021.

[36] Andrew Maas, Quoc V. Le, Tyler M. O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y. Ng. Recurrent neural networks for noise reduction in robust asr. In *INTERSPEECH*, 2012.

[37] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[38] Geoffrey J. McLachlan and Suren Rathnayake. On the number of components in a Gaussian mixture model. *WIREs Data Mining and Knowledge Discovery*, 4(5):341–355, 2014.

[39] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. Coherent Dialogue with Attention-Based Language Models. In *AAAI*, 2017.

[40] Yajie Miao. Kaldi+pdnn: Building dnn-based asr systems with kaldi and pdnn, 2014.

[41] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. volume 2, pages 1045–1048, Jan 2010.

[42] Juan Daniel Valor Miró, Joan Albert Silvestre-Cerdà, Jorge Civera, Carlos Turró, and Alfons Juan. Efficiency and usability study of innovative computer-aided transcription strategies for video lecture repositories. *Speech Communication*, 74:65–75, 2015.

[43] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[44] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[45] David Nolden. *Progress in Decoding for Large Vocabulary Continuous Speech Recognition*. PhD thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, April 2017.

[46] David Nolden, Hermann Ney, and Ralf Schlüter. Time conditioned search in automatic speech recognition reconsidered. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[47] Mohammadreza Asghari Oskoei and Huosheng Hu. Support Vector Machine-Based Classification Scheme for Myoelectric Control Applied to Upper Limb. *IEEE Transactions on Biomedical Engineering*, 55(8):1956–1965, 2008.

[48] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[49] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training Recurrent Neural Networks, 2012.

[50] Alejandro Pérez, Gonçal Garcés Díaz-Munío, Adrià Giménez, Joan Albert Silvestre-Cerdà, Albert Sanchis, Jorge Civera, Manuel Jiménez, Carlos Turró, and Alfons Juan. Towards cross-lingual voice cloning in higher education. *Engineering Applications of Artificial Intelligence*, 105:104413, 2021.

[51] Alejandro Manuel Pérez González de Martos. *Deep Neural Networks for Automatic Speech-To-Speech Translation of Open Educational Resources.* PhD thesis, Universitat Politècnica de València, 2022.

[52] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[53] Anupama Ray, Sai Rajeswar, and Santanu Chaudhury. Text recognition using deep BLSTM networks. In *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pages 1–6, 2015.

[54] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[55] Anthony Rousseau, Paul Deléglise, and Yannick Estève. TED-LIUM: an automatic speech recognition dedicated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 125–129, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[56] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[57] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays. Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition, 2015.

[58] Antonia Schulte, Rodrigo Suarez-Ibarrola, Daniel Wegen, Philippe-Fabian Pohlmann, Elina Petersen, and Arkadiusz Miernik. Automatic speech recognition in the operating room–an essential contemporary tool or a redundant gadget? a survey evaluation among physicians in form of a qualitative study. *Annals of Medicine and Surgery*, 59:81–85, 2020.

[59] Dev Shah, Wesley Campbell, and Farhana H. Zulkernine. A comparative study of lstm and dnn for stock market forecasting. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4148–4155, 2018.

[60] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, mar 2020.

[61] Yongzhe Shi, Wei-Qiang Zhang, Meng Cai, and Jia Liu. Efficient One-Pass Decoding with NNLM for Speech Recognition. *IEEE Signal Processing Letters*, 21(4):377–381, 2014.

[62] Joan Albert Silvestre-Cerdà, Miguel Del Agua, Gonçal Garcés, Guillem Gascó, Adrià Giménez-Pastor, Adrià Martínez, Alejandro Pérez González de Martos, Isaías Sánchez, Nicolás Serrano Martínez-Santos, Rachel Spencer, Juan Daniel Valor Miró, Jesús Andrés-Ferrer, Jorge Civera, Alberto Sanchís, and Alfons Juan. transLectures. In *Proceedings of IberSPEECH 2012*, pages 345–351, Madrid (Spain), 2012.

[63] Joan Albert Silvestre-Cerdà, Alejandro Pérez, Manuel Jiménez, Carlos Turró, Alfons Juan, and Jorge Civera. A system architecture to support cost-effective transcription and translation of large video lecture repositories. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3994–3999. IEEE, 2013.

[64] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020.

[65] Andreas Stolcke and Jasha Droppo. Comparing human and machine errors in conversational speech transcription. *CoRR*, abs/1708.08615, 2017.

[66] Carlos Turro, Ignacio Despujol Zabala, Aristóteles Cañero, and Jaime Busquets. Deployment and analysis of lecture recording in engineering education. *Proceedings - Frontiers in Education Conference, FIE*, 2015, 02 2015.

[67] Carlos Turro, Miguel Ferrando-Bataller, Jaime Busquets, and Aristóteles Cañero. Polimedia: a system for successful video e-learning. 06 2009.

[68] Zoltán Tüske, George Saon, and Brian Kingsbury. On the limit of English conversational speech recognition. *CoRR*, abs/2105.00982, 2021.

[69] Juan Daniel Valor Miró, Joan Albert Silvestre-Cerdà, Jorge Civera, Carlos Turró, and Alfons Juan. Efficient generation of high-quality multilingual subtitles for video lecture repositories. In *Design for Teaching and Learning in a Networked World: 10th European Conference on Technology Enhanced Learning, EC-TEL 2015, Toledo, Spain, September 15-18, 2015, Proceedings 10*, pages 485–490. Springer, 2015.

[70] Vidushi Vashishth, Anshuman Chhabra, and Deepak Kumar Sharma. GMMR: A Gaussian mixture model based unsupervised machine learning approach for optimal routing in opportunistic IoT networks. *Computer Communications*, 134:138–148, 2019.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *ArXiv*, abs/1706.03762, 2017.

[72] Changhan Wang, Morgane Rivière, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Miguel Pino, and Emmanuel Dupoux. VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. *CoRR*, abs/2101.00390, 2021.

[73] Chenguang Wang, Mu Li, and Alexander J. Smola. Language Models with Transformers. *CoRR*, abs/1904.09408, 2019.

[74] Felix Weninger, Hakan Erdogan, Shinji Watanabe, Emmanuel Vincent, Jonathan Le Roux, John R Hershey, and Björn Schuller. Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr. In *Latent Variable Analysis and Signal Separation: 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings 12*, pages 91–99. Springer, 2015.

[75] Dominika Woszczyk, Stavros Petridis, and David Millard. Domain adversarial neural networks for dysarthric speech recognition, 2020.

[76] I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification, 2019.

[77] Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[78] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014.

[79] Luke Zhou, Kathleen C. Fraser, and Frank Rudzicz. Speech Recognition in Alzheimer's Disease and in its Assessment. In *Proc. Interspeech 2016*, pages 1948–1952, 2016.

List of Figures

List of Tables

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APPENDIX

SUSTAINABLE DEVELOPMENT GOALS

Degree to which the work relates to the Sustainable Development Goals (SDGs).

Sustainable development goals	**High**	**Medium**	**Low**	**Not applicable**
SDG 1. **No poverty.**				**X**
SDG 2. **Zero hunger.**				**X**
SDG 3. **Good health and well-being.**			**X**	
SDG 4. **Quality education.**	**X**			
SDG 5. **Gender equality.**				**X**
SDG 6. **Clean water and sanitation.**				**X**
SDG 7. **Affordable and clean energy.**				**X**
SDG 8. **Decent work and economic growth.**			**X**	
SDG 9. **Industry, innovation and Infrastructure.**		**X**		
SDG 10. **Reduced Inequality.**		**X**		
SDG 11. **Sustainable cities and communities.**				**X**
SDG 12. **Responsible consumption and production.**				**X**
SDG 13. **Climate action.**				**X**
SDG 14. **Life below water.**				**X**
SDG 15. **Life on land.**				**X**
SDG 16. **Peace and justice strong institutions.**			**X**	
SDG 17. **Partnerships to achieve the goal.**			**X**	

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Reflexion on the relation of the TFG/TFM with the SDGs and with the most related SDG(s).

This work fits with the United Nations' Sustainable Development Goals (SDGs). In particular, with SDG 4, on "Quality Education", which aims to "ensure inclusive and equitable quality education and promote lifelong learning opportunities for all".
Within SDG 4, the most related targets to this work are:

- *4.3 By 2030, ensure equal access for all women and men to affordable and quality technical, vocational and tertiary education, including university.*
- *4.4 By 2030, substantially increase the number of youth and adults who have relevant skills, including technical and vocational skills, for employment, decent jobs and entrepreneurship.*
- *4.5 By 2030, eliminate gender disparities in education and ensure equal access to all levels of education and vocational training for the vulnerable, including persons with disabilities, indigenous peoples and children in vulnerable situations.*

This work contributes to increasing accessibility to educational resources for everyone (target 4.3), including people with hearing disabilities and persons with fewer resources or without access to formal education systems (target 4.5), aside from those whose acces to educational and formation resources requires them to learn and master a foreign language (target 4.4).