UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Transformer models for Machine Translation and Streaming Automatic Speech Recognition

Author: Pau Baquero Arnal

Directors: D. Alfons Juan Císcar
D.-Ing. Hermann Ney

March, 2023

Transformer models for Machine Translation and Streaming Automatic Speech Recognition

Pau Baquero Arnal

Thesis performed under the supervision of doctors Alfons Juan Císcar and Hermann Ney and presented at the Universitat Politècnica de València in partial fulfilment of the requirements for the degree of *Doctor en Informàtica*.

València,
September 26, 2022

Work supported by the predoctoral research scholarship ACIF/2017/055 and EU's H2020 research and innovation programme under grant agreement no. 761758 (X5gon).

Acknowledgements

Durants aquests anys que han sigut la meua etapa predoctoral en el grup de recerca MLLP he tingut la sort de compartir viatge amb un equip de persones que, a més de ser excel·lents en la seua forma de treballar, també destaquen per les seues qualitats humanes. Junt a ells he crescut i he aprés moltíssim, i he compartit una part de la meua vida que segur que em marcarà de cara al futur. A ells els done les gràcies per acompanyar-me en esta etapa tan crítica en la meua formació. En ordre aleatori però d'igual importància: Gonçal, Javi I., Javi J., Álex, Adrià G., Adrià M., Javi J., Joan Albert, i sense deixar de banda al meu director de tesi Alfons Juan i els altres professors del grup Jorge Civera i Albert Sanchis. A ells els done les gràcies pel suport, per la confiança, i per l'equip que heu sabut conformar al llarg dels anys i que m'ha acollit durant aquests anys.

Vull també aprofitar per donar les gràcies per donar-me caliu i suport tot aquest temps a la meua família, en especial a ma mare Xelo, a la meua germana Marina, a la meua parella Clara, i als meus amics, la llista dels quals no m'atrevisc a fer perquè no sabria on tallar-la ni com ordrenar-los. Gràcies a tots.

Pau Baquero Arnal
València
Novembre 2022

Abstract

Natural language processing (NLP) is a set of fundamental computing problems with immense applicability, as language is the natural communication vehicle for people. NLP, along with many other computer technologies, has been revolutionized in recent years by the impact of deep learning. This thesis is centered around two keystone problems for NLP: machine translation (MT) and automatic speech recognition (ASR); and a common deep neural architecture, the Transformer, that is leveraged to improve the technical solutions for some MT and ASR applications.

ASR and MT can be utilized to produce cost-effective, high-quality multilingual texts for a wide array of media. Particular applications pursued in this thesis are that of news translation or that of automatic live captioning of television broadcasts. ASR and MT can also be combined with each other, for instance generating automatic translated subtitles from audio, or augmented with other NLP solutions: text summarization to produce a summary of a speech, or speech synthesis to create an automatic translated dubbing, for instance. These other applications fall out of the scope of this thesis, but can profit from the contributions that it contains, as they help to improve the performance of the automatic systems on which they depend.

This thesis contains an application of the Transformer architecture to MT as it was originally conceived, achieving state-of-the-art results in similar language translation. In successive chapters, this thesis covers the adaptation of the Transformer as a language model for streaming hybrid ASR systems. Afterwards, it describes how we applied the developed technology for a specific use case in television captioning by participating in a competitive challenge and achieving the first position by a large margin. We also show that the gains came mostly from the improvement in technology capabilities over two years including that of the Transformer language model adapted for streaming, and the data component was minor.

Resum

El processament del llenguage natural (NLP) és un conjunt de problemes computacionals amb aplicacions de màxima rellevància, que juntament amb altres tecnologies informàtiques s'ha beneficiat de la revolució que ha significat l'impacte de l'aprenentatge profund. Aquesta tesi se centra en dos problemes fonamentals per al NLP: la traducció automàtica (MT) i el reconeixement automàtic de la parla o transcripció automàtica (ASR); així com en una arquitectura neuronal profunda, el Transformer, que posarem en pràctica per a millorar les solucions de MT i ASR en algunes de les seues aplicacions.

l'ASR i MT poden servir per obtindre textos multilingües d'alta qualitat a un cost raonable per a un gran ventall de continguts audiovisuals. Concretament, aquesta tesi aborda problemes com el de traducció de notícies o el de subtitulació automàtica de televisió. l'ASR i MT també es poden combinar entre ells, generant automàticament subtítols traduïts, o amb altres solucions de NLP: amb resum de textos per produir resums de discursos, o amb síntesi de la parla per crear doblatges automàtics. Aquestes altres aplicacions es troben fora de l'abast d'aquesta tesi però poden aprofitar les contribucions que conté, en la mesura que ajuden a millorar els resultats dels sistemes automàtics dels quals depenen.

Aquesta tesi conté una aplicació de l'arquitectura Transformer al MT tal com va ser concebuda, mitjançant la qual obtenim resultats de primer nivell en traducció de llengües semblants. En capítols subseqüents, aquesta tesi aborda l'adaptació del Transformer com a model de llenguatge per a sistemes híbrids d'ASR en viu. Posteriorment, descriu l'aplicació d'aquest tipus de sistemes al cas d'ús de subtitulació de continguts televisius, participant en una competició pública de RTVE on obtenim la primera posició amb un marge significant. També demostrem que la millora es deu principalment a la tecnologia desenvolupada i no tant a la part de les dades.

Resumen

El procesamiento del lenguage natural (NLP) es un conjunto de problemas computacionales con aplicaciones de máxima relevancia, que junto con otras tecnologías informáticas se ha beneficiado de la revolución que ha significado el aprendizaje profundo. Esta tesis se centra en dos problemas fundamentales para el NLP: la traducción automática (MT) y el reconocimiento automático del habla o transcripción automática (ASR); así como en una arquitectura neuronal profunda, el Transformer, que pondremos en práctica para mejorar las soluciones de MT y ASR en algunas de sus aplicaciones.

El ASR y MT pueden servir para obtener textos multilingües de alta calidad a un coste razonable para una diversidad de contenidos audiovisuales. Concretamente, esta tesis aborda problemas como el de traducción de noticias o el de subtitulación automática de televisión. El ASR y MT también se pueden combinar entre sí, generando automáticamente subtítulos traducidos, o con otras soluciones de NLP: resumen de textos para producir resúmenes de discursos, o síntesis del habla para crear doblajes automáticos. Estas aplicaciones quedan fuera del alcance de esta tesis pero pueden aprovechar las contribuciones que contiene, en la meduda que ayudan a mejorar el rendimiento de los sistemas automáticos de los que dependen.

Esta tesis contiene una aplicación de la arquitectura Transformer al MT tal y como fue concebida, mediante la que obtenemos resultados de primer nivel en traducción de lenguas semejantes. En capítulos subsecuentes, esta tesis aborda la adaptación del Transformer como modelo de lenguaje para sistemas híbridos de ASR en vivo. Posteriormente, describe la aplicación de este tipus de sistemas al caso de uso de subtitulación de televisión, participando en una competición pública de RTVE donde obtenemos la primera posición con un marge importante. También demostramos que la mejora se debe principalmenta a la tecnología desarrollada y no tanto a la parte de los datos.

Contents

Chapter 1

Introduction

1.1 Framework and motivation

Technologies working with human language have immense applicability, as language is the natural way in which people communicate: from automatic captioning to a device answering to voice commands, going through machine translation, text summarization, voice synthesis, and more. These technologies have accrued importance over time ever since the inception of computers, and have exploded in terms of both commercial and academic interest in the last decade, when the combination of machine learning approaches, neural networks, large amounts of data, and a higher raw computational power availability seems to have breached a barrier in terms of performance and capabilities of these automatic systems. This thesis is focused on problems that are fundamental to many speech and human language technologies, namely Machine Translation (MT) and Automatic Speech Recognition (ASR).

This work was done with the Machine Learning and Language Processing research group (MLLP), part of the Valencian Research Institute for Artificial Intelligence (VRAIN) inside the Universitat Politècnica de València (UPV). This research group has managed a series of EU and Spanish Government funded research and innovation projects related to the development of state-of-the-art speech and language computer technologies and their application into different types of online learning environments such as Open Educational Resources (OER) repositories, MOOC platforms, etc.

We will present some of these projects for a clear image of the context of this thesis. Some of the projects predate this work, while some others are contemporaneous with it. The work for this thesis was funded by a predoc-

toral research grant from the autonomic government Generalitat Valenciana (ACIF/2017/055), and started in December 2017. Other projects are here presented to provide the research context both preceding and concurrent with this work.

The first of these projects was the Europaen Union's FP7 transLectures project (2011-2014), which goal was to apply state-of-the-art automatic speech recognition and machine translation for transcribing and translating educational resources with automatic help, and developing a platform integrating these technologies with large repositories of video lectures. This project wanted to help break the language barrier for the consumption of online educational materials by providing automatic subtitles in different languages.

After transLectures ended in 2014, other research projects followed with similar alignments: EU's CIP European Multiple MOOC Aggregator (EMMA, 2014–2016), the Spanish Economy Ministry's Multilingual Open Resources for Education (MORE, 2016–2018), EU's Horizon 2020 Cross Modal, Cross Cultural, Cross Lingual, Cross Domain and Cross Site Global OER Network (X5GON, 2017–2020), the Spanish Ministry of Science and Education sponsored Multilingual Subtitling of Classrooms and Plenary Sessions (Multisub, 2019–2021), and finally the project EXPERT (2022, ongoing) supported by the Erasmus+ Education program. All in all, the MLLP-VRAIN research group has maintained an ongoing line of research around speech and human language technologies for more than 10 years, covering both the basic technologies and their application to particular use cases.

Machine Translation and Automatic Speech Recognition, the two research topics that are discussed within this thesis, are core to human language technologies, and improving the performance of MT and ASR systems has a great impact on not only their direct applications but also of other systems down the line. For instance, the results achieved in transLectures and EMMA were very successful, and after 2016 it became clear that ASR and MT technologies were entering the state were they could be used to produce accurate enough multilingual subtitles of video lectures. In fact, since 2014 they are in active use in *UPV[MEDIA]*, the university's institutional video lecture repository, to automatically generate multilingual subtitles for the digital content created by UPV lecturers. A pipeline was defined integrating ASR and MT to build speech translation capabilities: from audio in one language to its text translation in another language. Even more, text-to-speech technologies have started to be used by other members of the same research group to pursue a complete speech-to-speech system, capable of enabling automatic dubbing of audiovisual material. All these possibilities depend on the performance of the fundamental

ASR and MT systems, both in terms of their quality in their output and in terms of their speed and computational demand. The primary contribution from this thesis is to ASR, but it contains some work in MT.

In fact, the work on this thesis started with machine translation. Applying the latest neural Transformer architecture, we were able to reach excellent results in MT tasks such as news translation and related languages translation. The effectiveness of the Transformer architecture being confirmed, we identified a clear way to transfer the gains to other sequence-related tasks from Automatic Speech Recognition and integrated that new architecture into streaming-capable ASR systems that outperformed all previous published results. By doing so, this work strengthened the core technologies that serve as a foundation to technologies working with human language, both in research and in a multitude of potential applications.

1.2 Scientific and technological goals

This section describes the scientific and technological goals pursued in this work.

1. Reach state-of-the-art MT results with the newest neural architectures with a focus on related languages, as the two official languages of the UPV university, Catalan and Spanish, both share an origin in Latin and have potentially exploitable similarities as Romance languages.

2. Introduce new neural models in streaming ASR systems to improve the state-of-the-art of this technology while complying with stringent latency requirements.

3. Apply the reached streaming ASR technical capabilities to a relevant use case by constructing ASR systems that demonstrate the applicability of the technology in competitive and relevant public challenges.

1.3 Document structure

This document is structured in five sequential chapters that cover the different topics and scientific and technological goals proposed in this thesis. Chapter 2 offers preliminary foundational concepts and knowledge relevant for the rest of the thesis. The first scientific goal as presented in this introduction is addressed in Chapter 3, which contains a proposal for neural MT architec-

ture and results with the Transformer architecture achieving excellent results on News translation and a task focused on related languages (Spanish and Portuguese). Chapter 4 addresses the second scientific goal of improving the state-of-the-art in streaming ASR technology. By transferring the Transformer architecture from MT to serve as a language model and adapting it to fit into a streaming-capable ASR hybrid decoder, we were able to achieve improvements on ASR system performance while maintaining compliance of strict latency requirements on long-duration audios. Chapter 5 tackles the third and last goal of this thesis, by taking part in a competitive challenge issued by the main public TV broadcaster of Spain, the Radio Televisión Española (Radio Televisión Española), and achieving the best results of all participants by a wide margin, by employing the technology advances described in Chapter 4 to build specific streaming ASR systems for the Spanish language.

As previously mentioned, this work is made inside the MLLP-VRAIN research group from UPV. The scientific and technical developments, experimentation and evaluations here described have been carried out primarily by the author of this thesis except when explicitly it is otherwise stated. Contributions by others are listed a the end of each chapter when the work described was done in collaboration with others.

The experienced reader can skip the preliminary concepts given in Chapter 2, introducing fundamental machine learning knowledge and basic concepts of human language and speech computer technology such as language models, machine translation, and automatic speech recognition.

Chapter 2

Preliminaries

This chapter serves as a broad foundation to the basic concepts and terminology needed for this work. We will approach some of the problems presented by natural language processing with machine learning models and algorithms designed for sequences, as we will need to work with both sequences of sounds and words, as well as their correspondence to each other.

The Machine Learning field is introduced in Section 2.1. Section 2.2 presents optimization techniques for neural networks. Language models are introduced in Section 2.3. Section 2.4 introduces the principal neural topologies for sequences: the Recurrent Neural Network (RNN), the attention mechanism, and the Transformer model. Finally, the problems of machine translation and automatic speech recognition are presented in Sections 2.5 and 2.6, respectively.

2.1 Machine Learning

Machine Learning or ML is a field of study devoted to computer programs that can learn to perform tasks from examples or experience [Mur22]. We will approach these challenges from a probabilistic perspective, treating all unknown values as *random variables* following a *probability distribution*. The two problems approached in this work will serve as examples, namely Machine Translation (MT) and Automatic Speech Recognition (ASR). When translating from one language to another, the whole output sentence will be considered a random variable following a probability distribution conditioned on the input sentence. When transcribing audio into words, the output sentence will be viewed as a random variable with its probability distribution conditioned on the audio.

Depending on the nature of the output variable, we can fit most machine learning problems in one of these two sets:

- **Classification**: in these problems, the output variable y can take a value from a discrete set $\mathcal{Y}$ of labels known as *classes*: $y \in \mathcal{Y} = \{0, 1, ..., C\}$. A clear example of this is the classification of an image into predefined categories.
- **Regression**: in these problems, the output variable y takes continuous values: $y \in \mathbb{R}$. An example of regression is the prediction of the maximum temperature in a given day.

ML algorithms can be split into three main groups according to the qualities of the data they feed on:

- **Supervised learning**: perhaps the most common form of ML, in these tasks both the input x and the output y are already known in the training data, and we want the model to learn the mapping from the pairs of (x, y). From a probabilistic standpoint, the resulting model fits the distribution $p(y|x)$.
- **Unsupervised learning**: these algorithms are designed to work only with the input data x. As learning some mapping without knowledge about the corresponding output is a really hard problem, these tasks usually revolve around fitting the data. From the probabilistic perspective, we can view most of these tasks as fitting the distribution $p(x)$. A few exceptions to this general rule have been produced, like learning a machine translation system using only monolingual data [ALA19]. This fits partially into a variant of unsupervised learning called **self-supervised learning**, where an originally unsupervised problem is turned into a more approachable supervised learning by auto-generating the labels.
- **Reinforcement learning**: in this class of problems, the system interacts with its environment over time taking actions. The system is not told which action is correct, but has to learn from its consequences via a reward or punishment signal that is programmed to depend on the outcome. A potential difficulty these algorithms face is that the reward or punishment is often delayed in time and it is not directly clear which action in the past is responsible for the outcome.

Although perhaps not immediately evident, MT and ASR are classification tasks. In both kinds of systems, the output is a discrete variable: the sentence.

The fact that the output is a sequence of symbols is an added complication that does not alter the nature of the classification task. In both cases, the end result of the training is a model defining a distribution $p(y|x)$, where y is a sequence of labels and x is the input. In this work, we only make use of supervised learning algorithms. For ASR, we use manually transcribed audio; and for MT, pairs of human translations with their original.

2.2 Optimization

Parameter estimation is at the core of all machine learning algorithms. Models are fit to data by solving an optimization problem for the set of *parameters* θ minimizing an *objective function* $\mathcal{L}$, also called *loss function*. The *parameter space* $\Theta \subseteq \mathbb{R}^D$ represents all the possible values the parameters can take, where D is the number of parameters. The problem is to find the optimal set of values θ^* for the parameters:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta) \tag{2.1}$$

For instance, take the objective function $\mathcal{L}(y, x, \theta) = -\log p_\theta(y|x)$ for a model with parameters θ and data pairs x, y where y is the correct class or label for the input x. This is one one of the most common objective functions used to fit probabilistic machine learning models to data. Minimizing the negative log class posterior probability is equivalent to maximizing the true class posterior probability, in other words performing a *maximum likelihood estimate* for the class posterior probability distribution. In this work, all of our models were trained according to this objective function.

In general, finding global optima for high-dimensional continuous functions is computationally intractable, and we must retreat our aspirations into finding a good local optimum. In this section, we will give a brief overview of the optimization techniques applicable to neural networks. For other types of probabilistic models used in this work, their optimization will be reviewed in their own sections.

For neural networks, we will define our model and loss function as continuously differentiable functions and apply iterative optimization methods based on the gradient of the loss function with respect to the parameters. The simplest of these is *gradient descent*, where we move our parameters a step η_t in the direction of the maximum descent as given by the gradient:

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta) = \theta_t - \eta_t g \tag{2.2}$$

The gradient $\nabla \mathcal{L}(\theta)$ is extremely expensive to compute over the full training dataset. In practice, we compute the gradient over small batches of it as we iterate over the data. Thus, this gradient does not only depend on the value of the parameters but also on the particular data in the batch z_t. This is known as *stochastic gradient descent* or SGD:

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta, z_t) = \theta_t - \eta_t g_t \tag{2.3}$$

The shape of $\mathcal{L}$ on a high-dimensional parameter space Θ can be extremely complicated, with irregular or sloping regions and flat regions. In this conditions, finding a good local optimum with gradient descent is a problem in its own right. To tackle this problem, several methods have been produced.

Firstly, note that the step size η_t has been defined to depend on the step t. This is intentional, as it allows us to define a *learning rate schedule*, adjusting the step size over time to reach a better optimum in less time. Typical schedules include:

- piecewise constant: $\eta_t = \eta_i$ if $t_i \leq t < t_{i+1}$
- exponential decay: $\eta_t = \eta_0 e^{-\lambda t}$
- polynomial decay: $\eta_t = \eta_0 (\beta t + 1)^{-\alpha}$

It is also common to quickly increase the learning rate at the beginning, also known as *learning rate warmup*, and afterwards apply one of the aforementioned schedules. The motivation for this is that usually the random initialization will put θ_0 in a ridged region of $\mathcal{L}$, where the high slope of the gradient will cause changes in θ_t even with a modest η_t. The purpose of initially limiting the learning rate is to find a flat region in the parameter space, and then exploit that flatness with a larger learning rate. Later on, to ensure convergence, the learning rate must be decreased again.

Even with advanced learning rate schedules, SGD can still move too slowly when in flat regions of $\mathcal{L}$. A typical heuristic to help the algorithm is known as *momentum*. The concept is to accelerate along directions that were good in the past, and slow down if the gradient changes direction:

$$m_t = \beta m_{t-1} + (1 - \beta) g_t \tag{2.4}$$
$$\theta_t = \theta_{t-1} + \eta_t m_t \tag{2.5}$$

Inspired by these ideas, many more methods to explore the landscape of the loss function and find good local optima have been produced. The purpose of this work is not to do an extensive review of optimization techniques for neural networks. This section merely serves as an introduction to the core concepts of the field. For the interested reader, we will cite a few of the most popular ones, such as AdaGrad [DHS11], RMSProp [Hin14], and Adam [KB15].

2.3 Language Models

An essential notion for ML systems that output sentences is the Language Model (LM). LMs estimate the *a priori* probability distribution for sentences. They assign a probability for a sentence w to be produced: $p(w)$. In order to compute its probability, the sentence is naturally subdivided into a sequence of words w_1^T and the joint probability for the sequence is obtained via chain rule decomposition:

$$\begin{aligned} p(w) = p(w_1^T) = p(w_T | w_0^{T-1}) p(w_0^{T-1}) = \cdots = \\ = p(w_T | w_0^{T-1}) p(w_{T-1} | w_0^{T-2}) \dots p(w_2 | w_0^1) p(w_1 | w_0) = \prod_{t=1}^{T} p(w_t | w_0^{t-1}) \end{aligned} \tag{2.6}$$

Where w_a^b designates a sequence of symbols w_t where t goes from a to b and w_0 is always the same special symbol designating the start of a sentence. This way, the model is reduced to finding the probability of a particular word w_t given its *history* or preceding words w_0^{t-1}. There are several approaches to the language modelling task. In this section, we will introduce the most relevant to this work, namely unigrams, n-grams, and neural LMs.

Unigrams are the simplest kind of LM. They assign a probability to each word fully disregarding its history and position in the sentence. It is easy to show that a global optimum for the maximum likelihood estimate of this kind of probability distribution can be found over a closed vocabulary by straightforwardly counting the occurrences of each word in the data and dividing by the total number of words. Unigrams have poor predicting power for sentences but

can serve other purposes related to establishing how common or rare a given word is. With an unigram model, the probability for the sentence would follow the equation:

$$p(w) = \prod_{t=1}^{T} p(w_t) = p(w_T)p(w_{T-1}) \dots p(w_2)p(w_1) \tag{2.7}$$

A bit more sophisticated than unigrams, n-grams take into account the last $n-1$ words in the history to predict the current word. They are a type of *Markov model*, which assume that future only depends on the current *state* (in this case, represented by the $n-1$ last words) and not on events further in the past. With this assumption, equation equation 2.6 is simplified into a *Markov chain* of order $n-1$:

$$p(w_1^T) = \prod_{t=1}^{T} p(w_t | w_{t-n+1}^{t-1}) \tag{2.8}$$

For illustration purposes, let's take a bigram model. For each bigram $w_{t-1}w_t$ the probability $p(w_t|w_{t-1})$ would be defined by the model in a lookup table. The probability for a sentence w_1^T would be computed as the following Markov chain of order 1:

$$p(w_1^T) = \prod_{t=1}^{T} p(w_t|w_{t-1}) = p(w_T|w_{T-1})p(w_{T-1}|w_{T-2}) \dots p(w_2|w_1)p(w_1|w_0) \tag{2.9}$$

Where w_0 is a special symbol designating the start of the sentence. With these kinds of models, it can also be shown that there is a closed-form global optimum for the maximum likelihood estimate of the probability distribution. It consists in counting the occurrences of the n-gram and dividing by the counts of the $n-1$-gram consisting of all its words except the last:

$$p(w_n|w_0^{n-1}) = \frac{c(w_0^n)}{c(w_0^{n-1})} \tag{2.10}$$

Where c represents a function counting the number of occurrences in the training data. This estimate, however, assigns zero probability to the n-grams that are absent from the data, which is an unwanted outcome (if the $n-1$-gram

in the denominator is also missing, both counts are zero and the probability would technically become undefined, though it is also treated as zero). There are so-called *discount* methods that smoothen the distribution by taking some probability mass from the observed n-grams and redistributing it to the unobserved n-grams. This can be done uniformly or taking into account the distribution of lower order n-grams, in what is known as Kneser-Ney discount [NEK94].

The last kind of LM, the neural LM, is based on a completely different philosophy where $p(w_t|w_1^{t-1})$ is determined by a neural network. These neural models convert each word in the vocabulary to its own learned continuous representation in a vector space, operation that is called *word embedding*. They use the sequence of word embedding vectors as the input to a neural network, which outputs the probability distribution for the next word, taken as a *softmax* over the vocabulary. The neural network model needs to have a topology appropriate to process sequences. We will later review some neural topologies for sequences used in this work, like the recurrent neural networks and the Transformer networks. Although for this work we use relatively small LMs apt to be integrated as a component in a system to solve another problem such as MT or ASR, LMs present a problem to solve on their own. In particular, large neural LMs have acquired a huge attention after the impact of large pretrained models such as BERT [Dev+18] and GPT [Bro+20] had both on the research community and the industry.

The performance of LMs is assessed with a metric called *perplexity*. Perplexity PP can be defined as the inverse of the probability of a given test sentence, normalising for length T via a Tth root so that the number is expressed in terms of words instead of the whole sentence:

$$PP(w_1^T) = \sqrt[T]{\frac{1}{p(w_1^T)}} \tag{2.11}$$

An equivalent definition can be made in terms of entropy, where $H(w_1^T)$ indicates the average number of bits needed to encode each word:

$$PP(w_1^T) = 2^{H(w_1^T)} = 2^{-1/T \log_2 p(w_1^T)} = \sqrt[T]{\frac{1}{p(w_1^T)}} \tag{2.12}$$

A popular intuition of perplexity is how many words equally probable to those in the sentence would the model be doubting to chose among. A perplexity of 100 means the average word w_t in the sentence is, according to the model, as probable as one in a hundred. The best and lowest possible perplexity for any sentence is 1, true only in the trivial case that it is the only possible sentence. In practice, perplexity is often evaluated not on the sentence level, but over a full test set.

2.4 Neural Model Topologies for Sequences

Before tackling the main problems of machine translation and automatic speech recognition, we will outline the principal neural topologies for sequences that were used in this work. The reader is assumed with a basic understanding of what a neural network consists of, and is referred to [Mur22, Chapter 13] for more details.

2.4.1 Recurrent Neural Network

Recurrent Neural Networks or RNNs define a hidden state h_t for each position t in the sequence. This hidden state depends on the input x_t and the previous state h_{t-1}. The output y_t depends on the hidden state h_t. The weights are shared for all positions.

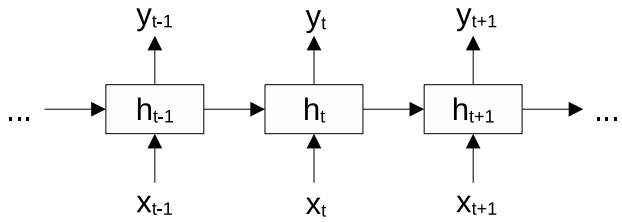

Figure 2.1: Graphical representation of a standard, non-autoregressive RNN

The RNN is said to be *autoregressive* if the output y_{t-1} is fed back into the hidden state h_t either in replacement or in addition to x_t. Autoregressive RNNs can be straightforwardly used as language models. They represent a clear break from the Markovian limitations, as h_t depends on all previous hidden states all the way to h_1. Thus, y_t depends on the full history y_1^{t-1}.

In theory, RNNs are capable of modelling long-term dependencies between the sequence. In practice, not only there is a limited amount of information from the past that can fit into a h_t vector, but they also struggle to successfully learn the parameters to do it effectively. One of the most popular RNNs, the Long Short-Term Memory or LSTM [HS97], was one of the first recurrent neural topologies to achieve good capabilities to recall information from several steps back in the sequence, by carefully designing a computation graph that allowed the gradient to flow through the back-propagation algorithm with a minimal amount of activation functions in the middle that caused the gradient either to explode or to vanish. The compact forms of the equations for an LSTM cell are:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.13}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.14}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.15}$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.16}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.17}$$
$$h_t = o_t \circ \tanh(c_t) \tag{2.18}$$

Where σ is the sigmoid activation function $\sigma(x) = (1 + e^{-x})^{-1}$ and $\circ$ represents the pointwise multiplication of two vectors. Note that the vector c_t is undisturbed by activation functions. Figure 2.2 depicts the above equations in a graphical way.

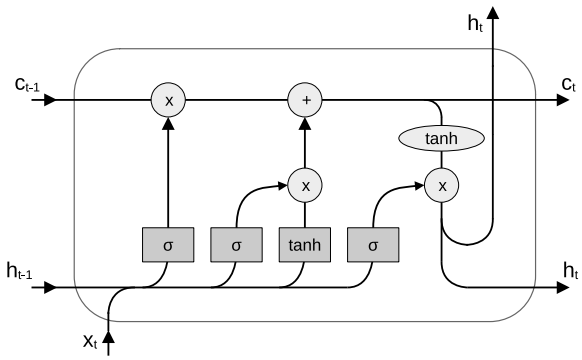

Figure 2.2: Representation of a LSTM unit. Blocks with σ or tanh represent a feed-forward followed by the corresponding activation function.

Although LSTMs remain in wide application and are the RNNs that were used for this work, the principle on which they work has been applied to generate a multitude of variants, ranging from minor modifications like the LSTM with peephole connections [GS00] to more extreme restructurings like the Gated Recurrent Unit or GRU [Cho+14].

2.4.2 Attention Mechanism

RNNs can perform an ample assortment of tasks with sequences, but struggle with problems where both the input and output are variable-length problems, also known as Sequence-to-Sequence (Seq2Seq) problems. Neural end-to-end approaches to MT or ASR are clear examples of these problems. The early approaches to these tasks used Encoder-Decoder models [SVL14] where an Encoder RNN would compress all the information in the source sequence into a fixed-length vector, and then a Decoder RNN would generate the output sequence conditioned on that vector:

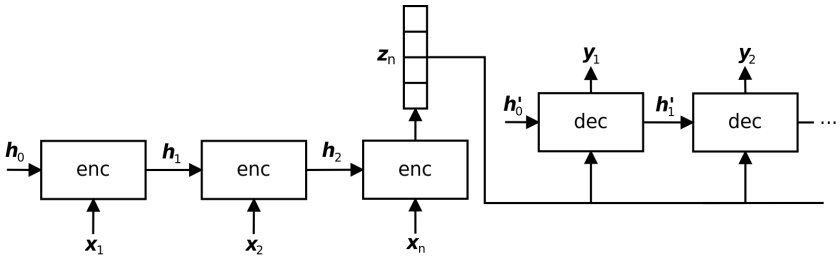

Figure 2.3: Sequential Encoder-Decoder architecture

Although this architecture can work on small problems, it is not well suited to long sentences with a very large vocabulary, as it becomes increasingly harder to fit all the relevant information about the source sentence in a single fixed-length vector representation. The *attention mechanism* was devised specifically to address this issue [Bah+15]. Instead of using a single vector condensing the whole input, such a vector is dynamically obtained at each decoder step representing certain parts of the input sequence. Specifically, it retains all outputs from the Encoder RNN and computes a weighted average of them to obtain a *context vector* c_t for the decoding step t. This allows the model to focus on particular source positions for the relevant information; a differentiable focus of attention so that a gradient-based optimization algorithm remains applicable.

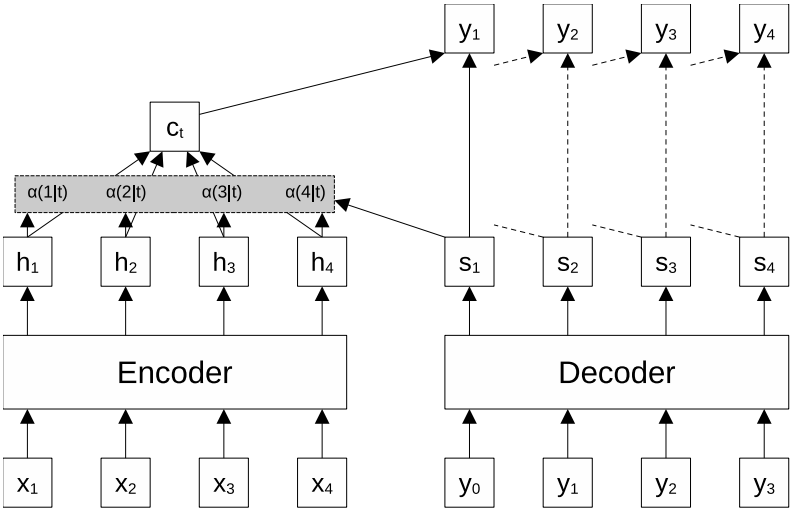

Figure 2.4: Attention mechanism with an autoregressive Encoder-Decoder architecture

We can proceed to write down the set of equations that define these models. We have:

- **RNN on the source side (encoder)**

$$h_1^N = \text{RNN}_{\text{encoder}}\left(x_1^N\right) \tag{2.19}$$

  Here, x_1^N represent the sequence of input vectors, which can be word embeddings or other kinds of input, and h_1^N represent the outputs from the encoder RNN for each position.

- **Context vector**

$$c_t = \sum_{n=1}^{N} \alpha(n|t) \cdot z_n \tag{2.20}$$

  This is the aforementioned weighted average of the source encodings. The $\alpha(i|t)$ is the set of weights. It is represented as a conditional probability distribution to show that all the values should add up to one and be non-negative. There are many ways to define such a function, we will review some of them later in this section.

- **RNN on the target side (decoder)**

$$s_t = \text{RNN}_{\text{decoder}}\left(y_1^{t-1}, c_1^{t-1}\right) \tag{2.21}$$

The decoder RNN takes as its input the previous outputs just like a normal autoregressive RNN, but also the history of context vectors so that the decoder has access to the information attended from the input.

- **Output vector**

$$y_t = \text{softmax}\left(\text{FF}(s_t, c_t)\right) \tag{2.22}$$

A feed-forward network with a softmax activation function is the final component that computes the output for the network at each decoding position.

Coming back to the attention weights $\alpha(n|t)$, they are computed using softmax function to ensure their differentiability, non-negativity and that $\sum_n \alpha(n|t) = 1$:

$$\alpha(n|t) = \frac{\exp\left(e(n,t)\right)}{\sum_{k=1}^{N} \exp\left(e(k,t)\right)} \tag{2.23}$$

Where $e(n,t)$ is the output of some *scoring function*. This score is expected to reflect the relevance of the encoded representation h_n to decide the final output y_t. There are several alternatives for the scoring function, most of which fit into two categories:

- **Additive attention**. It consists of a single-layer feed-forward network. It is called *additive* because the projections of both h_n and s_t are added together before the activation function, according to the regular workings of feed-forward nets. This kind of attention can be represented by the equation:

$$e(n,t) = v^T \cdot \tanh(Uh_n + Ws_t) \tag{2.24}$$

Where U, W, and v are the parameters of the function.

- **Multiplicative attention**. This variant computes the scores as a dot product of linear projections of h_n and s_t:

$$e(n,t) = (U_1 h_n)^T \cdot (U_2 s_t) = h_n^T W s_t \tag{2.25}$$

Where U_1 and U_2 are the parameters. Alternatively, a single matrix W can serve that role, as it can easily be seen that $W = U_1^T U_2$ in this case. In multiplicative attention, the $e(n,t)$ function can include a division by $\sqrt{d}$ where d is the dimension of the vectors, which is only there to normalize the variance of the result, facilitating gradient stabilization.

A minor variant of attention used in Transformer models [Vas+17] also includes a linear projection of the h_n vectors before the weighted average. In other words, it defines two distinct projections of h_n to so-called *key* and *value* vectors: the former for computing the attention scores, and the latter which will add up to the context vector.

2.4.3 Transformer

The Transformer is a neural architecture for sequences alternative to the RNNs [Vas+17]. They have revolutionized the field of Natural Language Processing (NLP), becoming the state-of-the-art over the last few years in many tasks such as language modeling or machine translation. Transformers are also being increasingly adopted in other fields like image recognition due to their highly parallelizable architecture leading to fast computation and their outstanding performance in sequential tasks.

Compared to RNNs, the Transformer model is based in a radically different principle. The dependency between elements of the sequence is not modelized by a recurrence, but by the attention mechanism. This way, the Transformer introduces the idea of *self-attention*, or attention over the elements of the same sequence. It also enhances the power of the attention mechanism, with the proposals of *multi-head attention* and *positional encodings*. We will briefly review each of these ideas and afterwards present the complete architecture of the Transfomer model.

- **Self-attention**. This proposal intends to replace the recurrency with the attention mechanism to handle sequences. There are two arguments for this replacement. The first relates to the speed of the operation. The self-attention mechanism is fully parallelizable, while the recurrency needs to be computed sequentially: the state for position $n-1$ is needed before the state for position n can be computed. The second refers to the long-term dependencies that the data in a sequence can have. For example, information about words far apart can still be relevant when determining singular or plural, or even semantic information. The recurrency has a harder time modeling long-term dependencies, while the self-attention

mechanism can attend to every position directly and without intermediate states degrading the information.

- **Multi-head attention**. Instead of computing a single attention function, a set of attention heads is defined and the attention mechanism is run separately on each head. It is expected that the model can learn to attend to different information from each subspace. The proposal projects the original vectors h_n and s_t to heads of lower dimensionality which dimensions add up to that of the original vector. This way, the mechanism becomes richer in terms of possibilities with the minimum sacrifice in terms of a higher computational cost.

- **Positional encodings**. The position and order of each vector is often of critical importance. The attention mechanism as previously defined is completely agnostic to the positions in the sequence: it scores each vector according to its contents, and does a weighted average. To be able to fully dispose of the recurrency, positional information must be included in the input vectors themselves. Information about the position can be incorporated in a similar way to information about the word. Similar to word embeddings, each position can be encoded by its own high-dimensional vector representation. Both representations can be combined into a single vector space by a simple sum if they are of the same dimensionality.

  Positional encodings can be learned by gradient descent much like word embeddings. [Vas+17] also proposes positional encodings can be fixed with an arrangement that lets a neural net based in linear projections extract all the relevant information. Based in sines and cosines, it exploits the property that a couple $(\sin(x), \cos(x))$ can be turned into $(\sin(x + k), \cos(x + k))$ via a linear projection — in this case, a rotation by k. Fixed positional encodings include several such pairs in a wide range of periods to be able to represent all possible dependency lengths. The exact definition for the proposed fixed positional encodings is:

$$\mathrm{PE}_{t,2i} = \sin\left(\frac{t}{C^{2i/d}}\right), \quad \mathrm{PE}_{t,2i+1} = \cos\left(\frac{t}{C^{2i/d}}\right) \tag{2.26}$$

  Where t is the position being encoded, i is the array index where the value is stored, and C is the maximum sequence length, typically high (e.g. 10000) so that the highest periods never come again close to the $t = 0$ disposition of $(\sin 0, \cos 0) = (0, 1)$. The fixed representation has the advantage that can better generalize to sequences of rarely seen lengths,

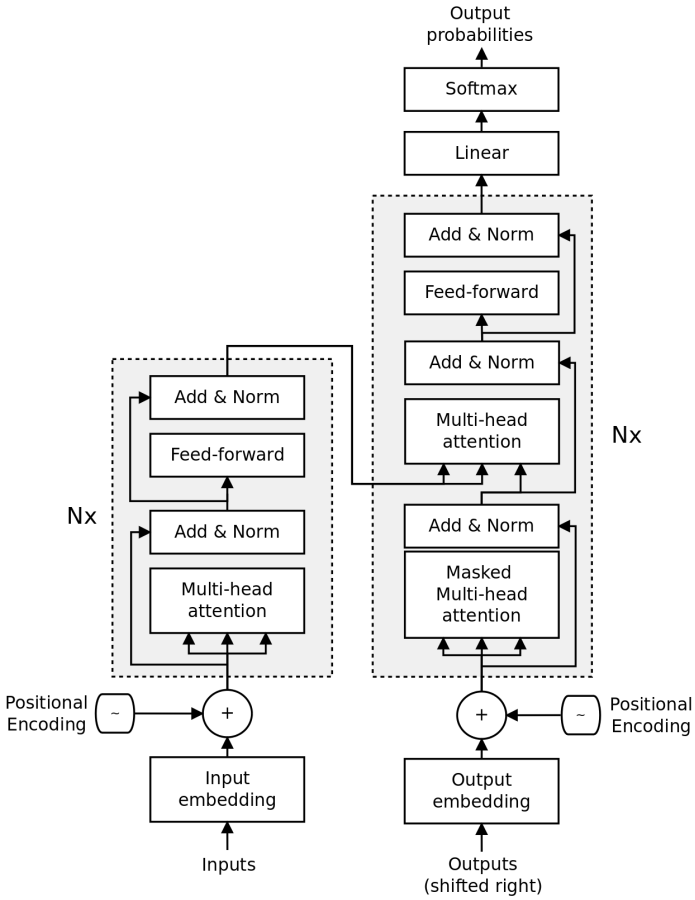

Figure 2.5: The full Transformer architecture. From [Mar22], adapted from [Vas+17]

or even can be uses with sequences of unseen lengths, which learned encodings are incapable of representing altogether.

The full Transformer architecture, depicted in Figure 2.5, has an encoder-decoder structure. It uses several stacked blocks comprising self-attention layers and feed-forward layers, including a residual connection after each layer.

2.5 Machine Translation

Machine Translation (MT) comprises the problem of automatically translating from a source sentence x in the original language into a target sentence y in the desired language. The probabilistic approach involves defining a model providing $p(y|x)$ and searching for the best possible y. In the traditional statistical approach to MT (SMT), the Bayes theorem is applied to decompose this formula into two separate sub-models:

$$\hat{y} = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y) \tag{2.27}$$

Where $p(x|y)$ represents the *translation model*, while $p(y)$ is an already familiar *language model* for the target language. The translation model, derived from tabulated counts of co-occurrences for sub-sentence units or *phrases*, is trained on parallel text data: collections of sentences coupled with their corresponding translations. This is called *Phrase-Based Machine Translation* (PBMT).

A more recent approach, Neural Machine Translation (NMT), has been leveraged to excellent results. Based on an array of encoder-decoder architectures, it now consistently outperforms traditional SMT methods and has become the state-of-the-art in MT. NMTarchitectures compute $p(y|x)$ directly, working as an autoregressive neural LM that is conditioned on the input sequence x:

$$p(y|x) = \prod_{t=1}^{T} p(y_t|y_0^{t-1}, x) \tag{2.28}$$

Where y_0 is always a special symbol designating the start of a sequence. To find the best possible sentence as defined by its probability, beam search algorithms are often used, as greedy decoding usually leads to suboptimal outcomes. In these algorithms, a set of *active hypotheses* is maintained at each position t. Each timestep, the next possible word for each of them is explored by consulting

the probabilities defined by the model over the whole vocabulary, and there is some heuristic pruning based on their probability to avoid an exponentially growing number of active hypotheses.

A problem with NMT models compared to PBMT is how to make use of the monolingual data. Given that most of the data is monolingual text, disregarding this resource is a recipe for suboptimal models. PBMT systems include a language model trained on monolingual texts, but NMT models are usually monolithic. To tackle this problem, a still unbeaten approach is to employ a MT system in the opposite direction to synthetically generate *backtranslations* from monolingual data in the target language [SHB16a].

Translation is inherently an open-vocabulary problem, but NMT models operate with a fixed vocabulary: the probability distribution at each step comes out of a softmax operation, each position of the output vector representing one particular word in that fixed vocabulary. Some approaches have been made to build NMT systems capable of open-vocabulary translation, most of them around the idea of breaking up the vocabulary into sub-word units that can be combined to form new words. Byte Pair Encoding or BPE [SHB16b] was a pioneer work in this direction and is also used in this work, but there are other alternatives based on similar fundamentals [Kud18; PEV20].

A metric to objectively and automatically assess the performance of an MT system is hard to obtain, as a single source sentence usually has multiple valid translations. One of the most popular automatic metrics is the BLEU, or *bilingual evaluation understudy*, based in similarity with a reference transcription [Pap+02] (higher is better). Another metric that was used in this work it the TER, or *translation edit rate*, measuring the minimum edit distance to transform the output hypothesis into the human-translated reference, including insertions, substitutions, deletions, and phrasal shifts as edit operations [Sno+06] (lower is better).

Transformers have now become the standard choice in NMT, after consistent and solid performance increases over previous NMT models such as attention-based RNN encoder-decoder models like the one described in section 2.4.2.

2.6 Automatic Speech Recognition

The problem of Automatic Speech Recognition (ASR) is to automatically obtain the transcription from an acoustic signal. As in MT, the problem can be formally described as a classification problem, defining a model $p(y|x)$ and searching the space of possible ys for the one with highest probability. In this case, y is a sequence of words and x a sequence of vectors characterizing the acoustic signal. Like MT, this problem has also been traditionally approached by decomposition following the Bayes theorem:

$$\hat{y} = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y) \tag{2.29}$$

Where $p(y)$ is modelled by a *language model* and $p(x|y)$ by an *acoustic model* (AM). The language model is trained and serves a role similar to those we have already seen. The AM $p(x|y)$ estimates the probability that the observed acoustic sequence x has been generated by the sentence y. As such, it serves a double role: it needs to provide not only information about how the words can be pronounced, but also about their alignment to the observed signal.

To handle the acoustic signal, it is converted into a sequence of acoustic feature vectors. These vectors should include all the needed information to characterize the signal and discriminate between the spoken words. Typically, a feature vector is obtained every e.g. 10 milliseconds. A widely employed method to extract these acoustic vectors is the *Mel-Frequency Cepstral Coefficients*.

Habitually, AMs are modelled via *Hidden Markov Models* (HMM) [Lee88; Hun90] together with a pronunciation lexicon. The pronunciation lexicon contains the transcription of each word into phonemes. A particular HMM is defined for every hypothesis, concatenating the finite state representation of its phonemes. This construction is represented in figure 2.6. When computing $p(x|y)$, we transit from one state to the next following its *transition probabilities*. Each state emits an acoustic vector according to its own *emission probability*.

Transition probabilities are scalar values in a transition matrix. Traditionally, emission probabilities were modelled as *gaussian mixture models* (GMM). These GMMs define, for each state q, a continuous probability distribution over the acoustic vector space $p(x|q$. Both transition and emission probabilities can be estimated with the well-known Baum-Welch algorithm for HMMs,

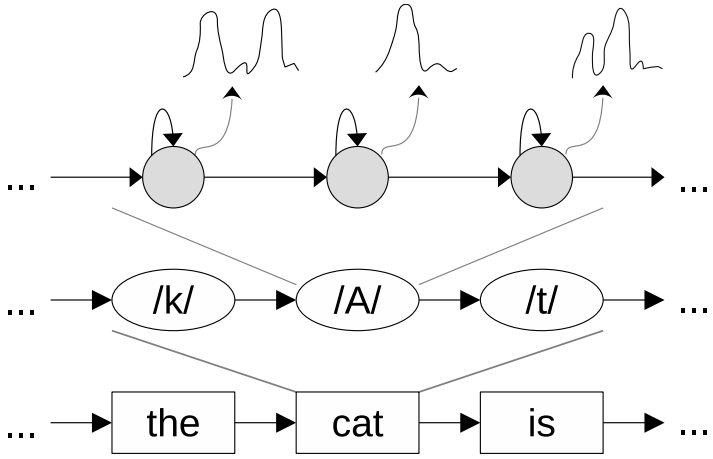

Figure 2.6: Acoustic HMM model for a hypothesis containing "the cat is" as part of the sentence. The HMM states are on the top of the diagram.

an instance of the more general Estimation-Maximization (EM) algorithm; or they can be estimated following the Viterbi approximation.

More recently, GMM emission probabilities have been successfully replaced with the better-performing neural models. These models do not directly model $p(x|q)$ but rather take the acoustic vector x as their input and output the distribution over the states $p(q|x)$. To be able to integrate these neural models with HMM-based ASR systems, the Bayes rule is applied to recover the usable $p(x|q)$:

$$p(x|q) = \frac{p(q|x)p(x)}{p(q)} \tag{2.30}$$

HMM-based ASR systems using neural nets for this role are known as *hybrid systems*, as they use a combination of both neural nets and more traditional HMMs for acoustic modelling [BM94]. In contrast to HMM-GMM and hybrid systems, we find the *end-to-end* systems, which are monolithic neural models directly modelling $p(y|x)$, the probabilities over the output space given the acoustic sequence. Lately, Transformer-based end-to-end systems are beginning to achieve levels of performance comparable to hybrid systems, even surpassing them in some specific tasks [Mia+20b; Wan+20; Gul+20; VA21].

The work on ASR that is presented in this thesis focuses on streaming-enabled systems. Most published work on streaming-capable ASR systems is done with end-to-end systems, which are more easily adapted to this setup [Lu+20; Che+21; Xie+22; Xue+22; Yu+21; TKW21]. In contrast, in this thesis we work not with end-to-end but with hybrid systems for the streaming setup. This serves as a different line of inquiry to what is primarily being pursued in the field and also has promising prospects: as of the time this work is being done, the state-of-the-art ASR results are still obtained using hybrid systems, which wil be reflected throughout the international competitions presented during this thesis.

ASR performance is assessed using the *word error rate* or WER. This metric is derived from the minimum edit distance from the system output, or hypothesis, to a reference human transcription:

$$\text{WER} = \frac{I + D + S}{N} \tag{2.31}$$

Where I, D, and S correspond respectively to insertions, deletions, and substitutions; while N indicates the total number of words in the reference. A WER of 23% means that 23% of the words have been wrongly recognized, be it because they are missing, don't belong, or have been transcribed as a different word.

Chapter 3

Machine Translation Systems for News and Related Languages (WMT)

3.1 Introduction

The first goal of this thesis is to build state-of-the-art MT systems. For this, we participated in the WMT18 German→English news translation shared task and in the WMT19 Portuguese↔Spanish news translation shared task, specifically designed for related languages.

Neural Machine Translation (NMT) has made great advances over the last decade, and in particular it came to consistently outperform Phrase-Based Machine Translation (PBMT) and PBMT-NMT combinations since the 2016 WMT shared news translation task [Boy+16]. In this chapter, we introduce a NMT model architecture that was being explored in our research group, which we compared to the novel and better-performing Transformer architecture, which has been shown to provide the state-of-the-art performance in MT while requiring relatively short times to train.

We also describe our work on parallel-corpus preprocessing and filtering, as well as data augmentation, aspects that have gained importance with the ever-increasing amount of available data of varying quality crawled from the Internet. In particular, WMT18 presented the challenge of the addition of the mugh larger and noisier parallel corpus ParaCrawl. For the WMT19 Related Languages task Portuguese↔Spanish, we achieved the best results of all participants on both directions with a domain-adapted Transformer model.

3.2 2D Alternating RNN

In this section, we describe the general architecture of the 2D alternating RNN model. This architecture was being explored by our research group in the framework of 2D translation models [Kal+15; BBN18; EBV18]. These architectures are based on the premise that translation is a fundamentally two-dimensional problem, where each word of the target sentence is liable to be explained by every word in the source sentence in some way. Two-dimensaional NMT models define the distribution $p(y_t|x_0^N, y_0^{t-1})$ by jointly encoding the source sentence x_0^N and the output history y_0^{t-1}, whereas the usual encoder-decoder architectures encode them separately.

Typical 2D RNN architectures employ a custom LSTM cell with two previous states instead of one, each coming from its own dimension. This introduces dependencies in the computation graph which force the computation to be done in quadratic time with respect to the input, both for the training as well as the recognition process. We proposed an architecture, depicted in Figure 3.1, defining a two-dimensional RNN translation model leveraging regular 1D recurrent cells, such as LSTM or GRUs without any modification. This makes the computation time linear instead of quadratic, assuming maximum parallelization is applied.

Similarly to encoder-decoder NMT models, the 2D Alternating RNN MT architecture obtains a context vector c_t for each output position t. This context vector is linearly projected to the vocabulary size and a softmax function is applied to obtain the probability distribution for the next word $p(y_t|X_0^N, y_0^{t-1}$. The method to obtain this context vector, however, is different.

Firstly, a two-dimensional grid is defined with one dimension for the source sentence and the other dimension for the target sentence. Each cell s_{tn} in the grid corresponds to the pair (y_t, x_n). From this grid, we define a layer-like structure or *block*, where the ith block has such a grid s_{tn}^{i-1} as its input, and another one as its output s_{tn}^{i}. The input to 2D model is obtained by concatenating the word embeddings from the source and target sentences:

$$s_{tn}^{0} = \begin{bmatrix} y_t \\ x_n \end{bmatrix} \tag{3.1}$$

Inside a block there are two layers containing recurrent cells. The first, a bidirectional RNN, goes along the source dimension; while the second, a unidirectional RNN, goes along the target dimension. They process each row, or

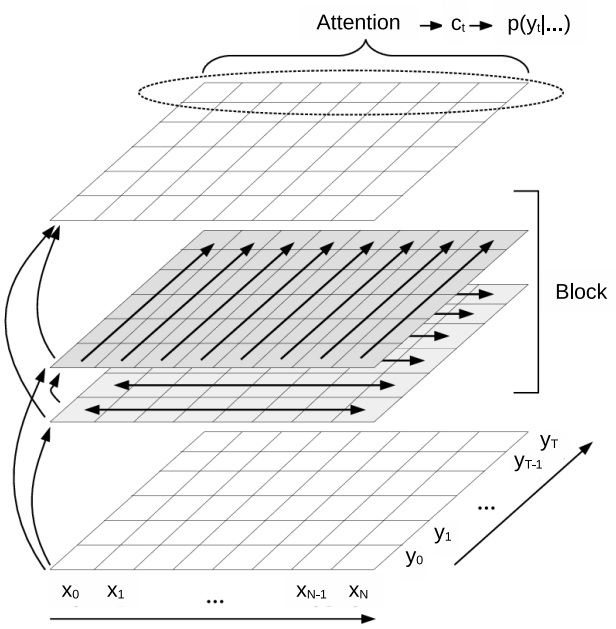

Figure 3.1: The 2D alternating RNN architecture. White grids on the top and bottom represent the input/output of a block. Arrows inside grey grids represent the directions of the RNNs, while the vertical arrows on the left depict how the layers are interconnected. Arrows on the bottom indicate the source and target dimensions.

column, independently of one another. As depicted in Figure 3.1, the grid is an input to both layers. The first layer is also connected to the second layer, while both output to the next grid s^i_{tn}.

The context vector is obtained from the last block leveraging a form of attention mechanism. This scores the vectors according to a learned linear scoring function followed by a softmax to extract the weights. These weights are used to perform a weighted sum and recover the context vector:

$$c_t = \text{Attention}([s^I_{t0}, \ldots, s^I_{tN}]) \tag{3.2}$$

This model was evaluated in the Related News translation task, and the experimentation is described in section 3.4.

3.3 German→English news translation (WMT18)

In this section we describe the MT system built by the MLLP for the news translation shared task of the EMNLP 2018 Third Conference on Machine Translation (WMT18) in the German→English direction. We also describe our work on parallel corpus preprocessing and filtering, an aspect that gained importance in WMT18 due to the addition of the large and noisy ParaCrawl corpus to the task. Furthermore, regarding data augmentation, we report how we extended the supplied parallel dataset with data based on automatic backtranslations from the provided target language monolingual corpora.

3.3.1 Data preparation

In this section, we described the techniques we used to leverage the provided WMT German↔English data (both parallel and monolingual) to improve our MT system results: corpus preprocessing, corpus filtering, and parallel data augmentation via backtranslation. These tasks acquired relevance in WMT18, with the addition of the new noisy ParaCrawl parallel corpus crawled from the internet, which sextuplicates the amount of available parallel data in this language pair with respect to the previous editions. There are ∼36 million sentence pairs in ParaCrawl alone, versus ∼6 million in the rest of the corpora taken together, for a total of ∼42 million sentence pairs. This distribution is illustrated in table 3.1.

Table 3.1: Size by corpus of the WMT18 parallel dataset

Corpus	Sentences (M)
News Commentary v13	0.3
Rapid (press releases)	1.3
Common Crawl	1.9
Europarl v7	2.4
ParaCrawl	36.4
WMT18 total	42.3

While it is true that the large size of the ParaCrawl parallel corpus makes it a valuable resoure for MT system training, it contains much more noise compared to the rest of the WMT18 corpora: misaligned sentences, wrong languages, meaningless sentences, and other artifacts that may hinder system training for the purpose of German→English translation. In our experiments, we observed that preprocessing and filtering this corpus is necessary to turn it into training data that can effectively increase the translation quality of the model. In fact, using the ParaCrawl corpus "as is", we even observed a degradation in all metrics of quality, as described in detail in Section 3.3.3.

With respect to data augmentation, the usage of relevant in-domain monolingual data has shown to be key in order to improve NMT systems [SHB16a]. We leveraged the provided monolingual resources in WMT18 via backtranslations to increase the accuracy of the system.

Corpus preprocessing

We performed a standard preprocessing as suggested by the WMT18 organization [18] using the provided scripts, comprising punctuation normalization, tokenization, cleaning, and truecasing via standard Moses [Koe+07] scripts. Additionally, we removed from the training corpora any sentence containing characters outside the Latin UTF interval (u0000-u20AC) plus the euro sign (€). This decision enabled us to reduce the vocabulary size by eliminating sentences belonging to languages other than German or English, or that are not useful for the translation of online news between them.

Corpus filtering

Our aim in data filtering is to leave out noisy sentence pairs from the parallel corpora. That is, meaningless sentences or sentences in other languages that were present in the ParaCrawl corpus. To this end, we trained two separate 9-gram character-based language models, one for German and another one for English. These language models were trained on the newstest2014 development set using the SRI Language Modelling Toolkit [Sto+11]. Using these language models, we scored the sentences in the whole WMT18 corpus, including ParaCrawl, according to their perplexity value; in a manner similar to the techniques described by [Yas+08; FGK10; AHG11].

Accordingly, a score was obtained for each sentence pairs combining their separate perplexity scores s, t with the geometric mean $\sqrt{s \cdot t}$. The geometric mean of two perplexities is approximately the perplexity of their concatenation, and exactly if both are the same length. We selected subsets of different sizes from the WMT18 corpus taken as a whole, taking the n lowest-scored sentence pairs. Filtering out high-perplexity pairs simultaneously attains two goals: to filter out the noise and to provide some degree of domain adaptation, since the scoring models were trained with in-domain data.

Exploiting monolingual data

To make the most out of all the provided data for the task, we augmented the WMT German↔English parallel dataset with synthetic source sentences, automatically backtranslating from the provided target language monolingual corpora, following the approach outlined by [SHB16a].

In particular, we trained an English→German NMT system based on our best configuration for German→English. We then used this system to generate our synthetic source German sentences from a subset of the WMT18 English monolingual corpora. This provided us with a significant amount of new sentence pairs to use as in-domain synthetic training data.

3.3.2 System description

We decided to build an NMT system based on the Transformer architecture [Vas+17]. We opted for a pure NMT system due to the solid advances made by this field over the previous years, which led this kind of systems to systematically outperform both the more traditional PBMT systems and PBMT-NMT combinations, as outlined in Section 3.1. In particular, our choice of the Transformer architecture, was based on its state-of-the-art results and relatively short computational time for its training. To build our Transformer models, we used the Sockeye NMT framework [Hie+18].

We based our systems on the Transformer "base" configuration with 65M parameters over the "big" configuration with 213M parameters, due to its shorter training time and smaller need of computational resources, in exchange for a minor expected decrease in translation quality as described in [Vas+17]. Training time was a relevant constraint to be able to complete our experiments in time for participation in the shared task. Thus, our models used 6 self-attentive layers on the encoder and decoder, with a model dimension of 512 units and a feed-forward dimension fo 2048 units.

During training, we applied 0.1 dropout and 0.1 label smoothing, the Adam optimization scheme [KB15] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate annealing: we set an initial learning rate of 0.0001 and scaled it by a factor of 0.7 whenever the validation perplexity did not improve in 8 consecutive checkpoints, each checkpoint being validated every 2000 parameter updates. The system was trained with a batch size of 3000 tokens and a maximum sentence length of 75.

For our internal experiments, our systems were trained after applying 20k BPE operations [SHB16b], but we increased this amount to 40k BPE operations for our final submissions. The final system consists of an ensemble of 4 independent training runs of our best configuration, based on a linear combination of the individual model probabilities at each timestep.

3.3.3 Experimental evaluation

In this section, we outline the experimental setup, report our experiments and results on corpus filtering, describe our setup for data augmentation via back-translation, and discuss our final German→English NMT system evaluation and results.

Experimental setup

For our experiments, we used newstest2015 as the development set and newstest2017 as the test set. We also report the results obtained with the official test set newstest2018. We evaluated our systems using the BLEU [Pap+02] and TER [Sno+06] measures. All reported scores were computed with the system output formatting as indicated by the instructions provided by the WMT18 organization.

Results on corpus filtering

Here, we show and interpret the results obtained with the corpus filtering techniques described in section 3.3.1. Table 3.2 summarizes the translation quality results obtained with different subsets of the WMT18 parallel subset. We can see that using the full 42M sentence pairs, including the ParaCrawl corpus "as is", leads to significant performance degradation in all quality metrics compared to using the 6M sentence pairs of the WMT18 corpus excluding the ParaCrawl. We also observe that using an excessively restricted training set when applying filtering (5M sentence pairs) also leads to degradation in comparison to the WMT18 corpus without ParaCrawl. Otherwise, our filtering approach is effective at selecting useful training data from ParaCrawl: filtered training datasets over the original 6M sentence pairs provide significant improvements in quality, even with a small increase to 7.5M pairs. At the other extreme, going over 15M filtered sentence pairs meant quality metrics started degrading again.

Table 3.2: Results of 9-gram character-based LM filtering, by number of selected sentence pairs

	newstest2015		newstest2017		newstest2018	
Subset (# of sentence pairs)	BLEU (%)	TER (%)	BLEU (%)	TER (%)	BLEU (%)	TER (%)
Full WMT18 dataset (42M)	20.6	71.1	21.3	70.2	26.2	64.2
minus ParaCrawl (6M)	31.1	55.4	32.0	54.8	39.1	46.3
Filtered corpus (5M)	30.3	56.3	31.4	55.5	38.7	46.5
Filtered corpus (7.5M)	32.8	54.0	33.7	56.5	41.5	44.5
Filtered corpus (10M)	33.0	53.7	34.5	52.9	42.2	43.7
Filtered corpus (15M)	33.4	53.2	34.3	52.7	42.2	43.6

As Table 3.2 shows, the best results on newstest2015 and newstest2017 were obtained with the 10M and 15M subsets. As the quality was very similar in both cases, we opted to use the 33% smaller 10M subset to train our systems on, for its significantly faster convergence in system training time. Thus, the 10M subset filtered corpus is used as a basis for the subsequent work.

Bactranslation setup

Here, we detail how we augmented the WMT18 German↔English parallel training dataset based on the backtranslation technique. First, we trained an English→German NMT system using the same configuration as our best German→English system as described in Section 3.3.2, and trained it with the 10M sentence pairs filtered subset of the WMT18 parallel dataset. The resulting English→German NMT system obtained 27.4 BLEU on newstest2017. Using this system, we translated into German a random sample of 20M English sentences from NewsCrawl 2017 – the most recent in-domain corpus among the provided WMT18 monolingual corpora. This provided us with 20M synthetic in-domain sentence pairs of German→English as additional training data. This was added to the 10M selected sentence pairs, comprising a total of 30M sentence pairs as an augmented training corpus which was used for the final systems.

Final system evaluation and results

We will now summarize the most relevant results for the final German→English NMT system trained for WMT18. These are shown in Table 3.3. Our baseline model was trained excluding the ParaCrawl corpus from the training data and used 20K BPE operations. The first step to improve over this baseline system was filtering the full WMT18 corpus including ParaCrawl, as explained in Section 3.3.3. Table 3.3 only shows the results of the best-performing system from the filtering evaluation. The 10M filtered corpus provides an improvement of 2.5 BLEU and 1.7 TER over the baseline model in the newstest2017 test set, showing how our data filtering approach was effective to extract useful sentences from the noisy ParaCrawl corpus in order to improve the performance of our system.

Table 3.3: Summary of German→English system evaluation results

	newstest2015		newstest2017		newstest2018	
System (# of sentences)	BLEU (%)	TER (%)	BLEU (%)	TER (%)	BLEU (%)	TER (%)
WMT18 corpus (6M)	31.1	55.4	32.0	54.8	39.1	46.3
Filtered corpus (10M)	33.0	53.7	34.5	52.9	42.2	43.7
+ backtransl. (10M+20M)	34.3	52.0	35.9	51.2	44.7	41.1
Ensemble (x4)	34.6	51.9	36.2	51.0	45.1	40.8

For the final systems, we added 20M synthetic sentence pairs with backtranslations as described in Section 3.3.3. For balance, we oversampled the 10M filtered training set by duplicating it, giving us a final training set of 40M sentence pairs. We also increased the number of BPE operations from 20K to

40K. A single NMT system trained with this configuration yielded 35.9 BLEU and 51.2 TER in the newstest2017 test set, representing a significant improvement of 1.4 BLEU and 1.7 TER over the model without the synthetic data. This is explained by a combination of the additional sentence pairs and the increase in vocabulary size. Training a system with this final configuration took approximately 120 hours on a single-GPU machine with an Nvidia GeForce GTX 1080 Ti.

Finally, our primary submission for WMT18 consisted of an ensemble of 4 independent training runs with this final configuration, resulting in 36.2 BLEU and 51.0 TER in newstest2017, and 45.1 BLEU and 40.8 TER in the official test set newstest2018.

3.3.4 Comparative results

Here we will review the primary submission results of all participants in the Shared Task. Table 3.4 shows the results of all participants in the German→English news translation shared task. The table contains the results in the official score *Ave. z* [Boj+18]. To obtain this score, human direct assessments were made and then standardized according to each individual assessor's mean and variance. The final score for the system is computed as the average of its segment scores. In addition to the *Ave. z* score, the corresponding BLEU figures were included in the table whenever the information was publicly available.

The conference organization ranked the participant systems by clusters according to the official score *Ave. z* and identified by grouping systems together so that they significantly outperform, according to the Wilcoxon rank-sum test, all other systems in a lower rank.

Note that as the BLEU scores were not computed by the organization, there may be some inconsistencies in how the participants assessed their own submissions, which could explain the apparently anomalous result of JHU reporting a BLEU much lower than other participants with a worse Ave. z score or that were even in a lower rank.

Table 3.4: Primary submission results, in BLEU, of the German→English news translation shared task in the hidden test set *newstest2018*

Rank	Team	BLEU (%)	Ave. z
1	RWTH [Gra+18]	48.4	0.413
	UCAM [SGB18]	48.0	0.395
	NTT [MSN18]	46.8	0.359
	ONLINE-B	—	0.346
	MLLP	**45.1**	**0.321**
	JHU [KDT18]	37.0	0.317
	UBIQUS-NMT	—	0.315
	UEDIN [Had+18]	43.9	0.261
11	LMU-NMT [Huc+18]	42.2	0.162
	NJUNMT-PRIVATE	—	0.149
13	ONLINE-G	—	–0.074
14	ONLINE-F	—	–0.296

3.3.5 Conclusions

The MLLP group of the Universitat Politècnica de València participated in the WMT18 news translation shared task in the German→English direction. Our primary submission was an ensemble of four NMT models based on the Transformer architecture, trained using a combination of filtered parallel data and synthetic in-domain data using backtranslations from monolingual corpora.

Our results confirm the capability of the Transformer NMT architecture to attain highly competitive MT results with a relatively low training and inference computational cost. We also showed the relevance of data curation, in this case in the form of filtering and data augmentation leading to significant improvements in translation quality.

3.4 Related Language translation task (WMT19)

In this section we describe the MT systems developed by the MLLP research group for the Related Languages Translation Shared Task of the ACL 2019 Fourth Converence on Machine Translation (WMT19). For this task, we participated in both directions of the Portuguese↔Spanish language pair using NMT models. We also tried out our 2D Alternating RNN NMT architecture. We report results for this approach and the Transformer [Vas+17] NMT ar-

chitecture. A domain-adapted Transformer system achieves the best results of all submitted systems on both directions of the shared task.

3.4.1 Baseline systems

Here we describe the training corpora as well as the baseline model configurations. Two different model architectures were used: the Transformer architecture and our proposal of 2D alternating RNN. The performance was evaluated by the BLEU metric [Pap+02].

Data preparation

The training data consisted of the JCR, Europarl, news-commentary, and wikititles corpora.

Table 3.5: Statistics of the datasets used to train the Spanish↔Portuguese MT systems

Corpus	Sent.(K)	Words(M)		Vocab.(K)	
		Es	Pt	Es	Pt
JCR	1650	42	40	264	264
Europarl	1812	53	52	177	156
news	48	1	1	49	47
wikititles	621	1	1	292	295
dev	1.5	0	0	6	6
test	1.5	0	0	6	6

The data was processed using the standard Moses pipeline [Koe+07] with the standard scripts for punctuation normalization, tokenization, and truecasing. We applied 32K BPE [**sennrich2017bpe**] operations learned jointly over the source and target languages, including into the vocabulary only those tokens occuring at least 10 times in the training data.

Transformer baseline models

For the Transformer model, we used the "base" configuration consisting in 512 model size, 8 attention heads, and 2048 feed-forward size; trained on a single GPU machine. The batch size was 4000 tokens, carrying out gradient accumulation by temporarily storing gradients and updating the weights every 4 batches, allowing for an effective batch size of 16000 tokens. We used dropout [Sri+14] with probability 0.1 of dropping a single connections, and

label smoothing [Sze+16] where we distribute 0.1 probability weight among the target vocabulary at each label. We stored a checkpoint every 10000 updates, and for inference used the average of the last 8 checkpoints. We used the Adam optimizer [KB15] with $\beta_1 = 0.9$, $\beta_2 = 0.98$. The learning rate was updated following an inverse square root schedule, with an initial learning rate of $5 \cdot 10^{-4}$ and 4000 warm-up updates. The training was performed using the Fairseq toolkit [Ott+19].

2D alternating RNN baseline model

For the 2D alternating RNN models, we used GRU for the recurrent cell, with an embedding size of 256 and 128 as the size of each layer of the block. The model consisted of a single block. The batch size was 20 sentences, with a maximum length of 75 subword units. We used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$. We used learning rate annealing with an initial learning rate of 10^{-3}, halving it after 3 checkpoints without improvement in development perplexity. A checkpout was saved every 5000 updates. The model was built using our own toolkit. Due to time constraints, the 2D alternating RNN model was only trained for the Portuguese→Spanish direction.

Baseline Results

Table 3.6 shows the evaluation results for the Portuguese→Spanish and Spanish→Portuguese systems. For the Portuguese→Spanish direction, the Transformer model obtained 57.4 BLEU in the open test set, and 51.9 in the hidden test set of the competition. The 2D alternating RNN model achieved 55.1 and 49.7 BLEU, respectively. These results showed how a very preliminary version of the 2D alternating RNN model was able to approach competitive results for this task. It is worth noting that this was achieved with a model with significantly fewer parameters: 14.9M compared to its Transformer counterpart with 60.2M parameters.

Table 3.6: Baseline BLEU scores on the Portuguese→Spanish and Spanish→ Portuguese tasks

	Pt→Es BLEU (%)		Es→Pt BLEU (%)	
System	test	test-hidden	test	test-hidden
Transformer	57.4	51.9	51.2	45.5
2D alternating RNN	55.1	49.7	-	-

3.4.2 Fine-tuning

NMT models perform best when trained with in-domain data. However, most available parallel sentences came from corpora belonging to other domains: either institutional documents or internet-crawled general content, not from news. Therefore, we find a significant domain mismatch between the domains of the training data and the test data. In such cases, even small amounts of in-domain training data can be leveraged to improve system performance. This is done by carrying out an additional specific training step, referred to as the fine-tuning step, using the in-domain data after the main training has finished. Fine-tuning has been widely used to adapt general-purpose models trained with general-domain data to specific domains using only small amounts of in-domain data [LM15; SHB16a]

To check how big the mismatch between out-of-domain and in-domain data was, we trained two language models: one using only the general training data from Table 3.5, and another one using only the in-domain development data. Both LMs were 4-gram language models trained using the SRI Language Modelling Toolkit [Sto+11]. We then computed the perplexity of the open test set using these two language models. The out-of-domain LM obtained a perplexity of 298.0, whereas the in-domain LM achieved a perplexity of 81.9. This striking difference exposes the significant mismatch between train and test data, supporting the idea of carrying out fine-tuning.

In order to understand the impact and behaviour of the fine-tuning process, we analyzed the model's performance as a function of the number of fine-tuning epochs. Figures 3.2 and 3.3 show the results of this procedure for the Portuguese→Spanish and Spanish→Portuguese tasks, respectively. In both language pairs, the first epochs are most beneficial to system performance, while additional epochs bring diminishing returns until the curve flattens.

3.4.3 Comparative results

Here we will review the primary submission results of all participants in the Shared Task. Our primary submission was the fine-tuned Transformer system for both tasks. The submission was made with the checkpoint that achieved the best performance on the fine-tuning dev data. Table 3.7 shows the results of the Portuguese→Spanish and Spanish→Portuguese shared tasks. The results are evaluated in terms of BLEU and TER.

In both tasks, our system outperformed all other participants by a significant margin. In the Portuguese→Spanish task, our submission outperforms the next

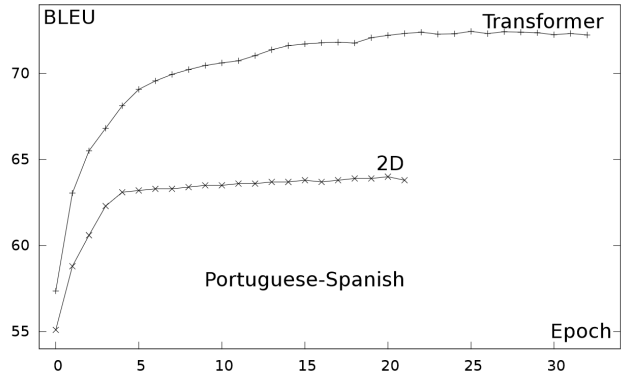

Figure 3.2: BLEU scores in the open test set as a function of the number of fine-tuning epochs on the Transformer and 2D alternating RNN models for the Portuguese→Spanish task

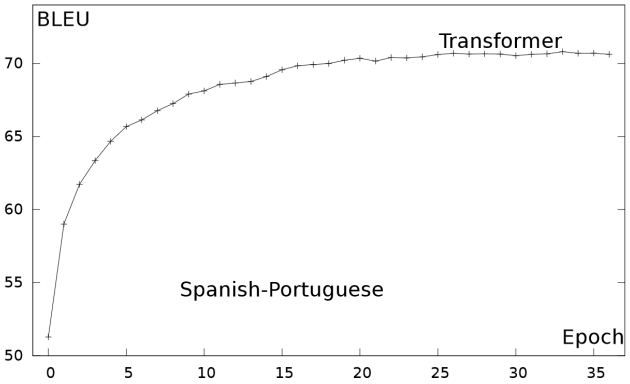

Figure 3.3: BLEU scores in the open test set as a function of the number of fine-tuning epochs on the Transformer model for the Spanish→Portuguese task

Table 3.7: Primary submission results of the Portuguese→Spanish and Spanish→Portuguese shared tasks in the hidden test set

Portuguese→Spanish		
Team	BLEU (%)	TER (%)
MLLP	**66.6**	**19.7**
NICT	59.9	25.3
U. Helsinki	58.4	25.3
Kyoto U.	56.9	26.9
BSC	54.8	29.8
UBC-NLP	52.3	32.9

Spanish→Portuguese		
Team	BLEU (%)	TER (%)
MLLP	**64.7**	**20.8**
UPC-TALP	62.1	23.0
NICT	53.3	29.1
U. Helsinki	52.0	29.4
UBC-NLP	46.1	36.0
BSC	44.0	37.5

best system by 6.7 BLEU and 5.6 TER. Similarly, in the Spanish→Portuguese taks, our system is at a distance of 2.6 BLEU and 2.2 TER from the second best. We attribute this success to the domain adaptation via fine-tuning, using part of the competition's development data as in-domain training data.

3.4.4 Conclusions

We approached the similar language translation task similarly to other translation tasks. NMT models and specifically the Transformer architecture fared well in this task without specific adaptations to the similar-language setup, obtaining excellent results with BLEUs around 65%.

Applying a domain adaptation approach based on fine-tuning on in-domain data, we achieved the best results among all participants by a significant margin in both Portuguese→Spanish and Spanish→Portuguese. We believe these results are explained by the domain mismatch between the training and test data, and not specific to the similarity between Spanish and Portuguese.

We tried out the 2D alternating RNN model architecture, testing it in the Portuguese→Spanish task. With relatively small embedding and hidden unit vector sizes and a shallow architecture, there was a gap between its performance and that of the fleshed-out Transformer model.

3.5 Conclusions

Within the framework of the MLLP research group, we participated in the WMT18 and WMT19 machine translation shared tasks. In WMT18, we participated in the news translation task in the German→English direction with an ensemble of four NMT Transformer models. With an emphasis on filtering and data augmentation, we attained highly competitive MT results and showed the relevance of data curation, leading to significant improvements in translation quality.

In WMT19, we participated in the similar language translation task with a Transformer model in both directions of Portuguese→Spanish and Spanish→Portuguese. With fine-tuning on in-domain data and a single Transformer model, we achieved the best results among all participants in both directions. We also tested our 2D alternating RNN architecture proposal, finding a performance gap between itself and the fully fleshed-out Transformer architecture.

The work described in this chapter was done in collaboration with other members of the MLLP research group, in particular with Javier Iranzo-Sánchez, and resulted in two publications in which each is recognized as the first author, as detailed in the conclusions of the thesis in Chapter 6.

Chapter 4

Streaming Automatic Speech Recognition with Transformer Language Models

4.1 Introduction

The second goal of this thesis is to improve the state-of-the-art in streaming ASR technology. Previous progress in the ASR field has brought great attention to the streaming tasks due to their immense applicability. The streaming setup for ASR comes with additional challenges and restrictions. First, the automatic system must deliver the output in real time as the continuous audio stream is processed. Second, the system cannot defer the output transcription until the end of the acoustic stream. As a consequence, the main challenge is finding the best way to come the closest possible to the accuracy of state-of-the-art off-line ASR under the streaming conditions. That is, delivering continuous output transcriptions within a short delay, i.e. not much longer than a second, with respect to the acoustic audio stream.

Lately, many authors are currently exploring the use of monolithic neural networks, also known as end-to-end systems, both to ASR in general and to streaming ASR in particular [MHL20; Zha+20; Mia+20a; Zey+19]. End-to-end systems are comparatively simple and easy to build from widely available deep learning toolkits and require minimal human expert knowledge. However, despite their simplicity and promising prospects, it was still unclear whether or not they will soon definitively surpass state-of-the-art hybrid systems combin-

ing separate acoustic and language neural models under the more conventional HMM paradigm.

This chapter focuses on language models for hybrid ASR systems for the purpose of building streaming ASR systems. Following our previous work on real-time one-pass decoding with LSTM LMs [Jor+19] and further realization of a streaming one-pass decoder achieving live transcription with minimal accuracy loss with a delay of less than a second [Jor+20], the work described in this chapter brings further improvements to the performance of this architecture, inspired by the latest developments reported in the literature for language modelling in the off-line setup [Iri+19], by leveraging the power of Transformer as language models. Our empirical study on the LibriSpeech and TED-LIUM tasks shows that the introduction of Transformer LMs (TLM) lead to top, state-of-the-art recognition accuracy rates and latencies under streaming conditions.

This chapter is organized as follows. Section 4.2 describes the decoder architecture we designed and its specific adaptations enabling streaming capabilities in hybrid ASR systems. Section 4.3 presents the alterations made to the Transformer language models to adapt them to a streaming use. Section 4.4 contains the description of the experimental study and its results, as well as the corresponding analysis. Finally, Section 4.5 offers some conclusions to the chapter.

4.2 Streaming one-pass decoder

We proposed the direct use of neural LMs in a one-pass History Conditioned Search (HCS) strategy [Nol17]. HCS-based decoders group hypotheses by their history, each group representing all state hypotheses sharing the same LM history. This way, word histories only need to be considered when handling word labels, and thus can be ignored during dynamic programming at the intra-word state level [NO00]. HCS makes large decoding sub-networks able to be safely removed during search, thereby lowering memory requirements. The HCS approach allows us to deal with any length of LM histories, potentially even infinite, but other challenges remain to build a real-time decoder. The most important of them is the computational cost required by the neural model, in particular the Softmax layer, that can be excessively expensive for large vocabularies in order to even approach real-time capabilities. In the following, we describe the solutions that we implemented in order to address these challenges.

4.2.1 Static look-ahead tables

LM look-ahead is a well-established pruning technique consisting of adjusting the LM score for each hypothesis h and time t by taking into account every possible word w to follow [Nol17]. Strictly, this technique implies computing an extra set of look-ahead scores $p(w|h)$ for each history h and word w. The cost of this computation, already high when using n-gram LMs, becomes prohibitive when using neural-based LMs.

To ensure the look-ahead computation is done within strict time constraints, a common technique is to build pre-computed static look-ahead tables from simplified n-gram LMs. That is, look-ahead tables are built from m-grams with $m < n$ from a "big" n-gram LM so that all of them fit concurrently in memory. The pre-computed look-ahead table is used for $p(w|h)$ except for when a word-end node is reached, situation where the look-ahead score is replaced by the big LM probability. This procedure allows for both the look-ahead tables and different queries to the big LM to be greatly reduced, bringing huge improvements to the speed of the decoder.

4.2.2 Variance regularization and lazy evaluation

As previously mentioned, one of the main drawbacks of using neural-based LMs for large vocabulary ASR decoders is the computational cost of the Softmax layer, primarily derived from the calculation of the normalization term. To address this issue, we follow [Shi+14] and include a Variance Regularization term to the training optimization function that allows for a drastic decrease in required compute. This technique makes the softmax normalization term approximately constant so that the probability of a word y given a history h can be approximated as

$$p(y_t|y_0^{t-1}) = \frac{\exp(\text{nn}(y_0^{t-1}) \cdot a_y)}{\sum_i \exp(\text{nn}(y_0^{t-1}) \cdot a_i)} \approx \frac{\exp(\text{nn}(y_0^{t-1}) \cdot a_y)}{D} \tag{4.1}$$

where $\text{nn}(y_0^{t-1})$ is the final layer of the neural network given the input h, a_i is the weight for the output i, and D is the constant being approximated as the normalization term. This technique involves introducing a special term into the optimization function involving the variance of the softmax normalization term, so that the neural network "wants to" minimize that variance during training. At the end of the training, D is taken as the mean of the normal-

ization term in the training data, from which the actual normalization term is expected to deviate only minimally.

To speed up the decoding, the Variance Regularization technique was used in conjunction with a lazy evaluation strategy delaying the evaluation of the neural LM as much as possible. Each neural history h is internally represented as the tuple (y, h', V) where y is the last word of h, $h' = (y', h'', V')$ the previous history state, and V is either $\varnothing$ (empty) or the hidden state for h. During decoding, each time a word-end node y is reached, the following steps are executed:

1. Look up V'. If not found, compute the RNN for it from h'.
2. Compute the estimated $p(y|h) \approx \frac{\exp(\mathrm{nn}(h) \cdot a_y)}{D}$
3. Create the new state $h = (y, h', \varnothing)$

By this approach, new histories are created at a negligible cost, since the forward step in the neural model is only carried out when a word-end node is reached for the first time. Once the state for the neural model is obtained, it is cached. Furthermore, in practice most histories will be pruned before any hypothesis reaches a word-end node, saving significant amount of computation. This saving is compounded by the fact that the Softmax computation is approximated by a constant term as the normalization factor, as previously mentioned.

4.2.3 Novel pruning techniques

Two new pruning techniques were implemented specifically for the one-pass decoder, in addition to the conventional pruning methods. These techniques apply to the search process, and not at all to the neural models themselves. A problem we encountered was that in some situations, the lack of LM history recombination together with the histogram pruning (a maximum number of active hypotheses at each time frame) produced a decrease in transcription quality. This was apparent for some long sentences where most of their active hypotheses were identical except for long term differences in their LM history, effectively making the beam decoder behave as a greedy decoder. To avoid this behaviour, a LM history recombination parameter (LMHR) was introduced, where two different histories are recombined if their last n words are the same. This kind of recombination forces the decoder to consolidate prefixes and thus to focus its exploration of possible decisions on the present time frame. A similar technique was introduced in [HZD14], although with a different moti-

vation. Another pruning scheme that was introduced is LM histogram pruning (LMHR), which sets the maximum number of new LM histories that can be created at each time frame.

We explored these pruning parameters in the two tasks that will be introduced in Section 4.4, namely LibriSpeech and TED-LIUM version 2. These experiments related to pruning parameters were made in collaboration with Javier Jorge and not the main focus of this thesis. They have been included in this section for a better understanding of the new pruning parameters in our custom one-pass decoder for hybrid ASR systems. The main experimentation and the focus of the work described in this thesis is located on Section 4.4.

The LMHR parameter is evaluated in Figure 4.1, where a Transformer language model (as will be described in Section 4.3) was used with different history window sizes to gauge the collusion between the limitation on the number of words coming from the LMHR parameter and coming from the LM window in terms of WER. We observe, in each task, an optimal value for the LMHR parameter irrespective of the LM window limitation: 12 for LibriSpeech and 9 for TED-LIUM. For LMHR values below the optimal, the differences in LM histories between the hypotheses are not allowed the weight they should have. For values above the optimal, the differences are often too far in the past and newer differences should have more weight in the present decision.

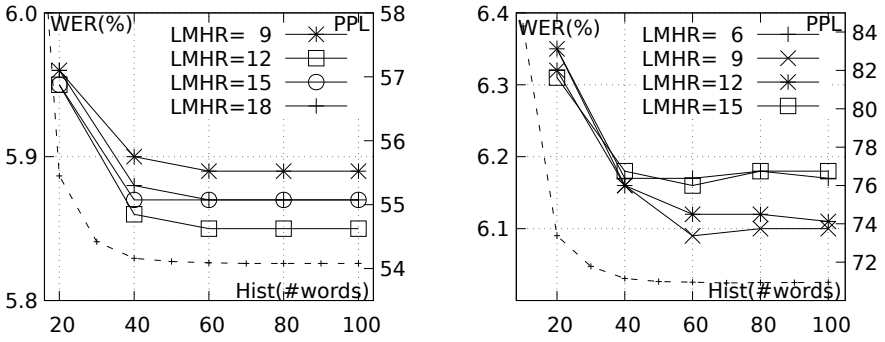

Figure 4.1: WER and PPL as a function of TLM history size for different LMHR values. Solid curves represent WER, measured on the left axis, while the dashed curve represents the PPL, measured on the right axis. LibriSpeech (left) and TED-LIUM (right).

The LMHR parameter is evaluated in Figure 4.2, with the same tasks and models as the LMHR parameter. As it is relevant for computational time, in this case the WER was measured against latency, varying other search param-

eters. We observe that not for all latencies the optimal LMHP stays the same, and that it is highly dependent on the task. The "LMHP=Inf" line represents no LM histogram pruning at all. It is not surprising that it entails the highest latencies and, as such, is often above the other lines in the figure.

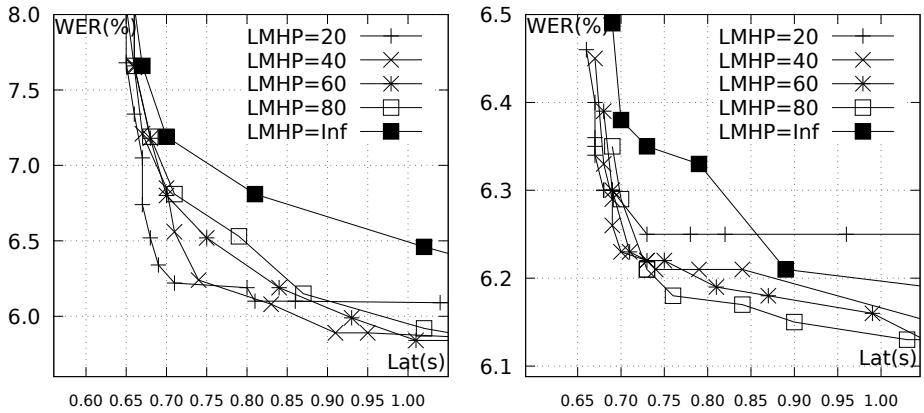

Figure 4.2: WER against latency (in seconds) for different LMHP values, for LibriSpeech (left) and TED-LIUM (right).

4.2.4 Streaming adaptation

The decoder structure described above has to be adapted to bridge the gap between real-time and actual streaming conditions. Under these conditions, the complete acoustic context is not available for a given acoustic frame. Therefore, we need to define a future context window to calculate the scores for each acoustic frame. This window directly implies a delay between the input and the output, and is also referred as acoustic lookahead.

Regarding this acoustic lookahead, we followed a similar strategy to [ZSN16], based on a sliding window of a given size over the acoustic sequence. The frames in the window are used to compute the forward and backward steps of a BLSTM over the sequence, obtaining a score for each frame within the window as in [Moh+15]. We move the sliding window frame by frame. Regarding the overlapping frames, to obtain the final score that will be provided to the decoder we use a uniform weighted average of the acoustic score of all windows for a given frame. In the extreme cases where the utterance is shorter than the window, zero padding is introduced up to the length of the window beyond the sequence.

Limiting the future context forbids us from performing full sequence normalization of acoustic features. To address this, we apply a weighted moving average technique, updating normalization statistics on-the-fly. An initial delay is introduced to gather statistics to initialize the mean and variance. A parameter, n_{norm}, controls the number of seconds that are used to compute the initial statistics. The first n_{norm} frames are normalized with this mean and variance. Afterwards, the mean and variance are updated frame by frame without incurring in any additional delay. This scheme causes an initial delay for the first few words from the stream out of the decoder. However, in a real streaming setup, this initial delay is small and the system will catch up quickly without any further impact to global latency or system performance. The technique is applied over a batch B_j of frames as

$$\hat{B}_j = B_j - \hat{\mu}_j \tag{4.2}$$

$$\hat{\mu}_j = \frac{f_{j-1} + \sum_{t=1}^{b+w} B_{j,t}}{n_{j-1} + b + w} \tag{4.3}$$

Where f_{j-1} represents the accumulated sum of values from previous frames up to batch B_{j-1}, $B_{j,t}$ the t-th frame in batch B_j, n_{j-1} stands for the number of frames until batch B_{j-1}, and b and w are the batch and window sizes, respectively. Our implementation sets f_j and n_j to be exponentially decaying accumulators, updated by weighting the contribution of previous batches with an adjustable parameter $\alpha \leq 1$:

$$f_j = \alpha \cdot f_{j-1} + \sum_{t=1}^{b} B_{j,t} \tag{4.4}$$

$$n_j = \alpha \cdot n_{j-1} + b \tag{4.5}$$

4.3 Streaming Transformer language models

Two key ideas are in play to adapt the Transformer LM (TLM) to the streaming setup. A first key idea is, like seen above, to add a Variance Regularization term during training as a way to fit the normalization term of the Softmax so that it deviates minimally from a constant value. This is critical to reduce the high computational cost of the linear projection to vocabulary size before the

Softmax activation during inference time, as we are enabled to approximate the normalization term by a constant.

A second key idea, concerning particularly TLMs and not other neural LMs such as LSTMs or other RNN LMs, is to limit the word history size. The strength of self-attentive LMs like the Transformer model resides in their ability to attend to all previous words and better condition their output based on their history. This makes them more versatile than RNNs, as they are not forced to compress all the history into a single vector, causing some information loss. In exchange for that, this fact implies that the whole history must be processed every time a next word is predicted. Although some workarounds exist that can expedite this process, it is still troublesome for the streaming setup: if we allow the history to grow without limit, both memory requirements and computational time will increase boundlessly as well. To circumvent this issue, we limit the Transformer history size to the n previous words and we do not store any internal state, alleviating the memory requirements. However, in exchange, we are forced to compute the output again with each call. As we will see during the empirical study, with the history sizes that we worked with, this was not an issue.

4.4 Experiments

4.4.1 Experimental setup

We evaluated the performance of our streaming decoder and models on the LibriSpeech ASR corpus [Pan+15] and the TED-LIUM release 2 corpus [RDE14]. Acoustic models were trained on the 961 hours for LibriSpeech and the 207 hours provided in TED-LIUM v2. As development and test data, we used the *-*other* for LibriSpeech, with 5.3 hours for dev and 5.1 hours for test, and the *-*legacy* for TED-LIUM, with 1.6 and 2.6 for development and test, respectively. Regarding the text data, we used the ∼800M words text provided for LibriSpeech, whereas for TED-LIUM the text data came from the six provided subsets extended with the TED-LIUM training audio transcriptions with up to 230M running words in total. System vocabularies were limited to 200K and 153K words for LibriSpeech and TED-LIUM, respectively. Out-of-vocabulary (OOV) ratios were less than 1% in both tasks.

The acoustic models were trained as follows. First, we trained context-dependent feed-forward DNN-HMMs with three left-to-right states, using the transLectures-UPV toolkit [Del+14]. The state-tying scheme follows a phonetic decision tree

approach [YOW94] resulting in 8.3K and 10.8K tied states for LibriSpeech and TED-LIUM, respectively. Gaussian mixture models were used to initialize alignments, from where feed-forward models bootstrapped the training of BLSTM-HMMs [Zey+17] as the acoustic model, trained with TLK and TensorFlow [Aba+15]. These BLSTMs were composed of 8 layers with 512 cells per layer and direction, trained with the cross-entropy criterion. Back-propagation through time was limited to a window size of 50 frames.

As to language modelling, we trained TLMs using fairseq [Ott+19] with a custom implementation of the Variance Regularization criterion. We followed the 'base' configuration of the Transformer, with 512 cells per layer, a feed-forward of 2048 units, and 8 attention heads. Only in TED-LIUM, the intersection between the provided vocabulary and the vocabulary of the training set resulted in a smaller output layer of 144K units. We ran our experiments with three different model sizes per task, with different number of Transformer layers: 12, 18, and 24.

Additionally, we explored LM combination by interpolating Transformer models with n-gram and/or LSTM LMs. Regarding n-gram LMs, we used the 4-gram ARPA LM provided with the LibriSpeech dataset (*fglarge*), while for TED-LIUM we explicitly trained a standard Kneser-Ney smoothed 4-gram LM with the same data as [RDE14] using SRILM [Sto+11]. A pruned version of these models was used to estimate the static look-ahead tables. LSTM LMs were trained with two specific criteria added to the objective function: Noise Contrastive Estimation [MT12] and Variance Regularization [Hor+07], and using the CUED-RNNLM toolkit [Che+16]. LSTM LMs consisted of a 256-unit embedding layer and two LSTM layers of 2048 units. Output Softmax layers have the same number of units as the respective TLMs for each task.

To assess and compare the different LM performances, we provide perplexities (PPL) computed over the development sets of each task. To evaluate overall ASR system quality, we computed Word Error Rate (WER) figures on the corresponding development and/or test sets. Additionally, we measured times for ASR systems both for the off-line and streaming setups. For the off-line case, we provide Real Time Factor (RTF) values. RTF is defined as the ratio between the computation time to transcribe a set of audios, and the duration of that set. For the streaming case, we provide mean system latencies. We define latency as the time elapsed between the system receiving the last input frame of an uttered token (in our case, word) and the point in time when the first hypothesis for that word is delivered. All time-measuring experiments were conducted on a machine with an Intel Xeon(R) CPU E5-1620@3.50GHz and a GPU GTX2080Ti with 12GB of RAM. The estimation of the scores for

the neural models (the acoustic BLSTM, the LSTM LM, and the TLM) was performed on the GPU, while the n-gram scoring and the rest of the decoding took place on the CPU.

The experimentation is structured as follows. First, Section 4.4.2 explores the performance of the different LMs considered in this work and studies the effect of limiting the history window of the TLMs. Then, Section 4.4.3 analyses ASR performance considering WER and RTF for different number of Transformer layers and search parameters. Next, Section 4.4.4 gauges the performance of systems interpolating different LMs including the best performing Transformers with other LMs. Finally, Section 4.4.5 compares the performance of the ASR systems, using the best found LM combination in both off-line and streaming conditions, and provides latencies for the latter.

4.4.2 Language model evaluation

Table 4.1 shows the PPLs on the development sets for the three types of LM model considered in this chapter: n-gram, LSTM, and Transformer models of 12, 18, and 24 layers. As expected, TLMs attained significantly better PPLs than the other LMs being considered. Furthermore, slight improvements of PPL were provided by increasing the number of Transformer layers. The question we are to explore is whether a streaming ASR system can exploit these better PPL figures and translate them to terms of WER and latency.

Table 4.1: LM perplexities for LibriSpeech and TED-LIUM

Model	LibriSpeech	TED-LIUM
n-gram	140.9	117.5
LSTM	72.5	86.7
Transformer 12L	60.7	74.3
Transformer 18L	58.1	72.4
Transformer 24L	56.2	71.0

Figure 4.3 shows the impact of limiting the history window in the evaluation of TLMs, in terms of PPL as a function of th history size in words. The numbers are computed over the developments sets of LibriSpeech (left plot) and TED-LIUM (right plot) for Transformer models of 12, 18, and 24 layers. The discontinuous line indicates the percentage of fully-seen sentences and is measured against the right vertical axis.

From this figure we realize that PPLs consistently improve in all cases as we increase the history size, until we reach approximately size 40. For history

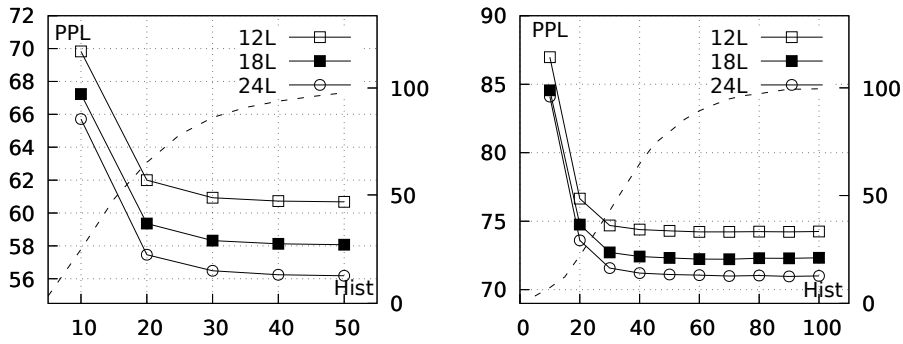

Figure 4.3: Transformer LM PPLs (left vertical axis) for different number of layers (in legend), and percentage (right vertical axis) of fully-seen sentences (discontinuous line), as a function of history size, for LibriSpeech (left) and TED-LIUM (right).

sizes over 40, further PPL improvements are negligible. In LibriSpeech, this behaviour is to be expected considering that 94% of sentences are shorter than 40 words. This result is somewhat more interesting in TED-LIUM, where a significant number of sentences (35%) are longer than 40 words. This latter case is, to some extent, more representative of a real streaming ASR scenario, for example a lecture of more than half an hour, where most of the history would fall out of any realistic history window size. For this reason, we chose a history window size of 40 words for the rest of the experiments described in this chapter.

4.4.3 ASR systems with Transformer language model

In this section, we analyse the performance of hybrid ASR systems with TLMs of varying number of layers under an off-line setup. The goal is to find a good trade-off between transcription quality, measured in WER, and speed, measured in RTF. As our aim is to bring these systems to a streaming setup, our search for this trade-off is strictly limited to those with RTF$<$1, to ensure they are able to process input audio streams in real time. System speed is adjusted by altering search parameters, particularly the prune.

Figure 4.4 shows WER (vertical axis) and RTF (horizontal axis) curves for ASR systems with TLMs of 12, 18, and 24 layers, computed over the development sets of LibriSpeech and TED-LIUM. As expected, the lowest RTF values are obtained with 12 layers. But taking into account also the WER, the best option for a good WER-RTF balance turns out to be different for each particular

task. For LibriSpeech, the TLM with 24 layers provides the best quality (5.6% WER) while complying with RTF constraints (∼0.9 RTF). On the other hand, for TED-LIUM, the LM with 12 layers is the fastest and also provides the best quality (5.8% WER, ∼0.8 RTF).

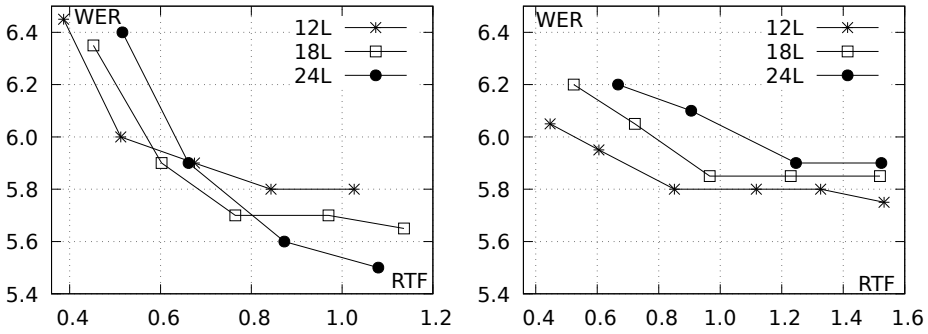

Figure 4.4: WER vs RTF of ASR systems with different Transformer LMs, for LibriSpeech (left) and TED-LIUM (right).

We interpret these results in the light that WER is particularly sensible to PPL changes in th LibriSpeech task. In other tasks such as TED-LIUM, this correlation is weaker and, as such, more exploration of the search space is always preferred with a faster LM, even when it provides an inferior PPL. In other words, computation time is better spent exploring more prefixes, instead of computing a more accurate LM score for each candidate prefix. Is is also important to remark that LibriSpeech provides four times more text data than TED-LIUM, and this may have an impact of how great the improvements of increasing the model size are. To us, all of the above explains the striking differing behaviour of the two tasks in Figure 4.4. For the remaining experiments, we selected the best-performing TLMs complying with an RTF$<$1: the 24-layer TLM for LibriSpeech and the 12-layer TLM for TED-LIUM.

4.4.4 ASR systems with language model combination

Our aim is to further increase overall ASR performance further than in the previous section by LM combination. Studying the effects of combining the best-performing TLMs with n-grams and/or LSTM LMs on our streaming decoder, we may arrive at a combination that enhances the transcription accuracy while remaining under RTF$<$1. For simplicity, and taking into account PPL results described in Table 4.1, we only studied LM combinations involv-

ing Transformer LMs. LM combinations are done by a linear interpolation, where the corresponding weights are the result of a minimization of PPL on the development set for each task. First, the impact of these interpolations is analysed in terms of PPL, and afterwards in terms of how the PPL gains are translated into ASR performance, in terms of both transcription quality and speed.

Table 4.2 shows interpolation weights (W%) and PPLs computed for each possible combination involving Transformer models over the development set in both tasks. Only the weights of the n-gram and LSTM LM models are shown, the Transformer LM receiving the remaining weight up to 100%.

Table 4.2: Interpolation weights (W%) and PPLs computed over the development sets of each task, for all LM combinations. Interpolation weights are shown only for models other than Transformer, which receives the remaining weight

	LibriSpeech		TED-LIUM	
Model	W(%)	PPL	W(%)	PPL
Transformer	–	56.2	–	74.4
+ n-gram	4	54.9	27	63.2
+ LSTM	14	54.6	39	68.3
+ n-gram + LSTM	2+13	54.4	27+19	61.0

We observe that LM combination has a positive effect on PPL, but the magnitude of its impact is diverse. On LibriSpeech, the improvement is very slight with only a 3% reduction on PPL when the three LMs are involved. Meanwhile, on TED-LIUM, the PPL reduction of up to 18% is quite significant. Interpolation weights reflect this phenomenon, and correspondingly, n-grams and LSTM models take lower weights on LibriSpeech (up to 15%) and higher weights on TED-LIUM (up to 46%).

Next, we study how these improvements in PPL may translate to ASR system performance. For the goal of building the best possible streaming ASR system, we must again explore different prune parameters, as LM combination increases the computational load of computing the LM scores during the decoding process, to ensure the RTF$<$1 requirement while keeping a good balance with transcription quality. Figure 4.5 shows WER, measured in the vertical axis, against RTF, measured in the horizontal axis, for all LM combinations present in Table 4.2, computed over the development sets of LibriSpeech and TED-LIUM.

For LibriSpeech, the interpolation has little effect on the WER, and most of the difference is made by varying the search parameters. This result is in line

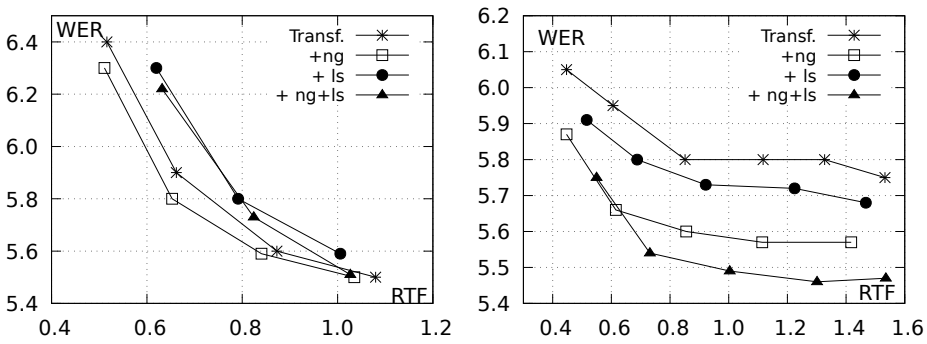

Figure 4.5: WER vs RTF of ASR systems with different LM combinations, for LibriSpeech (left) and TED-LIUM (right).

with the PPLs in Table 4.2, where the interpolation returned limited improvements. The Transformer + n-gram combination provided slight but consistent improvements for the quality-speed balance, as we can observe by the curve being always below the others. For TED-LIUM, the greater PPL improvements due to LM combination translated into WER figures. The Transformer + n-gram + LSTM LM combination offered the best quality at acceptable RTF$<$1 rates.

At this point, as final ASR systems, we selected those using the LM combination offering the lowest WER with an RTF$<$1, which are: Transformer + n-gram for LibriSpeech, and Transformer + n-gram + LSTM for TED-LIUM.

4.4.5 Streaming ASR systems

Finally, the two selected systems, one for each task, were tested under a true streaming setup. First, we measure WERs over the development and test sets, and compare them with the performance under an off-line setup, and observe the corresponding WER degradation. Then, we compare these results with other works up to that date. Finally, we provide measured system latencies. The streaming configuration providing the best result in [Jor+20] was used during these experiments. Particularly, the overlapping acoustic window was of size 50 and stride 1.

Tables 4.3 and 4.4 show WER figures of our final ASR systems for LibriSpeech and TED-LIUM, respectively, evaluated on off-line (restricted to RTF$<$1) and streaming conditions, and comparative results with other related works. First,

we can observe a small WER degradation when we move from off-line to a streaming setup. This is caused only due to our inability to compute acoustic statistics from the whole input stream for mean and variance normalization of acoustic features. Second, our streaming systems obtained competitive results, even when compared to other off-line works; and top, state-of-the-art results when compared to other streaming works evaluated under similar conditions.

Table 4.3: LibriSpeech results summary

System	dev	test
Off-line (RTF<1)	5.6	5.9
Streaming	5.9	6.4
Lüscher et al. [Lüs+19] (Off-line)	4.5	5.0
Moritz et al. [MHL20] (Off-line)	6.0	6.1
Moritz et al. [MHL20] (Streaming)	7.2	7.3
Zhang et al. [Zha+20] (Off-line)	–	5.6
Zhang et al. [Zha+20] (Streaming)	–	10.0

Table 4.4: TED-LIUM results summary

System	dev	test
Off-line (RTF<1)	5.5	6.2
Streaming	5.7	6.4
Zhou et al. [Zho+20] (Off-line)	5.1	5.6

As for latencies, our theoretical latency (0.6s) is dominated by the look-ahead window of 0.5 seconds, plus 0.1 seconds due to batch processing. Actual measured latencies are slightly higher: 0.9±0.4s in LibriSpeech, and 0.8±0.3s in TED-LIUM. We can compare this to theoretical latencies of other works, like the 2.2 seconds of [MHL20], or the 1.1 seconds of [Zha+20]. As a reference, the UK Office of Communications recommends, to TV broadcasters, a maximum latency of 3 seconds in live subtitling [Uni14].

4.5 Conclusions

Following our previous work on real-time one-pass decoding with hybrid ASR systems and LSTM language models, in the work described by this chapter we reported further improvements by replacing LSTM LMs with Transformer LMs. Two key ideas have been implemented to adapt TLMs to our streaming one-pass decoder: the incorporation of a Variance Regularization term to the optimization criterion during training, and the limitation of the word history

size during inference time. Empirical results showed that the incorporation of TLMs into our decoder lead to top recognition rates on both tasks, LibriSpeech and TED-LIUM release 2, under the streaming setup.

Chapter 5

Streaming Automatic Speech Recognition with Transformer Language Models for the Albayzin-RTVE contest

5.1 Introduction

The third goal of the thesis is to apply the reached streaming ASR technical capabilities to a relevant use case. This chapter revolves around a use case for the technology developed in Chapter 4 applied to TV broadcasting. Specifically, a research challenge issued by the main public TV broadcaster of Spain, the Radio Televisión Española (RTVE), namely the Albayzín-RTVE 2020 Speech-to-Text (S2T) challenge. Live audio and video streams such as TV broadcasts, conferences, lectures, and general-public streaming services over the Internet are becoming prevalent in recent years. The COVID-19 pandemic has only accelerated this trend, with video meeting platforms having experienced a rapid growth in use. With this phenomenon, automatic transcription and translation of such audio streams is a key feature for many of these services, for accessibility reasons, in order to reach wider audiences, or to ensure a proper understanding among non-native speakers.

There is a growing number of countries that require, even by law, that TV broadcasters provide accessibility options to people with hearing disabilities, with the minimum amount of content to be captioned increasing yearly [07; 08]. Some TV broadcasters and other live streaming services employ human

transcribers, manually transcribing from scratch the content of the media, as an initial solution to comply with both the legislation and the user expectations. However, a manual solution is a hard and difficult to scale task. Furthermore, is is usual for these organizations to not have many human resources available to devote to this particular task. Consequently, the relevance of the challenge of obtaining high-quality real-time streaming ASR systems for live TV captioning is undoubtable.

Our participation in the Albayzín-RTVE 2020 S2T Challenge consisted in the submission of a primary, performance-focused streaming ASR systems, plus three extra contrastive systems: two of them are latency-focused streaming systems, and the other is a more conventional off-line ASR system with a Voice Activity Detection (VAD) module. These systems followed the streaming ASR one-pass hybrid decoder strategy described in Chapter 4 and were built with our in-house transLectures-UPV ASR toolkit (TLK) [Del+14]. In contrast, most other participants in the contest used the widely-available Kaldi toolkit for DNN-HMM ASR systems [Álv+21]

This chapter is organized as follows. Section 5.2 contains a description of the challenge including an overview of the databases intrinsic to it. Section 5.3 details the ASR systems that we submitted to the challenge, how they were trained, and the relevant experimentation. Section 5.4 describes similar experiments but with the same data restriction as in the 2018 closed challenge, for a more comprehensive comparison of the state of the technology and its advance during these two years. Finally, Section 5.5 contains concluding remarks for the chapter.

5.2 Challenge Description and Databases

The Albayzín-RTVE 2020 S2T Challenge consists of automatically transcribing different types of TV shows from the main Spanish public TV station RTVE, and the assessment of ASR system performance in terms of Word Error Rate (WER) by comparing those automatic transcriptions with human-generated reference transcriptions [Lle+20a].

The MLLP-VRAIN group previously participated in the 2018 edition of the challenge in collaboration with the Human Language Tecnology and Pattern Recognition (HLTPR) of RWTH Aachen University [Jor+18]. The evaluation was carried out on the RTVE2018 database [Lle+18], which includes 575 hours of audio from 15 different TV shows broadcasted between 2015 and 2018. This database is partitioned into four sets: *train*, *dev1*, *dev2*, and *test* (*test-2018*).

In the 2018 challenge, our systems won both the open-condition and the closed-condition tracks [Lle+19], scoring 16.5% and 22.0% WER points respectively in the *test-2018* set.

For the 2020 editions of the challenge, a single open-condition track was proposed. Submitted systems were evaluated over the test set from the RTVE2020database (*test-2020*), including 78.4 speech hours at a sampling rate of 16 kHz from 15 different TV shows broadcasted between 2018 and 2019 [Lle+20b].

5.3 MLLP-VRAIN Systems

This section describes the (hybrid) ASR systems that we submitted to the Albayzín-RTVE 2020 S2T Challenge.

5.3.1 Acoustic Modelling

The AMs were trained using 205 filtered speech hours. They came from the RTVE2018 *train* set (totalling 187 hours), but we also split the *dev1* set into a *dev1-train* and a *dev1-dev* set, as we used part of the official dev set as internal training data. The *dev1-train* set contains 18 hours of transcribed speech, and this is the exact same partition as in our previous participation in 2018 [Jor+18]. The acoustic data was extended with about 3.7K hours of audio resources crawled from the Internet in 2016 and earlier. Table 5.1 summarises all training datasets along with their total duration in hours.

Table 5.1: Spanish transcribed speech resources for AM training

Resource	Duration
RTVE2018: *train*	187
RTVE2018: *dev1-train*	18
Internal: entertainment	2932
Internal: educational	406
Internal: user-generated content	202
Internal: parliamentary data	158
Voxforge [Vox]	21
TOTAL	3924

From the acoustic data, we extracted 16-dimensional MFCC features plus first and second derivatives, amounting to 48-dimensional acoustic feature vec-

tors, every 10ms. We trained a context-dependent feed-forward DNN-HMM with three left-to-right tied states for each triphoneme using the TLK toolkit [Del+14]. The state-tying scheme followed a phonetic decision tree approach [YOW94] that produced 10K tied states. The feed-forward models were used to generate an alignment from which a BLSTM-HMM AM was trained with 85-dimensional filterbank features, following [Zey+17] and using both TLK and TensorFlow [Aba+15]. The BLSTM had 8 bidirectional hidden layers with 512 LSTM celss per layer and direction. As in [Zey+17], we performed chunking during training, performing back-propagation through time up to a window size of 50 frames. SpecAugmentation was applied by means of time and frequency distortions [Par+19].

5.3.2 Language Modelling

As to language modelling, we trained both n-gram and neural-based (LSTM and Transformer) LMs to perform one-pass decoding with linear combinations of them as described in Chapter 4, using the text data sources and corpora described in Table 5.2.

Table 5.2: Statistics of Spanish text resources for LM training. S=Sentences, RW=Running words, V=Vocabulary. Units are in thousands(K)

Corpus	S (K)	RW (K)	V(K)
OpenSubtitles [Ope]	212,635	1,146,861	1,576
UFAL [Cor]	92,873	910,728	2,179
Wikipedia [Wik]	32,686	586,068	3,373
UN (WSMT corpus) [WSM]	11,196	343,594	381
NewsCrawl [WMT]	7,532	198,545	648
Internal: enterntainment	4,799	59,235	307
eldiario.es [Eld]	1,665	47,542	247
El Periódico [Per]	2,677	46,637	291
Common Crawl [201]	1,719	41,792	486
Internal: parliamentary data	1,361	35,170	126
News Commentary [WMT]	207	5,448	83
Internal: educational	87	1,526	35
TOTAL	369,434	3,423,146	5,785
Google-counts v2 [Lin+12]	-	97,447,282	3,693

For the n-grams, we trained 4-gram LMs using SRILM [Sto+11] with all text resources plus the Google-counts v2 corpus [Lin+12], accounting for 102G run-

ning words. The vocabulary size was limited to 254K words, with an OOV ratio of 0.6% computed over our internal development set.

For the neural LMs we considered the LSTM and Transformer architectures. In both cases, LMs were trained using a 1-gigaword subset extracted randomly from all available text resources, except Google-counts which were not strictly text but counts of co-occurrences. The vocabulary for the neural LMs was defined as the intersection between the n-gram vocabulary (254K) and that derived from the training subset. This intersection was taken to avoid zero probabilities for words that are present in the system vocabulary but not in the training subset. The n-gram words that remain out of this intersection are taken into account when computing perplexities by renormalizing the unknown-word score accordingly.

Specific training details for each neural architecture are as follows. On the one hand, LSTM LMs were trained using the CUED-RNNLM toolkit [Che+16]. The Noise Constrastive Estimation (NCE) criterion [MT12] was used to speed up model training, and the normalization constant learned from training was used during decoding [Che+15]. A model was selected based on the lowest perplexity over our internal development set, with 256-unit embedding and two hidden LSTM layers of 2048 units. On the other hand, Transformer LMs (TLM) were trained using a customized version of the Fairseq toolkit [Ott+19], selecting on the basis of lowest perplexity with a 24-layer Transformer with 768 units per layer, 4096-unit feed-forward, 12 attention heads, and an embedding of 768 dimensions. The Transformer network was trained with a Variance Regularization criterion, applying the learned constant regularization constant during the training to speed up the computation of TLM scores as in [Shi+14] and as described in Chapter 4. Both models were trained until convergence with batches limited to 512 tokens, with parameter updates every 32 batches.

5.3.3 Decoding Strategy

As described in Chapter 4, our hybrid ASR systems follow a real-time one-pass decoding by means of a History Conditioned Search (HCS) strategy [Nol17]. HCS allows us to leverage additional LMs during decoding while satisfying real-time constraints. Our decoding strategy introduces two parameters to control the trade-off between Real Time Factor (RTF) and WER: LM history recombination and LM histogram pruning. A static look-ahead table is needed by the decoder to use pre-computed look-ahead LM scores, and is generated from a pruned version of the n-gram LM. Section 4.2 expands more details about the decoding strategy that was applied.

Specifically for Transformer LMs, an approximation for the Softmax computation is applied to ensure the computation is performed in time; and the history is limited to a maximum number of words, as they otherwise have the inherent capacity of attending to potentially infinite word sequences. These adaptations are needed to meet the strict computational time and memory constraints imposed by the streaming requirement. More details about these adaptations are found in Section 4.3. By applying these modifications, our decoder acquires the capacity to deliver live transcriptions for incoming audio streams of indefinite length, with latencies lower-bounded by the context window size.

5.3.4 Experiments and Results

For the system development evaluations, we used the dev and test sets from the RTVE2018 database. We devoted our internally split *dev1-dev* set [Jor+18] for development purposes, whilst the *dev2* and *test-2018* sets were used to test ASR performance. Finally, *test-2020* was the blind test used by the organization to rank the participant systems. Table 5.3 contains essential statistics of these sets.

Table 5.3: Statistics of RTVE development and test sets, including our internally split *dev1-dev* set. Contains: total duration (in hours), number of files, average duration of samples in seconds (d_μ) $\pm$ standard deviation (σ), and running words (RW) in thousands (K)

Set	Duration (h)	Files	d_μ	$\pm$	σ	RW (K)
dev1-dev	11.9	10	4267	$\pm$	1549	120
dev2	15.2	12	4564	$\pm$	1557	149
test-2018	39.3	59	2395	$\pm$	1673	377
test-2020	78.4	87	2314	$\pm$	1576	519

First, we studied the perplexity (PPL) on the *dev1-dev* set of all possible linear combinations for the three types of LMs considered in this work: n-gram, LSTM, and Transformer. Table 5.4 provides the PPLs obtained by each of these interpolations, along with the optimum LM weights that minimized PPL in the *dev1-dev* set. The presence of the Transformer LM in the interpolation provides significantly lower perplexities in all cases and, consistent with this fact, it takes on a high portion of the interpolated weight when combined with other LMs. The TLM in isolation already attains a strong perplexity baseline of 63.3, and the best possible combination represents only a 6% relative, with all three LMs involved.

Second, we tuned decoding parameters to provide a good WER-RTF tradeoff on *dev1-dev*, with the hard constraint of RTF$<$1 to ensure the hard prereq-

Table 5.4: Perplexity (PPL) and interpolation weights, computed over the *dev1-dev* set, of all possible combinations of n-gram (ng), LSTM (ls) and Transformer (tf) LMs

LM Combination	PPL	Weights (%)
ng	179.5	–
ls	98.4	–
tf	63.3	–
ng + ls	93.2	15 + 85
ng + tf	61.1	6 + 94
ls + tf	60.7	13 + 87
ng + ls + tf	59.5	5 + 10 + 85

uisite of real-time processing of the input. From these hyperparameters, we highlight, due to their relevance, a LM history recombination of 12, LM histogram pruning of 20, and TLM history limited to 40 words.

At this point, we defined our participant off-line hybrid ASR system identified as *c3-offline* (contrastive system no. 3), consisting of a fast pre-recognition + Voice Activity Detection (VAD) step to detect speech/non-speech segments as in [Jor+18], followed by real-time one-pass decoding with our BLSTM-HMM AM, using a Full Sequence Normalization scheme and a linear combination of the three types of LMs: n-gram, LSTM, and Transformer. This system scored 12.3 and 17.1 WER points on *test-2018* and *test-2020*, respectively.

Next, as our focus was to develop the best-performing streaming-capable hybrid ASR system possible for this competition, we explored streaming-related decoding parameters to optimize WER on *dev1-dev*, using the BLSTM-HMM AM and the strongest LM consisting of a linear combination of all three LMs. From this exploration, a context window size of 1.5s and $\alpha = 0.95$ was chosen for the acoustic feature normalization technique as described in Section 4.2.4. This configuration was chosen for our primary submitted system, labelled *p-streaming_ 1500ms_ nlt*, achieving WER rates of 11.6 and 16.0 on *test-2018* and *test-2020*, respectively. This system, in contrast to the above described *c3-offline*, does not integrate any VAD module. Instead, this task is left for the decoder to carry out via the implicit non-speech model of the BLSTM-HMM AM.

The removal of the LSTM LM from the linear interpolation led to the contrastive system no. 1, identified as *c1-streaming_ 1500ms_ nt*. The motivation for this action was that the computation of LSTM LM scores is computationally expensive, while contributing only a 3% relative improvement to PPL

scores. This enhanced the stability of the system latency with almost no degradation to output quality in terms of WER: 11.6 and 16.1 points on *test-2018* and *test-2020*, respectively.

Both of the above described streaming systems, *p-streaming_1500ms_nlt* and *c1-streaming_1500ms_nt*, share the same theoretical latency of 1.5s due to the context window size. This parameter can be adjusted at decoding time, which can result in some mismatch between training and inference depending on how the system was trained. In our past experience with BLSTM-HMM AMs, this mismatch doesn't affect the final performance of the system. Adjusting the context window size allows us to tune the decoder balance for lower latencies or better transcription quality. We set for ourselves a final goal for the challenge in finding a system configuration providing state-of-the-art, stable latencies with minimal WER degradation.

Figure 5.1 illustrates the performance in WER on our development set as a function of the context window size, limited to one second at maximum. Keep in mind that in the framework that we use, the context window size is equivalent to the acoustic lookahead. As we focused on analysing AM performance, we used the n-gram LM for this study for efficiency reasons. In light of the results, we chose a window size of 0.6s for our short-latency system, as it brings a good balance between transcription quality and theoretical latency.

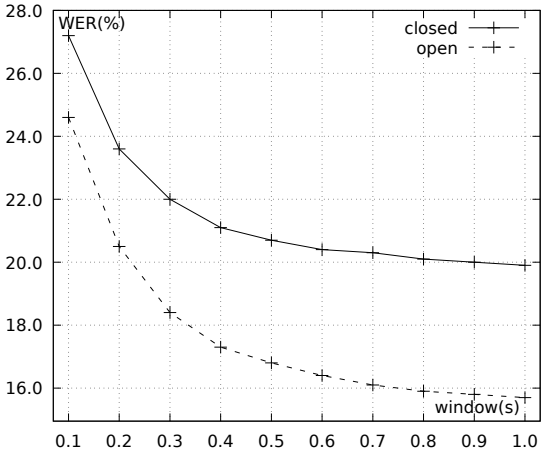

Figure 5.1: WER as a function of context window size (in seconds) for the streaming setup, computed over the dev1-dev set. This figure encompasses both the setup in this section (dashed line) and the setup for the closed-condition system described in Section 5.4 (solid line)

After fixing 0.6s as the acoustic window size, the last step for our latency-focused system was to balance WER against measured latencies for different pruning parameters and LM combinations. In our experiments, latency is measured as the time elapsed between the instant of an acoustic frame and the instant at which it is fully processed by the decoder. Latency figures are reported here at the dataset level, as the average of the latencies observed at a frame level over the whole dataset. Figure 5.2 depicts WER against measured latencies, computed over our internal development set *dev1-dev*. For each LM combination involving the Transformer LM, an exploration of different pruning parameters is represented. All measurements of latencies were performed on an Intel i7-3820 CPU @ 3.60GHz, with 64GB of RAM and a GeForce RTX 2080 Ti GPU.

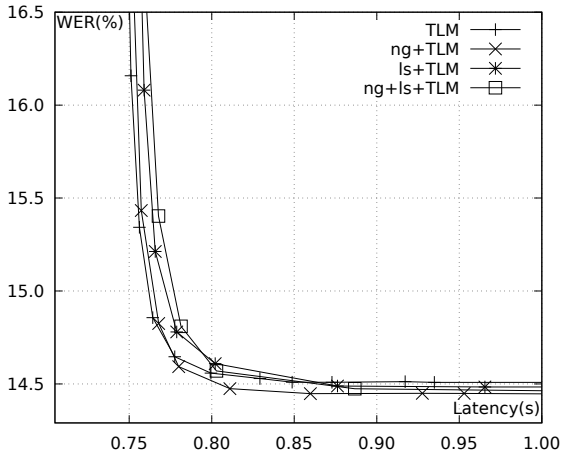

Figure 5.2: WER versus mean empirical latency (in seconds) on *dev1-dev*. Measured for different pruning parameters for the search, considering only LM combinations involving TLM. An acoustic window size of 0.6s was used in all cases.

We can appreciate that combination involving LSTM LMs (ls in the caption of the figure) are systematically shifted rightwards with respect to other combinations, meaning that the addition of the LSTM LMs has a negative impact on system latency, with no worthwhile improvement on output quality. This result corroborates our earlier decision to remove the LSTM LM in defining our contrastive system *c1-streaming_1500ms_nt*. Furthermore, the TLM used alone generally provides a good quality baseline, only ever slightly improved upon in terms of WER when including the other LMs, action that comes at a cost in an increased latency.

Hence, we selected the Transformer LM in isolation as the LM for our latency-focused streaming system. We presented this system as our contrastive system no. 2, identified as *c2-streaming_600ms_t*. Its measured latency on the internal development test *dev1-dev* was 0.81 ± 0.09 s ($\mu \pm \sigma$), and its error rate was of 12.3 and 16.9 WER points on *test-2018* and *test-2020*, respectively. With a very small WER degradation of 6% relative with respect to the primary system, we achieved state-of-the-art, stable empirical latencies. On production, the system has a baseline consumption of 9GB RAM and 3.5GB GPU memory. More than one decoding instance can use it simultaneously, adding 256MB RAM and one CPU thread per each incoming audio stream to be decoded. For instance, decoding four simultaneous audio streams in a single machine would use four CPU threads, 10GB RAM and 3.5GB of GPU memory.

Table 5.5 summarizes the results obtained for all four submitted ASR systems on the *dev2*, *test-2018*, and *test-2020* sets, with the extra addition of the results from our 2018 open-condition submitted system. Surprisingly, the offline system was surpassed by all three streaming systems on the official test set *test-2020*, by up to 1.1 absolute WER points (6% relative). We interpret this outcome as being caused by the Gaussian mixture HMM-based VAD module producing false negatives (speech being erroneously labelled as non-speech). The VAD module was trained with music and noise segments as non-speech, and it may misclassify speech passages with loud background music and/or noise as non-speech, which are often present in TV programmes. Furthermore, the Full Sequence Normalization technique might turn out to be inappropriate for some TV shows, as temporally local acoustic conditions may vary and become diluted in the full-sequence normalization, leading to minor inaccuracies in the acoustic scores that can degrade system performance. Nonetheless, it is remarkable that our 2020 systems significantly outperform the 2018 winning system, in the case of our primary submission by 28% relative WER on boh *dev2* and *test-2018*, and 25% in the case of our latency-focused system *c2-streaming_600ms_t*; while also being able to work under strict streaming conditions.

All these streaming ASR systems can be put into production environments using our custom gRPC-based server-client infrastructure [Str]. Indeed, ASR systems similar to *c2-streaming_600ms_t* and *c1-streaming_1500ms_nt* are already in production at our MLLP Transcription and Translation Platform [TP] for streaming and off-line processing, respectively. Both can be freely tested using our public APIs, accessible via the MLLP Platform.

Table 5.5: WER of the participant systems, including our open-condition system submitted to the 2018 challenge, computed over the *dev2*, *test-2018*, and *test-2020* sets. In bold, the result from our primary submission in the contest main test *test-2020*

System	*dev2*	*test-2018*	*test-2020*
p-streaming_1500ms_nlt	11.2	11.6	**16.0**
c1-streaming_1500ms_nt	–	11.6	16.1
c2-streaming_600ms_t	12.0	12.3	16.9
c3-offline	–	12.0	17.1
2018 open-cond. winner [Jor+18]	15.6	16.5	–

5.4 Closed-Condition Systems

For a more comprehensive comparison with our results from the similar 2018 challenge and to better track our progress these two years, experiments similar to those reported above were carried out using only the data available for the 2018 challenge under closed data conditions. This additional study was not done for the challenge itself, but for a more comprehensive comparison of the state of our technology and its advance during these two years. In this section, we present these experiments and their results.

5.4.1 Acoustic Modelling

As in [Jor+18], acoustic models were trained using only the *train* set together with the *dev1-train* partition that we defined from the *dev1* set (see Table 5.3) for a total of 205 hours, accounting for just 5.2% of the 3924 hours used to train our open challenge submissions. Other than the data, the AMs were built the same as under the open condition and as described in Section 5.3, using MFCC features, HMMs with state tying, BLSTMs, and SpecAugmentation.

5.4.2 Language Modelling

For language modelling, we followed the same steps as indicated in Section 5.3 and trained both n-gram and neural LMs. Similarly, the systems employed the LMs under one-pass decoding with linear combinations of them. We made use of significantly fewer text data to comply with the closed condition constraint. Specifically, we used the same data as in [Jor+18], comprising 5.2 M sentences (1.4% of the full data) and 96 M running words (2.2%) with a vocabulary size of 132 K, obtaining an OOV ratio less than 0.6% computed over our internal development sets and *test2018*, and less than 0.8% over *test2020*.

For the n-gram LM, we used a single model trained with the fraction of the data that was available, in contrast with our submissions to the open challenge, where we employed an interpolation of n-gram models trained on different subsets. As for the neural models, we decided against the use of LSTM LMs, and thus only Transformer LMs were considered. This decision was based on the empirical results reported in Section 5.3.4, with a similar reasoning that led to the removal of the LSTM LM to define our contrastive system *c1-streaming_1500ms_nt*. Apart from this decision and the training data, both the neural architecture and training methodology were kept the same.

5.4.3 Experiments and Results

We reproduced an empirical study similar to that described in Section 5.3 using the systems trained with limited data and the development and test sets detailed in Table 5.3. The first step was to evaluate n-gram and Transformer LMs, as well as their combination, in terms of perplexity on the internal development set *dev1-dev*. Table 5.6 shows the results, including the optimal interpolation weights found for the case of the interpolation. We observe the expected improvement of the Transformer LM over the n-gram LM, but, in contrast to Section 5.3, the interpolation does bring a noticeable improvement in perplexity where the n-gram LM takes a significant weight (30%), accordingly.

Table 5.6: Perplexity (PPL) and interpolation weights, computed over the *dev1-dev* set, of the n-gram (ng) and Transformer (tf) LMs, as well as their combination, when trained with restricted data

LM Comb.	PPL	Weights (%)
ng	164	–
tf	103	–
ng + tf	84	70 + 30

As a second step, we analyzed the size of the acoustic context window, which is equivalent to the theoretical latency. In particular, we studied the effect of the window size on system output quality. Figure 5.1 incorporated the WER as a function of AM context window size for the closed condition exploration, for window sizes under one second, and using the simpler n-gram LM in isolation. The performance of the open-condition system is also shown for a better comparison. From the results in Figure 5.1, and in agreement with the open-condition experimentation performed in Section 5.3, an acoustic window size

of 0.6 s was selected for further experiments designed to build the best possible streaming system considering both WER and real, measured latencies.

For the final streaming and latency tests, we studied the balance between WER and empirical latencies by exploring different pruning parameters and LM combinations. All measurements were made on the same hardware used to assess the open condition systems and described in Section 5.3. Figure 5.3 shows the WER as a function of the mean empirical latency computed over the internal development set *dev1-dev*. For comparison, the corresponding results from the open-condition systems are also included. We can see that in contrast to the open-condition results where the inclusion of the n-gram LM didn't offer much difference over the baseline, here a significant improvement is achieved by combining the n-gram and Transformer LMs, instead of using the Transformer LM alone. In consequence, we chose a system using an interpolated n-gram and Transformer LM to consider our final closed-condition system.

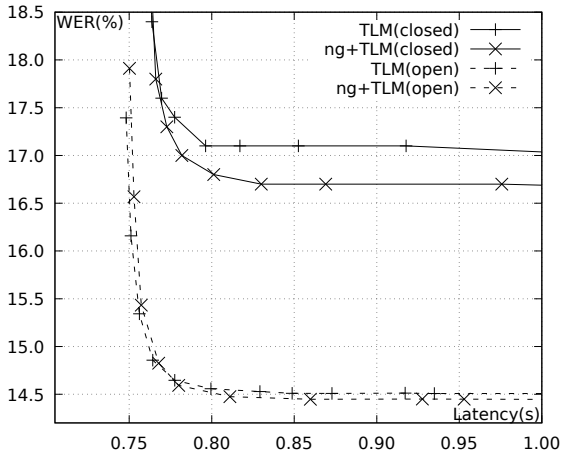

Figure 5.3: WER as a function of measured system latency (in seconds) for the closed-conditions systems (solid line), computed over the *dev1-dev* set by exploring different values for the pruning parameters during search. An acoustic window size of 0.6s was used in all measurements for this plot. For comparison, relevant results from the open-condition systems are also shown (dashed line)

Table 5.7 contains a summary of the most relevant results obtained with our systems, both under open and closed data conditions. For the closed-condition systems, we also included results for LMs trained without data restrictions, to check how far we go without expanding the acoustic data. This scenario better reflects the more usual case where text data is relatively much easier to obtain

than audio with reliable human transcriptions. For comparison purposes, Table 5.7 also includes the figures for the 2018 contest winners and results from other participants in the 2020 challenge.

Table 5.7: WER of our open- and closed-condition systems, together with statistics for the number of hours used for the AM, including for comparison: the winner systems of the 2018 challenge both open- and closed-condition, and other participants for this edition of the challenge. Highlighted in bold, the most relevant results to compare the performance of open- and closed-condition systems in the main test set *test-2020*

System	hours	*dev2*	*test-2018*	*test-2020*
open-p-streaming_ 1500ms_ nlt	3924	11.2	11.6	**16.0**
open-c2-streaming_ 600ms_ t	3924	12.0	12.3	**16.9**
closed-streaming_ 600ms_ nt2018	205	15.0	15.3	23.5
+ open ng		15.3	15.8	22.9
+ open ng+tf		13.0	13.7	**19.9**
closed-streaming_ 1500ms_ nt2018	205	14.7	15.3	23.1
+ open ng		15.3	15.8	22.9
+ open ng+tf		13.0	13.7	**19.9**
2018 open-cond. winner [Jor+18]	3800	15.6	16.5	–
2018 closed-cond. winner [Jor+18]	205	20.0	22.0	–
Vicomtech [Álv+21]	743	–	–	19.3
BRNO [Koc+21]	780	12.8	13.3	23.2
Sigma-UPM [PEH21]	615	–	–	27.7
Biometric Vox System [FG21]	1000	17.8	22.0	30.3

From the results in *test-2020*, we can observe that we can still obtain an acceptably well-performing streaming ASR system using only around 5% of the acoustic data and 2% of the text data, while there still is a wide margin between the open- and closed-condition systems. Interestingly, the WER reduction entailed by moving from an acoustic latency of 1.5s to 0.6s in the open-condition systems is largely gone in the closed-condition systems: completely negligible on *test-2018* and just 0.5 WER on *test-2020* in the case of text data restrictions being applied. Lifting the text data restrictions, the large-latency closed-condition system *closed-streaming_ 1500ms_ nt2018* is able to shed over 3 WER points, from 23.1 to 19.9, representing about a 15% relative improvement and landing among the best performing systems submitted by other participants under open conditions.

Regarding the change from the closed-condition LM to the unrestricted LM, both in the 1500ms and the 600ms window size systems, most of the improvement comes from the substitution errors: from 10.7 to 7.3 in the 1500ms case,

and from 10.9 to 7.4 in the 600ms case. This is due mostly to the unrestricted LM retrieving the correct word due to its expanded lexicon (as previously mentioned, the OOV rate falls from 0.8 to 0.6) and better tuned prediction probabilities. Some examples of words that are only correctly recognized with an unrestricted LM are "desacreditarme", "dosificador" or "hermeneuta".

It is to be remarked that our submission two years prior [Jor+18] achieved 16.5 WER on *test-2018*, achieving the best results of that contest, while here we obtained 11.6 WER points with a comparable amount of training hours (3924 vs. 3800). The decoder structure improvements and the Transformer LM used as described in this work are two of the factors that are involved, but we are unable to pinpoint which of the many changes are more critical in the overall improvement of performance in our systems.

5.5 Conclusions

In this chapter, we describe our participation in the Albayzín-RTVE 2020 Speech-to-Text Challenge with four ASR systems. The primary one, a performance-focused hybrid system *p-streaming_1500ms_nlt*, provided a score of 16.0 WER points on the contest test set *test-2020*, winning the challenge by a margin larger than 3 points from the second position participant while also being enabled for a streaming use, a technical feat that no other participant in the contest included in their systems, although with a relatively high theoretical latency of 1.5s, as latency was not the primary focus for this system. It also achieved a remarkable 28% relative WER improvement over the 2018 winning ASR system on *test-2018*. Almost the same performance was delivered by our first contrastive system *c1-streaming_1500ms_nt*: 16.1 WER points on *test-2020*, at a significantly lower computational expense. Focused on low latencies, our second contrastive system *c2-streaming_600ms_t* provided a solid performance of 16.9 WER points on *test-2020* with a theoretical latency of 0.6s and an empirical latency of 0.81 ± 0.09 ($\mu \pm \sigma$). Finally, our fourth ASR system was a contrastive off-line ASR system with a VAD module, *c3-offline*, providing a WER score of 17.1 points, somewhat higher than our streaming systems, fact that we attribute to the VAD module not being finely tuned to the particular task and to the limitations of Full Sequence Normalization when dealing with changes in local acoustic conditions.

In addition to the four ASR systems participating in the 2020 challenge, which included no restrictions to training data, we describe two additional systems trained under the same data restrictions as under the 2018 closed

challenge, for better comparison with those results and to better gauge the progress of our technology over these two years. Closed data conditions represent about 5% of the speech data and 2% of the text data that we employed to train our submitted systems. The first of the data-restricted systems, *closed-streaming_600ms_nt2018*, was focused on latency and achieved 23.5 WER points on *test-2020*, while a performance-focused system *closed-streaming_1500ms_nt2018* reduced this number to 23.1 WER points. Lifting the text data restriction and using their AMs with the unconstrained LMs, these figures are further reduced to 20.4 (at 600ms window size) and 19.9 (at 1500ms window size), an excellent quality of transcription using only 205h of RTVE2018 speech training data and thus not including any data related to the last two years of speech and accounting for only 5% of the total speech training data available. Nevertheless, these WER figures are not far behind those of the second-best 2020 open-condition system (19.3) and remain significantly ahead of results reported by other participants.

Part of the work contained in this chapter was done in collaboration with other members of the MLLP research group. In particular, the evaluation of the systems submitted to the contest was organized jointly with Javier Jorge, while the author of the thesis is the primary responsible for the second part where equivalent data-restricted systems were built and evaluated, systems that were not submitted to the contest but served to check that the improvement in technological capabilities developed in Chapter 4 is the main cause of system performance improvements.

Chapter 6

Conclusions and future work

6.1 Scientific and technological achievements

In this section, we discuss the achievement of the scientific and technological goals of this thesis formulated in Section 1.2. These goals were centered around MT and ASR, two fundamental keystones of human language computer technologies, bringing the latest neural architecture into action by reaching excellent results in some MT tasks and demonstrably improving the state-of-the-art in streaming ASR technological capabilities.

The first scientific goal of the thesis is covered in Chapter 3, where the fresh Transformer neural architecture was used to achieve excellent results in competitive MT challenges, namely the WMT18 German→English news translation task and the WMT19 similar language Portuguese↔Spanish translation task.

Having verified the capabilities of the Transformer as a translation model, Chapter 4 centers around the second scientific goal of this thesis: improving the state-of-the-art in streaming ASR technology. To achieve this, we leveraged the "decoder" part of the Transformer architecture as a language model and two adaptations were made to, in addition, make it suitable for streaming use.

The third and final goal is addressed in Chapter 5, where we applied the technical capabilities developed in Chapter 4 to a relevant use case, namely the Albayzín-RTVE 2020 Speech-to-Text challenge issued by RTVE, the main publich TV broadcaster of Spain. We demonstrated the best performance in the competition, by a large margin, and we performed additional data-restricted experiments afterwards to check that the improvements were primarily ex-

plained by our technological progress and not to the particular training/test data.

6.2 Publications

The work described in this thesis has directly yielded articles in international workshops, conferences, and journals. This section enumerates these contributions and states their relationship with the chapters of this thesis.

The work in competitive MT challenges leveraging the Transformer architecture to achieve excellent results in news translation and the first position in similar language translation presented in Chapter 3 resulted in a couple of workshop publications. The work was done in team and the author of this thesis is the first author of one of the two publications:

- Baquero-Arnal, P.; Iranzo-Sánchez, J.; Civera, J.; Juan, A. *The MLLP-UPV Spanish-Portuguese and Portuguese-Spanish Machine Translation Systems for WMT19 Similar Language Translation Task.* Proc. of WMT19, 2019, 179–184.
- Iranzo-Sánchez, J.; Baquero-Arnal, P.; Garcés Díaz-Munío, G. V.; Martínez-Villaronga, A.; Civera, J.; Juan, A. *The MLLP-UPV German-English Machine Translation System for WMT18.* Proc. of WMT18, 2018, 422–428.

The development of new technical capabilities to improve the state-of-the-art in Streaming ASR by adapting the Transformer decoder to serve as a LM with specific streaming adaptations, described in Chapter 4, yielded the following publication in an international conference:

- Baquero-Arnal, P.; Jorge, J.; Giménez, A.; Silvestre-Cerdà, J. A.; Iranzo-Sánched, J.; Sanchis, A.; Civera, J.; Juan, A. *Improved Hybrid Streaming ASR with Transformer Language Models.* Proc. of Interspeech 2020, 2127–2131.

The work related to the participation in the Albayzín-RTVE 2020 Speech-to-Text Challenge with systems leveraging the developed improved streaming ASR technology presented in Chapter 5 produced the following article in international journal, following an invite from the challenge organization as the winners of the contest to provide an extension to our work:

- Baquero-Arnal, P.; Jorge, J.; Giménez, A.; Iranzo-Sánchez, J.; Pérez, A.; Garcés Díaz-Munío, G.V.; Silvestre-Cerdà, J.A.; Civera, J.; Sanchis, A.; Juan, A. *MLLP-VRAIN Spanish ASR Systems for the Albayzín-RTVE 2020 Speech-to-Text Challenge: Extension.* Applied Sciences. 2022, 12, 804.

Finally, other publications not directly related to this thesis have been produced in collaboration with the MLLP-VRAIN research group:

- Iranzo-Sánchez, J.; Jorge, J.; Pérez-González-de-Martos, A.; Giménez, A.; Garcés Díaz-Munío, G. V.; Baquero-Arnal, P.; Silvestre-Cerdà, J. A.; Civera, J.; Sanchis, A.; Juan, A. *MLLP-VRAIN UPV Systems for the IWSLT 2022 Simultaneous Speech Translation and Speech-to-Speech Translation Tasks.* Proc. of IWSLT 2022, 255–264.

- Pérez González de Martos, A.; Giménez Pastor, A.; Jorge Cano, J.; Iranzo-Sánchez, J.; Silvestre-Cerdà, Joan A.; Garcés Díaz-Munío, G. V.; Baquero-Arnal, P.; Sanchis Navarro, A.; Civera Sáiz, J.; Juan Ciscar, A.; Turró Ribalta, C. *Doblaje Automático de Vídeo-Charlas Educativas en UPV[Media].* Proc. of IN-RED 2022, Forthcoming.

- Jorge, J.; Giménez, A.; Baquero-Arnal, P.; Iranzo-Sánchez, J.; Pérez, A.; Garcés Díaz-Munío, G.V.; Silvestre-Cerdà, J.A.; Civera, J.; Sanchis, A.; Juan, A. *MLLP-VRAIN Spanish ASR Systems for the Albayzín-RTVE 2020 Speech-to-Text Challenge.* Proc. of IberSPEECH 2021, 118–122.

- Iranzo-Sánchez, J.; Jorge, J.; Baquero-Arnal, P.; Silvestre-Cerdà, J. A.; Giménez, A.; Civera, J.; Sanchis, A.; Juan, A. *Streaming Cascade-Based Speech Translation Leveraged by a Direct Segmentation Model.* Neural Networks, 142, 303-315, 2021.

- Garcés Díaz-Munío, G. V.; Silvestre-Cerdà, J. A.; Jorge, J.; Giménez, A.; Iranzo-Sánchez, J.; Baquero-Arnal, P.; Roselló, N.; Pérez, A.; Civera, J.; Sanchis, A.; Juan, A. *Europarl-ASR: A Large Corpus of Parliamentary Debates for Streaming ASR Benchmarking and Speech Data Filtering/Verbatimization.* Proc. Interspeech 2021, 3695–3699.

- Iranzo-Sánchez, J.; Giménez Pastor, A.; Silvestre-Cerdà, J. A.; Baquero-Arnal, P.; Civera, J.; Juan, A. *Direct Segmentation Models for Streaming Speech Translation.* Proc. of EMNLP 2020, 2599–2611.

- Valor Miró, J. D.; Baquero-Arnal, P.; Civera, J.; Turró, C.; Juan, A. *Multilingual Videos for MOOCs and OER*. Journal of Educational Technology and Society, 21 (2), 1–12, 2018.

6.3 Future work

This work started on MT with the latest (at that time) and fresh Transformer model and pivoted to ASR when it became clear that the improvements in MT could be realized in other sequence-related tasks. Streaming ASR was chosen as it is a field in which the state-of-the-art is relatively hard to improve, since the most popular end-to-end approaches struggle to work with sequences of unbounded length and hybrid systems retain the best performances. In this thesis, the Transformer architecture was leveraged as a language model for streaming ASR systems.

As a future work, we intend to further improve the streaming hybrid ASR architecture by also adapting the Transformer architecture, or some of its variants, to do the work of the acoustic model. We are also interested in devising some mechanism to make the end-to-end models work correctly in a streaming fashion so that long-form speeches can be adequately transcribed by the system. Finally, we will also continue to explore the potential application of these technologies in the media industry, where automatic speech recognition and machine translation is becoming a standard element among the tools used by professional subtitlers and translators.

List of Figures

List of Tables

Acronyms

AM Acoustic Model. 22, 61, 62, 65, 66, 69, 70, 72, 74, 82

ASR Automatic Speech Recognition. 1–7, 11, 14, 22–24, 43–45, 47, 50–57, 60, 61, 63–65, 68, 72, 73, 75, 76, 78

BLEU Bilingual Evaluation Understudy. 21, 32–34, 36–38, 40

BLSTM Bidirectional Long Short-Term Memory. 48, 51, 52, 62, 65, 66, 69

BPE Byte Pair Encoding. 21, 31, 33, 36

DNN Deep Neural Network. 50, 60, 62

EM Estimation-Maximization. 23

GMM Gaussian Mixture Model. 22, 23

GRU Gated Recurrent Unit. 14, 26, 37

HCS History Conditioned Search. 44, 63

HMM Hidden Markov Model. 22, 23, 44, 50, 51, 60, 62, 65, 66, 68, 69, 79

LM Language Model. 9, 11, 20, 38, 44–52, 54–57, 62–74, 76, 81

LMHP Language Model Histogram Pruning. 48

LMHR Language Model History Recombination. 46, 47

LSTM Long Short-Term Memory. 13, 14, 26, 44, 50–52, 54–57, 62–65, 67, 70, 79, 81

MFCC Mel Frequency Cepstral Coefficients. 61, 69

ML Machine Learning. 5, 6, 9

MLLP Machine Learning and Language Processing. 1, 35, 40, 41, 68, 74, 77

MT Machine Translation. 1–7, 11, 14, 20–22, 25, 26, 28, 29, 35, 36, 40, 75, 76, 78, 81

NCE Noise Constrastive Estimation. 63

NLP Natural Language Processing. 17

NMT Neural Machine Translation. 20, 21, 25, 26, 29–31, 33–35, 38, 40

OER Open Educational Resources. 1

OOV Out Of Vocabulary. 50, 63, 69, 73

PBMT Phrase-Based Machine Translation. 20, 21, 25, 31

PPL Perlexity. 51–56, 64, 65

RNN Recurrent Neural Network. 5, 12–17, 21, 26, 35–37, 39, 40, 50, 79

RTF Real Time Factor. 51–56, 63, 64

RTVE Radio Televisión Española. 4, 59, 60, 75

SGD Stochastic Gradient Descent. 8

SMT Statistical Machine Translation. 20

TER Translation Edit Rate. 21, 32–34, 38, 40

TLM Transformer Language Model. 44, 49–54, 57, 58, 63–65, 67, 80

UPV Universitat Politècnica de València. 1–4

VAD Voice Activity Detection. 60, 65, 68, 73

VRAIN Valencian Research Institute for Artificial Intelligence. 1, 77

WER Word Error Rate. 24, 47, 51–57, 60, 61, 63–68, 70–74

Bibliography

[07] *Royal Decree 1494/2007 (Spain) on Accessibility to the Media*. Available online: `https://www.boe.es/buscar/act.php?id=BOE-A-2007-19968` (Accessed on 28 July 2022). 2007 (cit. on p. 59).

[08] *Law 1/2006 (Comunitat Valenciana, Spain) on the Audiovisual Sector*. Available online: `https://www.dogv.gva.es/va/eli/es-vc/l/2006/04/19/1/dof/vci-spa/pdf` (Accessed on 28 July 2022). 2008 (cit. on p. 59).

[18] *WMT18 organizers. WMT18 Shared Task: Machine Translation of News*. `http://www.statmt.org/wmt18/translation-task.html`. Accessed: 2022-06-28. 2018 (cit. on p. 29).

[201] CommonCrawl 2014. Available online: `https://commoncrawl.org` (accessed on 1 August 2022) (cit. on p. 62).

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015 (cit. on pp. 51, 62).

[AHG11] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. "Domain Adaptation via Pseudo In-domain Data Selection". In: *Proc. of the Conference on EMNLP*. Stroudsburg, Pennsylvania, USA, 2011, pp. 355–362 (cit. on p. 30).

[ALA19] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. "An Effective Approach to Unsupervised Machine Translation". In: *Proc. of the 57th Annual Meeting of the ACL*. Florence, Italy, July 2019, pp. 194–203 (cit. on p. 6).

[Álv+21] Aitor Álvarez et al. "The Vicomtech Speech Transcription Systems for the Albayzín-RTVE 2020 Speech to Text Transcription Challenge". In: *Proc. of IberSPEECH*. Valladolid, Spain, 2021, pp. 104–107 (cit. on pp. 60, 72).

[Bah+15] Dzimitry Bahdanau et al. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*. San Diego, California, USA, May 2015 (cit. on p. 14).

[BBN18] Parnia Bahar, Christopher Brix, and Hermann Ney. "Towards Two-dimensional Sequence to Sequence Model in Neural Machine Translation". In: *Proc. of the EMNLP*. Brussels, Belgium, 2018, pp. 3009–3015 (cit. on p. 26).

[BM94] Herve Bourlard and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. 1994. ISBN: 978-1-4613-6409-2 (cit. on p. 23).

[Boj+18] Ondrej Bojar et al. "Findings of the 2018 Conference on Machine Translation (WMT18)". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 272–303 (cit. on p. 34).

[Boy+16] Ondrej Boyar et al. "Findings of the WMT17". In: *Proc. of the First Conference on Machine Translation*. Berlin, Germany, 2016, pp. 131–198 (cit. on p. 25).

[Bro+20] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Proc. of NeurIPS*. 2020, pp. 1877–1901 (cit. on p. 11).

[Che+15] Xie Chen et al. "Improving the Training and Evaluation Efficiency of Recurrent Neural Network Language Models". In: *Proc. of ICASSP*. Brisbane, Queensland, Australia, 2015, pp. 5401–5405 (cit. on p. 63).

[Che+16] Xie Chen et al. "CUED-RNNLM — An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *Proc. of ICASSP*. Lujiazui, Shanghai, China, 2016, pp. 6000–6004 (cit. on pp. 51, 63).

[Che+21] Xie Chen et al. "Developing Real-Time Streaming Transformer Transducer for Speech Recognition on Large-Scale Dataset". In: *Proc. of ICASSP*. Toronto, Ontario, Canada, 2021, pp. 5904–5908 (cit. on p. 24).

[Cho+14] Kyunghyun Cho et al. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proc. of SSSST-8*. Doha, Qatar: ACL, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012 (cit. on p. 14).

[Cor] UFAL Medical Corpus. Available online: http://ufal.mff.cuni.cz/ufal_medical_corpus (accessed on 1 August 2022) (cit. on p. 62).

[Del+14] Miguel Del Agua et al. "The TransLectures-UPV Toolkit". In: *proc. of IberSPEECH*. Las Palmas de Gran Canaria, Spain, 2014, pp. 269–278 (cit. on pp. 50, 60, 62).

[Dev+18] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proc. of NACL*. Minneapolis, Minnesota, USA, 2018, pp. 4171–4186 (cit. on p. 11).

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *JMLR* 12 (2011), pp. 2121–2159. ISSN: 1532-4435 (cit. on p. 9).

[EBV18] Maga Elbayad, Lauretn Besacier, and Jakob Verbeek. "Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction". In: *Proc. of the 22nd CCNLP*. Brussels, Belgium, 2018, pp. 97–107 (cit. on p. 26).

[Eld] Eldiario.es. Available online: https://www.eldiario.es (accessed on 1 August 2022) (cit. on p. 62).

[FG21] Roberto Font and Teresa Grau. "The Biometrix Vox System for the Albayzin-RTVE 2020 Speech-to-Text Challenge". In: *Proc. of IberSPEECH*. Valladolid, Spain, 2021, pp. 99–103 (cit. on p. 72).

[FGK10] George Foster, Cyril Goutte, and Roland Kuhn. "Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation". In: *Proc. of the 2010 Conference on EMNLP*. Cambridge, Massachussets, USA, 2010, pp. 451–459 (cit. on p. 30).

[Gra+18] Miguel Graça et al. "The RWTH Aachen University English-German and German-English Unsupervised Neural Machine Translation Systems for WMT 2018". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 377–385 (cit. on p. 35).

[GS00] F.A. Gers and J. Schmidhuber. "Recurrent nets that time and count". In: *Proc. of the IJCNN*. Vol. 3. Como, Italy, Feb. 2000, pp. 189–194 (cit. on p. 14).

[Gul+20] Anmol Gulati et al. "Conformer: Convolution-augmented transformer for speech recognition". In: *arXiv preprint arXiv:2005.08100* (2020) (cit. on p. 23).

[Had+18] Barry Haddow et al. "The University of Edinburgh's Submissions to the WMT18 News Translation Task". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 399–409 (cit. on p. 35).

[Hie+18] Felix Hieber et al. "The Sockeye Neural Machine Translation Toolkit". In: *Proc. of the 13th Conference of the AMTA*. 2018 (cit. on p. 31).

[Hin14] George Hinton. *Lecture 6e on neural networks (RMSprop: Divide the gradient by a running average of its recent magnitude)*. 2014 (cit. on p. 9).

[Hor+07] Takaaki Hori et al. "Efficient WFST-Based One-Pass Decoding With On-The-Fly Hypothesis Rescoring in Extremely Large Vocabulary Continuous Speech Recognition". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.4 (2007), pp. 1352–1365 (cit. on p. 51).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 13).

[Huc+18] Matthias Huck et al. "LMU Munich's Neural Machine Translation Systems at WMT 2018". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 648–654 (cit. on p. 35).

[Hun90] Melvyn J. Hunt. "Figures of merit for assessing connected-word recognisers". In: *Speech Communication* 9.4 (1990), pp. 329–336 (cit. on p. 22).

[HZD14] Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin. "Cache based recurrent neural network language model inference for first pass speech recognition". In: *Proc. of ICASSP*. Florence, Italy, 2014, pp. 6354–6358 (cit. on p. 46).

[Iri+19] Kazuki Irie et al. "Language Modeling with Deep Transformers". In: *Proc. of Interspeech*. Graz, Austria, 2019, pp. 3905–3909 (cit. on p. 44).

[Jor+18] Javier Jorge et al. "MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge". In: *Proc. of IberSPEECH*. Barcelona, Spain, 2018, pp. 257–261 (cit. on pp. 60, 61, 64, 65, 69, 72, 73).

[Jor+19] Javier Jorge et al. "Real-Time One-Pass Decoder for Speech Recognition Using LSTM Language Models". In: *Proc. of Interspeech*. Graz, Austria, 2019, pp. 3820–3824 (cit. on p. 44).

[Jor+20] Javier Jorge et al. "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP*. Barcelona, Spain, 2020, pp. 7814–7818 (cit. on pp. 44, 56).

[Kal+15] Nal Kalchbrenner et al. "Grid long short-term memory". In: *arXiv preprint arXiv:1507.01526* (2015) (cit. on p. 26).

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *Proc. of the 3rd ICLR*. San Diego, California, USA, July 2015 (cit. on pp. 9, 31, 37).

[KDT18] Philipp Koehn, Kevin Duh, and Brian Thompson. "The JHU Machine Translation Systems for WMT 2018". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 438–444 (cit. on p. 35).

[Koc+21] Martin Kocour et al. "BCN2BRNO: ASR System Fusion for Albayzin 2020 Speech to Text Challenge". In: *Proc. of IberSPEECH*. Valladolid, Spain, 2021, pp. 113–117 (cit. on p. 72).

[Koe+07] Philipp Koehn et al. "Moses: Open Source Toolkit for Statistical Machine Translation". In: *Proc. of the 45th annual meeting of the ACL*. Prague, Czech Republic, 2007, pp. 177–180 (cit. on pp. 29, 36).

[Kud18] Taku Kudo. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates". In: *Proc. of the 56th ACL*. Melbourne, Victoria, Australia, 2018, pp. 66–75 (cit. on p. 21).

[Lee88] Kai-Fu Lee. "On large-vocabulary speaker-independent continuous speech recognition". In: *Speech communication* 7.4 (1988), pp. 375–379 (cit. on p. 22).

[Lin+12] Yuri Lin et al. "Syntactic annotations for the Google Books Ngram Corpus". In: *Proc. of the ACL System Demonstrations*. Jeju Island, Korea, 2012, pp. 169–174 (cit. on p. 62).

[Lle+18] Eduardo Lleida et al. *RTVE2018 Database Description*. Available online: http://catedrartve.unizar.es/reto2018/RTVE2018DB.pdf (accessed on 28 July 2022). 2018 (cit. on p. 60).

[Lle+19] Eduardo Lleida et al. "Albayzin 2018 Evaluation: The IberSpeech-RTVE Challenge on Speech Technologies for Spanish Broadcast Media". In: *Applied Sciences* 9.24 (2019). ISSN: 2076-3417 (cit. on p. 61).

[Lle+20a] Eduardo Lleida et al. *Albayzin Evaluation: IberSPEECH-RTVE 2020 Speech to Text Transcription Challenge*. Available online: `http://catedrartve.unizar.es/reto2020/EvalPlan-S2T-2020-v1.pdf` (accessed on 28 July 2022). 2020 (cit. on p. 60).

[Lle+20b] Eduardo Lleida et al. *RTVE2020 Database Description.* Available online: `http://catedrartve.unizar.es/reto2020/RTVE2020DB.pdf` (accessed on 28 July 2022). 2020 (cit. on p. 61).

[LM15] Minh-Thang Luong and Christopher Manning. "Stanford Neural Machine Translation Systems for Spoken Language Domain". In: *Proc. of the 13th International Workshop on Spoken Language Translation: Evaluation Campaign*. Da Nang, Vietnam, 2015, pp. 76–79 (cit. on p. 38).

[Lu+20] Liang Lu et al. "Exploring Transformers for Large-Scale Speech Recognition". In: *proc. of Interspeech*. Shanghai, China, 2020 (cit. on p. 24).

[Lüs+19] Christoph Lüscher et al. "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention - w/o Data Augmentation". In: *Proc. of Interspeech*. Graz, Austria, 2019, pp. 231–235 (cit. on p. 57).

[Mar22] Alejandro Pérez González de Martos. "Deep Neural Networks for Automatic Speech-To-Speech Translation of Open Educational Resources". PhD thesis. Universitat Politècnica de València, 2022 (cit. on p. 19).

[MHL20] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. "Streaming Automatic Speech Recognition with the Transformer Model". In: *Proc. of ICASSP*. Barcelona, Spain, 2020, pp. 6074–6078 (cit. on pp. 43, 57).

[Mia+20a] Haoran Miao et al. "Transformer-Based Online CTC-Attention End-To-End Speech Recognition Architecture". In: *Proc. of ICASSP*. Barcelona, Spain, 2020, pp. 6084–6088 (cit. on p. 43).

[Mia+20b] Haoran Miao et al. "Transformer-based online CTC/attention end-to-end speech recognition architecture". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6084–6088 (cit. on p. 23).

[Moh+15] Abdel-rahman Mohamed et al. "Deep Bi-directional Recurrent Networks over Spectral Windows". In: *Proc. of ASRU*. Scottsdale, Arizona, USA, 2015, pp. 78–83 (cit. on p. 48).

[MSN18] Makoto Morishita, Jun Suzuki, and Masaaki Nagata. "NTT's Neural Machine Translation Systems for WMT 2018". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 461–466 (cit. on p. 35).

[MT12] Andriy Mnih and Yee Whye Teh. "A Fast and Simple Algorithm for Training Neural Probabilistic Language Models". In: *Proc. of ICML*. Madison, Wisconsin, USA, 2012, pp. 419–426 (cit. on pp. 51, 63).

[Mur22] Kevin P Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022 (cit. on pp. 5, 12).

[NEK94] Hermann Ney, Ute Essen, and Reinhard Kneser. "On Structuring Probabilistic Dependencies in Stochastic Language Modelling". In: *Computer Speech and Language* 8 (1994), pp. 1–38. ISSN: 0885-2308 (cit. on p. 11).

[NO00] Hermann Ney and Stefan Ortmanns. "Progress in Dynamic Programming Search for LVCSR". In: *Proc. of the IEEE* 88 (2000), pp. 1224–1240 (cit. on p. 44).

[Nol17] David Nolden. "Progress in Decoding for Large Vocabulary Continuous Speech Recognition". Dissertation. Aachen: RWTH Aachen University, 2017 (cit. on pp. 44, 45, 63).

[Ope] OpenSubtitles. Available online: `http://www.opensubtitles.org` (accessed on 1 August 2022) (cit. on p. 62).

[Ott+19] Myle Ott et al. "Fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proc. of the 2019 Conference of the North American Chapter of the ACL*. Minneapolis, Minnesota, USA, 2019, pp. 48–53 (cit. on pp. 37, 51, 63).

[Pan+15] Vassil Panayotov et al. "LibriSpeech: An ASR Corpus Based on Public Domain Audio Books". In: *Proc. of ICASSP*. Brisbane, Queensland, Australia, 2015, pp. 5206–5210 (cit. on p. 50).

[Pap+02] Kishore Papineni et al. "BLEU: a Method for Automatic Evaluation of Machine Translation". In: *Proc. of the 40th annual meeting of the ACL*. 2002, pp. 311–318 (cit. on pp. 21, 32, 36).

[Par+19] Daniel Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. of Interpseech*. Graz, Austria, 2019, pp. 2613–2617 (cit. on p. 62).

[PEH21] Juan M. Perero-Codosero, Fernando M. Espinoza-Cuadros, and Luis A. Hernández-Gómez. "Sigma-UPM ASR Systems for the IberSpeech-RTVE 2020 Speech-to-Text Transcription Challenge". In: *Proc. of IberSPEECH*. Valladolid, Spain, 2021, pp. 108–112 (cit. on p. 72).

[Per] El Periódico. Available online: `https://www.elperiodico.com` (accessed on 1 August 2022) (cit. on p. 62).

[PEV20] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. "BPE-Dropout: Simple and Effective Subword Regularization". In: *Proc. of the 68th ACL*. Online, 2020, pp. 1882–1892 (cit. on p. 21).

[RDE14] Anthony Rousseau, Paul Deléglise, and Yannick Estève. "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks". In: *Proc. of LREC*. Reykjavik, Iceland, 2014, pp. 3935–3939 (cit. on pp. 50, 51).

[SGB18] Felix Stahlberg, Adrià de Gispert, and Bill Byrne. "The University of Cambridge's Machine Translation Systems for WMT18". In: *Proc. of WMT18*. Brussels, Belgium, 2018, pp. 504–512 (cit. on p. 35).

[SHB16a] Richo Sennrich, Barry Haddow, and Alexandra Birch. "Improving Neural Machine Translation Models with Monolingual Data". In: *Proc. of the 54th ACL*. Berlin, Germany, 2016, pp. 86–96 (cit. on pp. 21, 29, 30, 38).

[SHB16b] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *Proc. of the 54th ACL*. Berlin, Germany, 2016, pp. 1715–1725 (cit. on pp. 21, 31).

[Shi+14] Yongze Shi et al. "Efficient One-Pass Decoding with NNLM for Speech Recognition". In: *IEEE Signal Processing Letters* 21.4 (2014), pp. 377–381. DOI: `10.1109/LSP.2014.2303136` (cit. on pp. 45, 63).

[Sno+06] Matthew Snover et al. "A Study of Translation Edit Rate with Targeted Human Annotation". In: *Proc. of the 7th AMTA*. Cambridge, Massachussets, USA, 2006, pp. 223–231 (cit. on pp. 21, 32).

[Sri+14] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958. ISSN: 1532–4435 (cit. on p. 36).

[Sto+11] Andreas Stolcke et al. "SRILM at Sixteen: Update and Outlook". In: *Proc. of the 2011 IEEE Workshop on ASRU*. Waikoloa, Hawaii, USA, 2011 (cit. on pp. 30, 38, 51, 62).

[Str] MLLP gRPC Streaming API. Available online: `https://mllp.upv.es/git-pub/jjorge/MLLP_Streaming_API` (accessed on 9 September 2022) (cit. on p. 68).

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Proc. of the 27th Int. Conf. on NIPS - Volume 2*. Montreal, Quebec, Canada: MIT Press, 2014, pp. 3104–3112 (cit. on p. 14).

[Sze+16] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on CVPR*. Las Vegas, Nevada, USA, 2016, pp. 2818–2826 (cit. on p. 37).

[TKW21] Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. "Streaming Transformer Asr With Blockwise Synchronous Beam Search". In: *Proc. of IEEE SLT Workshop*. 2021, pp. 22–29 (cit. on p. 24).

[TP] MLLP Transcription and Translation Platform. Available online: `https:/ttp.mllp.upv.es` (accessed on 9 September 2022) (cit. on p. 68).

[Uni14] United Kingdom Office of Communications. *Measuring live subtitling quality. Results from the first sampling exercise*. 2014 (cit. on p. 57).

[VA21] Valentin Vielzeuf and Grigory Antipov. "Are E2E ASR models ready for an industrial usage?" In: *arXiv preprint arXiv:2112.12572* (2021) (cit. on p. 23).

[Vas+17] Ashish Vaswani et al. "Attention is All you Need". In: *Proc. of NIPS*. 2017, pp. 5998–6008 (cit. on pp. 17–19, 31, 35).

[Vox] Voxforge.org. `http://www.voxforge.org/` (accessed on 28 July 2022) (cit. on p. 61).

[Wan+20] Yongqiang Wang et al. "Transformer-based acoustic modeling for hybrid speech recognition". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6874–6878 (cit. on p. 23).

[Wik] Wikipedia. Available online: `https://www.wikipedia.org` (accessed on 1 August 2022) (cit. on p. 62).

[WMT] NewsCrawl corpus (WMT Workshop) 2015. Available online: `http://statmt.org/wmt15/translation-task.html` (accessed on 1 August 2022) (cit. on p. 62).

[WSM] UN corpus (WSMT Workshop) 2012. Available online: `https://www.statmt.org/wmt12/translation-task.html` (accessed on 1 August 2022) (cit. on p. 62).

[Xie+22] Yi Xie et al. "Compute cost amortized transformer for streaming ASR". In: *Proc. of Interspeech*. Incheon, Korea, 2022 (cit. on p. 24).

[Xue+22] Jian Xue et al. "Large-Scale Streaming End-to-End Speech Translation with Neural Transducers". In: *Proc. of Interspeech*. Incheon, Korea, 2022 (cit. on p. 24).

[Yas+08] Keiji Yasuda et al. "Method of Selecting and Training Data to Build a Compact and Efficient Translation Model". In: *Proc. of the third JCNLP*. Hyderabad, Andhra Pradesh, India, 2008, pp. 655–660 (cit. on p. 30).

[YOW94] Steve J. Young, James J. Odell, and Philip C. Woodland. "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proc. of the workshop on HLT*. Plainsboro, New Jersey, USA, 1994, pp. 307–312 (cit. on pp. 51, 62).

[Yu+21] Jiahui Yu et al. "Dual-mode ASR: Unify and Improve Streaming ASR with Full-context Modeling". In: *ICLR 2021*. 2021 (cit. on p. 24).

[Zey+17] Albert Zeyer et al. "A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition". In: *Proc. of ICASSP*. New Orleans, Louisiana, USA, 2017, pp. 2462–2466 (cit. on pp. 51, 62).

[Zey+19] Albert Zeyer et al. "A Comparison of Transformer and LSTM Encoder Decoder Models for ASR". In: *Proc. of ASRU*. Sentosa, Singapore, 2019, pp. 8–15 (cit. on p. 43).

[Zha+20] Qian Zhang et al. "Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss". In: *Proc. of ICASSP*. Barcelona, Spain, 2020, pp. 7829–7833 (cit. on pp. 43, 57).

[Zho+20] Wei Zhou et al. "The RWTH ASR System for TED-LIUM Release 2: Improvimg Hybrid HMM with SpecAugment". In: *Proc. of ICASSP*. Barcelona, Spain, 2020, pp. 7839–7843 (cit. on p. 57).

[ZSN16] Albert Zeyer, Ralf Schlüter, and Hermann Ney. "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Interspeech*. 2016, pp. 3424–3428 (cit. on p. 48).