UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Streaming Automatic Speech Recognition with Hybrid Architectures and Deep Neural Network Models

April 2022

Author: Javier Jorge Cano

Supervisors: D. Alfons Juan Ciscar
D. Jorge Civera Saiz

Agradecimientos

Lo primero, quiero remarcar que este texto solo refleja con meras palabras una parte de la gratitud total que siento hacia todas esas personas que me han acompañado durante este periodo de mi vida. Ellos y ellas saben mejor que nadie lo que significa este viaje, las dificultades y las alegrías que implica, y lo necesario que ha sido su apoyo en las distintas etapas del proceso. Enumeraré algunas de estas personas, pero la lista va más allá de los aquí reflejados, extiendo mi gratitud a ellos y ellas también, ya lo saben.

Querría agradecer a Alfons, Jorge y Albert en primer lugar por haberme dado la oportunidad de entusiasmarme de nuevo con la investigación y el trabajo en equipo, por creer en mi trabajo y darme la oportunidad de trabajar con las personas más brillantes que he conocido. Por otro lado, querría agradecer a los miembros del EQUIPO MLLP, con mayúsculas intencionadas, ya que en contadas ocasiones he encontrado un grupo de personas que den un significado tan positivo a esta palabra. Gracias a Adrià Giménez, Adrià Martínez, Pau, Álex, Javier Iranzo, Joan Albert y Gonçal, ha sido un placer trabajar codo con codo, aprender y mejorar con vosotros. Querría reservar gratitud adicional para poder compensar de alguna forma a Adrià Giménez todas las horas de conversaciones sobre ideas locas que hemos tenido durante estos años, ya que me han ayudado a ser mejor científico y profesional, y a alcanzar hitos que parecían inalcanzables. Por otro lado, no quiero dejar fuera las horas de procrastinación, que aunque quizá no tan instructivas, siempre son necesarias para sobrellevar el viaje. Querría extender también mi agradecimiento a personas cuyos caminos se separaron del mío en algún momento pero que también me acompañaron en el ámbito académico: Jesús, Emilio y Maite, gracias por nuestro tiempo juntos.

Quería agradecer también al Sen. Prof. Dr.-Ing. Hermann Ney y a todo su equipo en el RWTH por acogerme y darme la oportunidad de vivir experiencias nuevas y seguir creciendo fuera de mi zona de confort.

En el ámbito personal querría dar las gracias por todos esos momentos compartidos con mis amigos en la universidad, especialmente a Elías, Cristian, José Alemany, Guillem y José Francisco. Sin vosotros esto habría sido totalmente diferente, gracias.

Gracias también para aquellos que, aún sin entender muy bien todo lo que implicaba esto del doctorado, me han dado el apoyo incondicional en todas las ocasiones y todos los momentos de estos últimos años: mis padres Vicente y Amparo, mi hermana Blanca y Pedro, mis tíos Andrés y Javi y en especial para mi abuela Amparo, un ejemplo de resiliencia como no he visto en toda mi vida. Extiendo también mi gratitud a Domingo y Carmen, y a todos los miembros que incorporé a mi familia hace algunos años. Quiero también hacer una mención especial a Alfredo por ayudarme a darle otra perspectiva a la vida y ser un ejemplo a seguir en muchas ocasiones. Por último, gracias también a Neo y a Voga, por estar siempre dispuestos a escuchar incondicionalmente y simplemente estar ahí :).

Finalmente, no se han inventado las palabras para reflejar el apoyo que he tenido de la persona que ha compartido conmigo todos y cada uno de los momentos (buenos y malos) de este viaje, se ha sacrificado tanto o más que yo mismo, dándolo todo sin condiciones. Aïda, tú mejor que nadie sabes lo que ha sido esto y lo que ha significado para nosotros, gracias por estar SIEMPRE ahí.

Y gracias a ti, que estás leyendo esto, por dedicar tu valioso tiempo a leer el registro escrito de este periplo que supone el camino de la tesis doctoral, con todos los altibajos, los momentos maravillosos y no tan maravillosos, que al final se concretan en poner una diminuta piedra en la inconmensurable montaña del conocimiento.

Gracias.

Abstract

Over the last decade, the media have experienced a revolution, turning away from the conventional TV in favor of on-demand platforms. In addition, this media revolution not only changed the way entertainment is conceived but also how learning is conducted. Indeed, on-demand educational platforms have also proliferated and are now providing educational resources on diverse topics. These new ways to distribute content have come along with requirements to improve accessibility, particularly related to hearing difficulties and language barriers.

Here is the opportunity for automatic speech recognition (ASR) to comply with these requirements by providing high-quality automatic captioning. Automatic captioning provides a sound basis for diminishing the accessibility gap, especially for live or *streaming* content. To this end, *streaming* ASR must work under strict real-time conditions, providing captions as fast as possible, and working with limited context. However, this limited context usually leads to a quality degradation as compared to the pre-recorded or *offline* content.

This thesis is aimed at developing low-latency streaming ASR with a quality similar to offline ASR. More precisely, it describes the path followed from an initial hybrid offline system to an efficient streaming-adapted system. The first step is to perform a single recognition pass using a state-of-the-art neural network-based language model. In conventional multi-pass systems, this model is often deferred to the second or later pass due to its computational complexity. As with the language model, the neural-based acoustic model is also properly adapted to work with limited context. The adaptation and integration of these models is thoroughly described and assessed using fully-fledged streaming systems on well-known academic and challenging real-world benchmarks. In brief, it is shown that the proposed adaptation of the language and acoustic models allows the streaming-adapted system to reach the accuracy of the initial offline system with low latency.

Resumen

Durante la última década, los medios de comunicación han experimentado una revolución, alejándose de la televisión convencional hacia las plataformas de contenido bajo demanda. Además, esta revolución no ha cambiado solamente la manera en la que nos entretenemos, si no también la manera en la que aprendemos. En este sentido, las plataformas de contenido educativo bajo demanda también han proliferado para proporcionar recursos educativos de diversos tipos. Estas nuevas vías de distribución de contenido han llegado con nuevos requisitos para mejorar la accesibilidad, en particular las relacionadas con las dificultades de audición y las barreras lingüísticas.

Aquí radica la oportunidad para el reconocimiento automático del habla (RAH) para cumplir estos requisitos, proporcionando subtitulado automático de alta calidad. Este subtitulado proporciona una base sólida para reducir esta brecha de accesibilidad, especialmente para contenido en directo o *streaming*. Estos sistemas de *streaming* deben trabajar bajo estrictas condiciones de tiempo real, proporcionando la subtitulación tan rápido como sea posible, trabajando con un contexto limitado. Sin embargo, esta limitación puede conllevar una degradación de la calidad cuando se compara con los sistemas para contenido en diferido u *offline*.

Esta tesis propone un sistema de RAH en *streaming* con baja latencia, con una calidad similar a un sistema *offline*. Concretamente, este trabajo describe el camino seguido desde el sistema offline híbrido inicial hasta el eficiente sistema final de reconocimiento en *streaming*. El primer paso es la adaptación del sistema para efectuar una sola iteración de reconocimiento haciendo uso de modelos de lenguaje estado del arte basados en redes neuronales. En los sistemas basados en múltiples iteraciones estos modelos son relegados a una segunda (o posterior) iteración por su gran coste computacional. Tras adaptar el modelo de lenguaje, el modelo acústico basado en redes neuronales también tiene que adaptarse para trabajar con un contexto limitado. La integración y la adaptación de estos modelos es ampliamente descrita en esta tesis, evaluando el sistema RAH resultante, completamente adaptado para *streaming*, en conjuntos de datos académicos extensamente utilizados y desafiantes tareas basadas en contenidos audiovisuales reales. Como resultado, el sistema proporciona bajas tasas de error con un reducido tiempo de respuesta, comparables al sistema offline.

Resum

Durant l'última dècada, els mitjans de comunicació han experimentat una revolució, allunyant-se de la televisió convencional cap a les plataformes de contingut sota demanda. A més a més, aquesta revolució no ha canviat només la manera en la que ens entretenim, si no també la manera en la que aprenem. En aquest sentit, les plataformes de contingut educatiu sota demanda també han proliferat per a proporcionar recursos educatius de diversos tipus. Aquestes noves vies de distribució de contingut han arribat amb nous requisits per a millorar l'accessibilitat, en particular les relacionades amb les dificultats d'audició i les barreres lingüístiques.

Aquí radica l'oportunitat per al reconeixement automàtic de la parla (RAH) per a complir aquests requisits, proporcionant subtitulat automàtic d'alta qualitat. Aquest subtitulat proporciona una base sòlida per a reduir aquesta bretxa d'accessibilitat, especialment per a contingut en directe o streaming. Aquests sistemes han de treballar sota estrictes condicions de temps real, proporcionant la subtitulació tan ràpid com sigui possible, treballant en un context limitat. Aquesta limitació, però, pot comportar una degradació de la qualitat quan es compara amb els sistemes per a contingut en diferit o offline.

Aquesta tesi proposa un sistema de RAH en streaming amb baixa latència, amb una qualitat similar a un sistema offline. Concretament, aquest treball descriu el camí seguit des del sistema offline híbrid inicial fins l'eficient sistema final de reconeixement en streaming. El primer pas és l'adaptació del sistema per a efectuar una sola iteració de reconeixement fent servir els models de llenguatge de l'estat de l'art basat en xarxes neuronals. En els sistemes basats en múltiples iteracions aquests models son relegades a una segona (o posterior) iteració pel seu gran cost computacional. Un cop el model de llenguatge s'ha adaptat, el model acústic basat en xarxes neuronals també s'ha d'adaptar per a treballar amb un context limitat. La integració i l'adaptació d'aquests models és àmpliament descrita en aquesta tesi, avaluant el sistema RAH resultant, completament adaptat per streaming, en conjunts de dades acadèmiques àmpliament utilitzades i desafiants tasques basades en continguts audiovisuals reals. Com a resultat, el sistema proporciona baixes taxes d'error amb un reduït temps de resposta, comparables al sistema offline.

Contents

Agradecimientos **iii**

Abstract **v**

Resumen **vii**

Resum **ix**

Contents **xi**

1 Introduction **1**
1 Motivation . . . 2
2 Scientific goals . . . 3
3 Preliminaries . . . 5
3.1 Automatic Speech Recognition . . . 5
3.2 Feature extraction . . . 7
3.3 Acoustic Model . . . 7
3.4 Language Model . . . 10
3.5 Search . . . 11
3.6 Streaming Automatic Speech Recognition . . . 13
3.7 Benchmarking . . . 16
3.8 Evaluation of ASR systems . . . 17
4 Framework . . . 18
5 List of publications . . . 20
5.1 Paper 1 . . . 20
5.2 Paper 2 . . . 21
5.3 Paper 3 . . . 21
5.4 Paper 4 . . . 22

5.5 Paper 5 . . . 23
6 References . . . 24

2 Selected Papers 29
1 MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge . . . 31
1.1 Introduction . . . 34
1.2 RTVE database . . . 34
1.3 Closed-condition system . . . 35
1.4 Open-condition system . . . 41
1.5 Conclusions . . . 42
References . . . 43
2 Real-time one-pass decoder for speech recognition using LSTM language models . . . 45
2.1 Introduction . . . 47
2.2 One-pass decoder architecture . . . 48
2.3 Experiments . . . 52
2.4 Conclusions and future work . . . 57
References . . . 57
3 LSTM-based one-pass decoder for low latency streaming . . . 61
3.1 Introduction . . . 63
3.2 Streaming decoder . . . 65
3.3 Experiments . . . 66
3.4 Conclusions and future work . . . 70
References . . . 71
4 Live Streaming Speech Recognition Using Deep Bidirectional LSTM Acoustic Models and Interpolated Language Models . . . 73
4.1 Introduction . . . 75
4.2 Efficient One-pass Decoding using Interpolated Neural LMs . . . 84
4.3 Experiments . . . 87
4.4 Conclusion and future work . . . 100
References . . . 101
5 MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge . . . 105
5.1 Introduction . . . 108
5.2 Challenge description and databases . . . 109
5.3 MLLP-VRAIN Systems . . . 109
5.4 Conclusions . . . 117
References . . . 118

3 General discussion of the results 121
References . . . 130

4 Conclusions and future work 131
References . . . 134

List of Figures

1.1 From the acoustic signal in the time domain to the resulting $[w_1^N]^*$, the ASR pipeline goes through the feature extraction process and searches for the best sequence of words combining the input from the LM, the AM, and the pronunciation dictionary. 6

2.1 Evaluation of the impact of the LMHR parameter in terms of WER [%] for LibriSpeech and TED-LIUM. 54
2.2 Comparison of the impact of different values for the *LM-histogram-pruning* (LMHP) parameter in terms of WER and RTF for LibriSpeech and TED-LIUM. 55
2.3 Comparison of the one-pass HCS decoder and the two-pass HCS and WFST decoders in terms of WER and RTF for LibriSpeech (left) and TED-LIUM (right). 56
2.4 Frame sequence at the top and just below a sliding window of $w = 4$ frames at all steps embracing frame t, $\vec{x}_t$. 79
2.5 Computing the acoustic scores for $b = 3$ consecutive frames starting at t, $\vec{x}_t^{t+b-1}$, within a sliding window of size $w = 4$ during two consecutive b-step batches, B_{j-1} and B_j. 80
2.6 WER vs. window size in seconds for all tasks. 91
2.7 WER vs. latency in seconds with or without AMLA enabled for each task. 92
2.8 FSN, DTN and WMA normalization (with different α values) schemes evaluated on WER for segment-based tasks. 94
2.9 FSN, DTN and WMA normalization (with different α values) schemes evaluated on WER for video-based tasks. 94
2.10 WER (left y-axis) and PPL (right y-axis) as a function of TLM history limitation and different LMHR values for segment-based tasks. Solid curves represent WER, while the dashed curve is PPL. 96

2.11 WER (left y-axis) and PPL (right y-axis) as a function of TLM history limitation and different LMHR values for video-based tasks. Solid curves represent WER, while the dashed curve is PPL. 96
2.12 WER vs. latency in seconds varying LMHP values for segment-based tasks. 97
2.13 WER vs. latency in seconds varying LMHP values for video-based tasks. . 98
2.14 WER vs. latency in seconds considering different interpolation schemes with TLM for each segmented task. 99
2.15 WER vs. latency in seconds considering different interpolation schemes with TLM for each video task. 99
2.16 WER as a function of context window size (in seconds) for the streaming setup, computed over the *dev1-dev* set. 115
2.17 WER versus mean empirical latency (in seconds) on *dev1-dev*, measured with different prunning parameters, and considering only interpolation schemes that include TLM. 116

3.1 WER scores over project month for poliMedia Spanish/English (left) and VideoLectures.Net Slovenian/English (right) transcriptions. 125

List of Tables

2.1 Number of raw, aligned raw, aligned speech, and filtered speech hours as a result of applying the speech data filtering pipeline to the whole RTVE database. . . . 36
2.2 Corpus statistics of the text data used for LM training. . . . 37
2.3 Perplexities of the different LM components. . . . 39
2.4 Comparison of the CD-FFDNN-HMM and BLSTM-HMM acoustic models using the general n-gram language model. Results in WER % and relative WER % improvement. . . . 39
2.5 Comparison of different language model combinations using the BLSTM-HMM acoustic model in terms of perplexity, WER % and relative WER % improvement. . . . 40
2.6 Comparison of different audio segmentation/VAD techniques using the BLSTM-HMM acoustic model and the combination of the RNN LM + adapted n-gram LM. Results in WER % and relative WER % improvement and the ratio of dropped audio. . . . 41
2.7 Speed analysis in terms of RTF an its effect on the WER% over the *dev2* set, either with the submitted system and removing the pre-precognition step, using LIUM VAD instead. . . . 41
2.8 Statistics of the corpora. . . . 52
2.9 Comparison of WER, relative improvement of WER w.r.t the baseline, and perplexity results using different LM models on LibriSpeech and TED-LIUM test partitions. . . . 54
2.10 Comparison of WER, RTF and relative increase of RTF w.r.t to LMHP=100 on LibriSpeech and TED-LIUM test sets. . . . 55
2.11 Comparison of WER for similar RTFs between one-pass HCS decoder and the two-pass HCS and WFST decoders on LibriSpeech and TED-LIUM test partitions. . . . 57
2.12 Statistics of the corpora. . . . 67

2.13 Perplexity results on development partitions. 67
2.14 Impact of normalization context on WER, on LibriSpeech and TED-LIUM. 68
2.15 Impact of lookahead context on LibriSpeech and TED-LIUM on WER. . . 69
2.16 Impact of lookahead context on LibriSpeech and TED-LIUM on the latency ($n_{\text{l}} = n_{\text{lookahead}}$, $n_{\text{n}} = n_{\text{norm}}$). 70
2.17 WER results on test sets for LibriSpeech and TED-LIUM. 70
2.18 Basic statistics of dev and tests sets in the evaluation tasks: Duration in hours, number of samples (segments or videos), average duration of samples in seconds plus-minus standard deviation ($d_\mu \pm \sigma$), and running words (RW) in thousands (K). 87
2.19 Statistics of Spanish text resources used for language modeling. S=Sentences, RW=Running words, V=Vocabulary. Units are in thousands (K). 89
2.20 Perplexity (PPL) and weight (W) figures on development sets, considering single models and two-way and three-way interpolation of n-gram (N), LSTM-RNN LM (L) and Transformer LM (T). Interpolation weights were optimized by minimizing PPLs of the interpolated models. 90
2.21 WER and latency in seconds on the test sets using the optimized streaming systems for all the evaluation tasks compared to previous works. 100
2.22 Transcribed Spanish speech resources for AM training. 110
2.23 Statistics of Spanish text resources for LM training. S=Sentences, RW=Running words, V=Vocabulary. Units are in thousands (K). 112
2.24 Basic statistics of development and tests sets of RTVE databases, including our internal *dev1-dev* set: total duration (in hours), number of files, average duration of samples in seconds plus-minus standard deviation ($d_\mu \pm \sigma$), and running words (RW) in thousands (K). 113
2.25 Perplexity (PPL) and interpolation weights, computed over the *dev1-dev* set, of all possible linear combinations of n-gram (ng), LSTM (ls) and Transformer (tf) LMs. 113
2.26 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. 117

3.1 WER of the participants of the closed-condition track from IberSpeech 2018 on *test-2018*. 123
3.2 WER scores provided by X5Gon ASR systems and Google Cloud Speech-To-Text. 124
3.3 WER scores provided by offline and streaming-adapted M40 ASR systems. . 126
3.4 WER scores provided by MLLP and Google ASR systems on different content (es=Spanish, ca=Catalan, and en=English) and relative improvement. 126
3.5 WER, toolkit, decoder type and data in hours of the participant systems on *test-2020*, including the closed-conditioned version from our MLLP-VRAIN submission (str=streaming-ready). 128

Chapter 1

Introduction

1 Motivation

Natural Language Processing (NLP) technologies are focused on improving human interpersonal communications through learning and understanding of the mechanisms of natural language. This field can be framed at the junction where linguistics, computer science, and artificial intelligence intersect. One of its most popular branches is Automatic Speech Recognition (ASR), a sub-field devoted to converting the raw audio signal into a sequence of spoken words.

ASR has been widely studied over the last 60 years, starting from the 50s at Bell laboratories with isolated spoken digits recognition systems through the 90s with general-purpose continuous-speech systems (O'Shaughnessy 2008). Over the 2000s, ASR was well established in the industry of Interactive Voice Response (IVR) tasks, such as call centers or help-desk support with limited task-oriented capabilities. At the same time, research and academia lost interest in the field of speech recognition. At the beginning of the 10s, in 2011, Google released Google Voice Search, a feature that allows users to interact directly with their smartphones using their voice for querying its search engine without explicit limitations for domain or vocabulary. This feature was the first step, after many years, approaching general-purpose ASR systems to the end-user. This same year, Apple released Siri, its voice assistant, providing a game-changing experience of voice-based human-computer interaction. After Siri, others came: Amazon's Alexa, Microsoft's Cortana, and Google's Assistant, to name a few. These latest developments and the deep learning revolution fueled ASR research, which is now considered an interesting, exciting, and valuable research topic.

Along with the revitalized academia attention, there was a growing interest from the media industry during the last years due to the explosion of new digital content. This content ranges from the Internet media to digital television. Environments such as video streaming services or massive online open courses (MOOCs) are contexts where ASR can enrich the experience, for example, breaking barriers to access to the content due to hearing difficulties or enabling more efficient language processing and understanding (Yang 2020). In this regard, ASR systems purpose is two-fold: they can provide good enough automatic transcriptions to be used directly when content is provided (i.e.: lectures or conferences) and can reduce professional captioners' workload when supervision is required (i.e.: TV broadcasting or official records). These two scenarios should work both for live and recorded media. Furthermore, regarding the professional supervision, the workload reduction when using ASR systems is directly related to the system's performance (Wald et al. 2007), being the speed a critical aspect also when dealing with live broadcasts. This latter kind of content involves live captioning, the most demanding and stressful task for professionals, as it requires noticing errors, correcting them and continuing listening in a time-bounded manner (Wald et al. 2007). This kind of live captioning systems are widely-known as *streaming* systems, while the ASR systems working with recorded media are referred to

as *offline* systems. Certainly, providing state-of-the-art streaming ASR systems, with low error rates and low latency, performing live captioning of unconstrained content (such as debates or morning shows) is still a challenging problem. Indeed, this thesis is devoted to provide these high-quality streaming ASR systems that can work on live environments. To do this, the main focus is to ensure that the quality degradation is negligible with respect to the offline setup, allowing the system to provide proper and useful live transcriptions with low-latency that can be used directly (i.e.: live lectures and conferences) or with minor supervision (i.e.: live TV broadcasting).

2 Scientific goals

Conventional ASR systems work with recorded audio, where the entire content is available. This working regime allows the system to iterate over the content several times (multi-pass approach). However, this iterative approach involves several restrictions in terms of speed and quality, as the system should process the whole audio to perform the next iteration, and errors can be propagated from iteration to iteration.

To avoid this, the first challenge arising in this context is building a *one-pass* system that integrates state-of-the-art neural network models, that have largely demonstrated to be the best models for ASR (Mohamed, G. Dahl, Hinton, et al. 2009; Mohamed, George E. Dahl, and Hinton 2012; Seide, Li, and Yu 2011; Deng et al. 2013). In particular, neural language models (LMs) are commonly deferred to the last steps of the multi-pass approach (Si et al. 2013; Xu et al. 2018; Ogawa et al. 2018) due to their computational complexity. Including them appropriately in the first step will improve accuracy and reduce the number of errors. It is also essential to perform this integration efficiently to minimize the total recognition time while keeping accuracy as good as possible. On the other hand, neural-based acoustic models (AMs) must also be integrated and adapted to the streaming recognition process. Waiting for the whole utterance to be spoken is not an option in the streaming setting, as the user expects the feedback from the system as soon as possible. Therefore, partial outputs must be delivered after processing a short part of the incoming acoustic signal. Thus, being limited to performing only one-pass, the system must do its best to get the most out of the input, using the best models. In addition, the speed and efficiency of the system are critical as well, as time-constrained responses are expected. This requirement involves that the AM, which had the entire sequence available to provide acoustic scores in the offline setup, is now limited to a part of the utterance. The amount of context we want to deliver to the AM will condition the latency of the system, meaning that more context involves gathering more acoustic information, delaying the output of the system. Again, following the neural-based trend for the AM, the input pipeline and the whole sequence models need to be adapted to this limited setup. This adaptation leads to a second challenge, where the input pipeline

and the neural-based AM have to be adjusted and thoroughly evaluated under streaming conditions.

As mentioned above, this work is focused on a real requirement from the industry: high-quality and low-latency live captioning. However, the dataset landscape is not well-defined for this task, showing significant shortcomings in reflecting the quality of the system when deployed in real-world environments. Under these real-world conditions, the performance and accuracy of the system should be thoroughly tested against long non-restricted audio signals such as TV shows. This kind of task requires using long-span evaluation content that encourages the extra effort put on efficiency and improving the quality against adverse conditions. This task-oriented line of thought motivated the third and last scientific goal of this thesis.

To summarize, during this thesis, the following research questions will be addressed:

- **Is it possible to perform recognition in one-pass and real-time with a quality similar to that of a multi-pass system?**
- **Is it possible to perform recognition in streaming with low latency with a quality similar to that of an offline system?**
- **How this streaming ASR system can be integrated into production environments? (i.e., OERs or live TV broadcasts)**

These questions are materialized in the following scientific goals that will pursue the final production-ready streaming ASR system:

- **Provide a one-pass ASR system leveraging the neural LM in real-time**: This goal is the intermediate step to obtain a fully-operative streaming system, performing just one iteration of recognition using neural-based LM such as Long Short-Term Memory or Transformer architecture. Secondly, the decoding should not be only performed in one-pass but quick enough to work under real-time constraints.
- **Adapt the one-pass neural-based system to perform low-latency streaming recognition**: This goal aims to adapt the input pipeline (from audio to the recognizer, through the AM) to the streaming setup, adjusting the normalization and the computation of the acoustic features.
- **Evaluate the streaming ASR system in production environments**: Commonly and well-known ASR benchmarks were conceived to evaluate offline ASR pipelines, not reflecting well tasks or environments where a streaming system will be deployed. For this reason, in this thesis, we foster the evaluation of our proposals with real-world, unconstrained challenging tasks to explore the limits of our approach.

These challenges motivated the scientific interest in streaming ASR. For clarity and understanding, the following section will introduce the required key concepts that frame the scope of the scientific challenges in streaming ASR that were tackled during this thesis.

3 Preliminaries

3.1 Automatic Speech Recognition

The purpose of ASR is to convert spoken language into text. Unlike conventional classification or regression problems, the output of this process is structured and grows exponentially, being sequential. Therefore, this process is modeled as finding the best transcription, the word (string) sequence $w_1^N = \{w_1, ..., w_N\}$, giving the utterance, the sequence of acoustic observations $\mathbf{x}_1^T = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$, real-valued vectors coming from a feature extraction process over the speech signal. This process can be interpreted from a statistical point of view as getting the word sequence ($[w_1^N]^*$) that maximizes the posterior probability given the acoustic information $\mathbf{x}_1^T$, as follows:

$$[w_1^N]^* = \arg\max_{w_1^N} p(w_1^N \mid \mathbf{x}_1^T) \tag{1.1}$$

To approach this problem, we can rewrite Eq. 1.1 using the Bayes' theorem (Bayes 1763), obtaining the following:

$$[w_1^N]^* = \arg\max_{w_1^N} \frac{p(\mathbf{x}_1^T, w_1^N)}{p(\mathbf{x}_1^T)} \tag{1.2}$$

$$= \arg\max_{w_1^N} p(\mathbf{x}_1^T, w_1^N) \tag{1.3}$$

$$= \arg\max_{w_1^N} p(w_1^N)\, p(\mathbf{x}_1^T | w_1^N) \tag{1.4}$$

First, the term $p(\mathbf{x}_1^T)$ remains constant when searching for the best word sequence, meaning that the maximization can drop this part. After this, we nicely decompose the joint probability into the prior of the sequence of words ($p(w_1^N)$), provided by what is known as the *language model* (LM), and the posterior of the sequence of acoustic observations given a word sequence ($p(\mathbf{x}_1^T | w_1^N)$), provided by what is known as the *acoustic model* (AM). There is an additional model, usually omitted and seen as part of the AM, known as the *lexicon model*, that could be seen as a pronunciation dictionary translating words into their phonetic

representation. This modeling decomposition is commonly known as the generative approach, the baseline model that guided the development of this thesis. On the other hand, during the last years, there has been much interest in the discriminative approach (Chan et al. 2016), which tackles Eq. 1.1 directly, mainly to leverage the powerful neural networks models coming from the deep learning era, the so called Sequence-to-sequence models (seq2seq) (Chan et al. 2016).

The generative approach to modeling ends up with a search problem: finding the best word sequence according to Eq. 1.4, combining the knowledge from both the AM and the LM. This step is commonly known as *decoding*, a term coming from the noisy channel theory (Claude Elwood Shannon 2001). A naive approach could use a brute force algorithm enumerating all possible sentences that fit with the acoustic signal, evaluating them and returning one with the highest score. This approach becomes computationally infeasible for systems with the vocabularies considered nowadays (around 100K words or more). Therefore, efficient algorithmic techniques and data structures are required to allow us to benefit from *large-vocabulary continuous speech recognition* (LVCSR) systems.

To summarize, Figure 1.1 shows the main components of the ASR pipeline. They are described in the following sections: the feature extraction process, the AM and the pronunciation dictionary, the LM, and the decoder that implements the search algorithm.

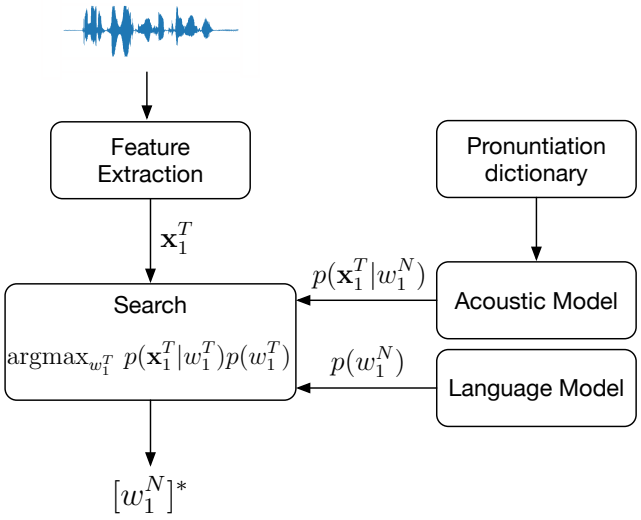

Figure 1.1: From the acoustic signal in the time domain to the resulting $[w_1^N]^*$, the ASR pipeline goes through the feature extraction process and searches for the best sequence of words combining the input from the LM, the AM, and the pronunciation dictionary.

3.2 Feature extraction

The feature extraction process, designed based on our knowledge on the human auditory system, aims to obtain a sequence of acoustic observations, real-valued vectors, capturing relevant information from the raw audio signal. This process involves various signal processing algorithms to emphasize, filter, and transform the signal, computed locally using typically an overlapping analysis window of 25 ms with a 10ms stride over the audio signal. The standard pipeline performing this transformation is well established nowadays, focusing on providing a compact representation of the speech signal that can help the underlying nature of the posterior AM. For example, for the models commonly used for the acoustic modeling, the Mel-frequency cepstral coefficients (MFCCs) are the most common features. These features are extracted applying the Mel scale in the frequency domain, imitating the human filtering (Davis and Mermelstein 1980). Alternatively, Gammatone features have demonstrated their effectiveness in noisy environments (Schlüter et al. 2007). More recently, neural network acoustic models leverage the direct use of the logarithmic Mel scale filter banks that ease data augmentation techniques (Park et al. 2019). In all these transformations, once the features are extracted from the raw audio, we end up with an appropriate input vector representation $\mathbf{x}_1^T$ for the AM module.

3.3 Acoustic Model

The acoustic model (AM) interacts directly with the acoustic observations, extracting knowledge from the transformed audio signal. This interaction involves dealing with the variability coming from the audio representation, regarding variations intra/inter speakers, such as context (i.e., phonemes being modified by its surrounded sounds) or style (i.e., variations in speaking rate or spontaneous speech), and different environmental and technical conditions (i.e., studio vs. on-device recordings). Due to this, the modeling task in this part of the ASR system should take all these sources of variations into account.

Unlike the conventional classification problem, the temporal dimension must be considered to model human speech properly. That is, the variability of the speech (i.e., short/long pronunciations of the same phonemes depending on the speaker) has to be extracted and modeled correctly to obtain accurate results. From Eq 1.4, we see that the AM plays its role as the second factor of the search equation: $p(\mathbf{x}_1^T|w_1^N)$. Under the statistical framework, the AM module has to provide a statistical model for the sequence of acoustic observations $\mathbf{x}_1^T$, conditioned on the sequence of discrete words w_1^N.

Providing the probability of the sequence of continuous vectors conditioned on the discrete word sequence is not straightforward. During the last 50 years, Hidden Markov Models (HMMs) (Baum and Eagon 1967; Baker 1990) have been widely applied and refined, modeling this conditional probability. HMMs naturally inte-

grate the sequence modeling with well-known and efficient algorithms to train and decode, along with continuous-density probability functions modeling real-valued vectors.

HMMs are the underlying acoustic models used in almost all the ASR systems that follow the generative framework. They come from the Markov chain theoretical framework, where a stochastic process is modeled as a finite state machine, with transitions and states. In addition, the probability of the random variable at a given time depends only on the value at the previous time step. Unlike the baseline Markov chain model, where the sequence of states followed by the stochastic process is observable and deterministic, in the HMM, randomness is introduced by associating an emission probability (density) function to each state. With this extension, HMMs describe signals in real-world applications, such as speech processing tasks, where the states of the system are not directly observed. For example, the most likely sequence of spoken phonemes in speech recognition. These Markov states provide the local stationary features combined globally to approximate the non-stationary nature of human speech production.

In the HMM framework, the hidden state sequence provides indirect information about the output sequence and helps us model the output probability distribution. These hidden states reflect the set of different setups of the abstract finite state machine that were visited during the generation of $\mathbf{x}_1^T$. To include this hidden sequence, the conditional probability over the set of all possible state sequences q_1^T can be marginalized.

$$p(\mathbf{x}_1^T|w_1^N) = \sum_{q_1^T} p(\mathbf{x}_1^T, q_1^T \mid w_1^N) \tag{1.5}$$

$$= \sum_{q_1^T} \prod_{t=1}^{T} p(\mathbf{x}_t, q_t \mid \mathbf{x}_1^{t-1}, q_1^{t-1}, w_1^N) \tag{1.6}$$

$$= \sum_{q_1^T} \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{x}_1^{t-1}, q_1^t, w_1^N) p(q_t \mid \mathbf{x}_1^{t-1}, q_1^{t-1}, w_1^N) \tag{1.7}$$

This formulation accounts for all the possible dependencies in the model regarding the sequence of acoustic observations, hidden states, and words. Typically, to reduce the complexity of the model, the following assumptions, commonly known as first-order Markov assumptions, are made:

- The probability of $\mathbf{x_t}$ depends only on the current state q_t.
- The transition probability depends only on the previous state q_{t-1}.

After these approximations, for each sequence w_1^N, an implicitly conditioned HMM, with parameters $\Theta_{w_1^N}$, is built to represent the conditioned probability $p(\mathbf{x}_1^T \mid w_1^N)$, denoted as $p_{\Theta_{w_1^N}}(\mathbf{x}_1^T)$. Hence, any sequence w_1^N can be transformed into a concatenation of these HMM states. Then, the formulation ends up as:

$$p_{\Theta_{w_1^N}}(\mathbf{x}_1^T) = \sum_{q_1^T} \prod_{t=1}^{T} p(\mathbf{x}_t \mid q_t) p(q_t \mid q_{t-1}) \tag{1.8}$$

In practice, the sum over all state sequences is approximated by the max function, known as the Viterbi approximation (Viterbi 1967), as follows:

$$p_{\Theta_{w_1^N}}(\mathbf{x}_1^T) \approx \max_{q_1^T} \prod_{t=1}^{T} p(\mathbf{x}_t \mid q_t) p(q_t \mid q_{t-1}) \tag{1.9}$$

In both cases, $p(\mathbf{x}_t \mid q_t)$ models the emission probability and $p(q_t \mid q_{t-1})$ the transition probability. Regarding the emission probability, this is commonly modeled by Gaussian Mixture Models (GMM), leading to the so-called GMM-based HMM (GMM-HMM). In terms of the transition probability, this is directly modeled as a lookup table. The resulting transition-emission process produces a sequence that is not directly observed in the original data, in the form of the alignment between the acoustic and word sequences, through a minimal pronunciation unit at phoneme level. We have one HMM for each phoneme, concatenating them to form word sequences by marginalization as in Eq. 1.5.

In practice, phonemes are replaced by context-dependent phonemes (or triphonemes) considering not only an isolated phoneme but also its immediate neighboring phonemes. For example, the word "dog", with the phonemes "/d/ /a/ /g/", is transformed into the triphonemes "♯-d-a d-a-g a-g-♯". This modeling accounts for different co-articulation of phonemes, providing better recognition results. However, this makes the number of phonemes grow drastically. For example, consider the 44 English phonemes, which will produce 44^3 possible triphoneme combinations, up to more than 85 000 symbols. Moreover, many of these symbols are never or rarely seen in the data, posing training difficulties. It is important to note that the number of potentially different HMMs could be intractable using this approach, so we resort to defining a limited set of shared HMM states, where each state represents a sub-word unit. Hence, any sequence w_1^N can be transformed into a concatenation of these shared HMM states. The selection of these shared HMM states is usually performed with Classification and Regression Trees (CART) (Young, Odell, and Woodland 1994). This technique reduces the number of distinct triphonemes, for example, in the current state-of-the-art systems, to 10-20K, thus reducing training complexity.

More recently, neural networks have been introduced to cope with the acoustic modeling part after showing that Feed-Forward Networks (FFNs) can perform a better job at estimating $p(\mathbf{x}_t \mid q_t)$ than GMMs (Bourlard and Wellekens 1990). To this purpose, the emission probability is reformulated in a discriminative fashion as follows:

$$p(\mathbf{x}_t \mid q_t) = \frac{p(q_t \mid \mathbf{x}_t)p(\mathbf{x}_t)}{p(q_t)}, \tag{1.10}$$

where $p(q_t)$ is estimated as the number of times this state is visited, and the posterior probability $p(q_t \mid \mathbf{x}_t)$ by a (deep) neural network (DNN). The term $p(\mathbf{x}_t)$ is not considered, as this remains constant during decoding. Over the last years, deeper FFNs (Hinton et al. 2012), Convolutional Neural Networks (CNNs) (Sainath et al. 2015; Bozheniuk et al. 2020), and Recurrent Neural Networks (RNNs) (Schuster and Paliwal 1997) have been used to approach the acoustic modeling. This refinement results in the current hybrid systems, combining both HMMs and neural networks.

3.4 Language Model

Language modeling aims to estimate $p(w_1^N)$, a vital component in any ASR system. This probability is typically factorized similarly to that of the AM, that is,

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n \mid w_1^{n-1}). \tag{1.11}$$

Following this approach, the most popular technique is the n-gram model (C. E. Shannon 1948). The n-gram model is based on the Markovian assumption where only the last m words matter to predict the next word:

$$p(w_n \mid w_1^{n-1}) \approx p(w_n \mid w_{n-m}^{n-1}). \tag{1.12}$$

These probabilities can be easily estimated from normalized word counts in the training corpora. In practice, sophisticated smoothing and discount techniques are applied to account for n-grams that are not observed in training and efficiently combine evidence from different n-gram orders (S. F. Chen and Goodman 1999).

Similar to what happened in the acoustic counterpart, neural networks were introduced to the task of LM in the last years. In contrast to n-gram LMs, neural-based LMs attempt to directly estimate $p(w_n \mid w_1^{n-1})$ via continuous word vector rep-

resentations, known as word embeddings (Bengio et al. 2003). However, there are fundamental differences regarding how the word history w_1^{n-1} is internally represented in neural-based LMs. For example, FFNs were successfully used in the aforementioned work (Bengio et al. 2003) using a window over the input that captures the previous word embeddings.

In Mikolov et al. 2010, RNN-based LMs demonstrated better performance than FFNs, leveraging the inner memory cells to learn sequences and patterns with arbitrary lengths, far beyond than the widely used count-models (3 or 4-grams). They were employed as LMs in ASR to compress the word history into a continuous vector representation. RNN-based LMs are usually conceived to process the sequence of word embeddings in a forward manner according to Eq. 1.11. Then, the initial RNN layer is followed by a FFN with a softmax activation function that defines a probability distribution over the vocabulary. Long Short-Term Memory (LSTM) networks appeared to overcome some of the inherent drawbacks with RNN modeling (Hochreiter and Schmidhuber 1997), and they became the standard neural recurrent model in ASR (Sundermeyer, Ralf Schlüter, and Hermann Ney 2012).

More recently, the Transformer architecture (Vaswani et al. 2017), designed originally for machine translation, has revolutionized the field of language modeling with impressive results using an attention mechanism (Baevski and Auli 2019; Dai et al. 2019; Radford et al. 2019), and more precisely, in ASR (Irie et al. 2019). In addition to this attention mechanism, and in contrast to LSTM, the architecture of the Transformer model avoids the recurrence, enabling the parallelization and speeding up the computation drastically.

3.5 Search

After AM and LM modeling, the problem is reduced to find the best word sequence (hypothesis) according to Eq. 1.4. For the sake of numerical stability, computations are performed after applying the logarithm to probabilities. Therefore the formulation ends up as follows:

$$[w_1^N]^* = \arg\max_{w_1^N} \left\{ \log p(w_1^N) + \max_{q_1^T} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid q_t) + \log p(q_t \mid q_{t-1}) \right\} \quad (1.13)$$

The decoding process follows the conventional beam-search Viterbi algorithm (Viterbi 1967; Hermann Ney 1984). This search algorithm involves several heuristics to prune the search space in order to reduce the time complexity and to achieve reasonable response times. For example, a small beam size or a limited number of active hypotheses is used. Among them, the LM scale is one of the most important heuristics. It follows the formulation in 1.13 to properly combine the AM

and the LM. According to 1.13, the AM provides its scores at each time t, while the LM score is computed when a new word is hypothesize during search. To compensate for this, the LM scale is included just as a scale factor, to boost the LM impact on the global decision. This scale factor is also known as the *grammar scale factor*.

The way in which the recognition (or decoding) step is carried out depends very much on its application setting. In the conventional ASR pipeline, referred to as *offline* or *batch* recognition, the whole audio sequence is available, and then we can inspect the complete utterance many times, limited only by the time that we want to devote to perform the decoding. This multi-pass approach is the most common, performing multiple iterations over the complete sequence to obtain the best hypothesis for the output word sequence. Motivated by the fact that, on each step, the set of hypotheses is refined, reducing their number, hence reducing the search effort. This set of hypotheses for each step is directly related to one of the previously introduced components, the LM. Indeed, to consider valid word sequences, the LM is transformed into a search structure that guides the decoding step. The transformation applied and how this model is used in decoding defines the nature of the decoder, for which several approaches have been proposed and studied (Nolden 2017). Nowadays, there are two dominant approaches to cope with this purpose, those based on Weighted Finite-State Transducer (WFST) (Mohri and Riley 1999; Mohri and Riley 2001; Povey et al. 2011), and those grounded on History Conditioned Search (HCS) (Hermann Ney and Ortmanns 2000; Nolden 2017). The main differences between these two approaches are the data structures employed to represent each component of the ASR system, either if they are statically precomputed or dynamically expanded during decoding. Both methods convert the AM and LM into complex data structures to perform a time-synchronous search. To alleviate the exponential complexity of the resulting search space, several pruning techniques are applied to reduce the number of paths or hypotheses to explore.

Indeed, while this multi-pass approach can reduce the computation and memory use dividing the recognition process into several stages, this also increases the response time of the ASR system as a whole, as several iterations have to be carried out. For this reason, several authors proposed one-pass decoding algorithms based on WFST decoders composing the aforementioned structures to perform a fast and memory-efficient decoding (Hori et al. 2007). However, the use of neural LMs, which provide the best performance, along with these one-pass search algorithms, poses additional problems. These problems are related to the highly-demanding computational requirements to compute LM scores. Existing one-pass decoders using neural-based LMs have been proposed in the literature, for example, approximating the neural model with a count-based model (Arisoy et al. 2014; Singh et al. 2017), or suggesting the use of lighter models, such as Gated Recurrent Units (GRU) networks, caching strategies, and complex hybrid parallelization methods (Lee et al. 2018). However, few of them have reached the technology readiness level needed for a real production environment where real-

time enabled decoders are required, and/or they do not integrate the LSTM LM completely, considering only partial information through caching or quantization strategies.

The discrete nature of the count-based LMs (n-gram models) is exploited to create the search structure mentioned above before decoding. However, after being reinterpreted as the search model, the complexity of the LM has a strong impact on the size of the search space. The more complex and rich the LM is, the larger the search structure becomes, thus involving much more memory and time requirements. Suppose state-of-the-art neural-based LMs are considered. In that case, we are changing the nature of the model from the traditional discrete count-based models to continuous representations and involving much more computation. Thus, integrating continuous neural LMs into the discrete search space becomes a challenging task. Due to this, multi-pass decoders leverage neural-based LMs performing additional recognition iterations. In this multi-pass approach, the underlying idea is, first, to use a lightweight LM, meaning an efficient and memory-bounded model, to conveniently perform a fast first-step of recognition. This step is followed by a second step over the resulting limited set of hypotheses from the first step, with a better (more extensive and more complex) LM, for example, a neural network-based LM. There are two main approaches to obtaining the second set of hypotheses, namely, generating n-best candidates or representing them in compact form using a word graph, or *lattice*, and then using the LM to score the resulting sentences or graphs (Sundermeyer, Tüske, et al. 2014; X. Chen et al. 2017; Xu et al. 2018).

3.6 Streaming Automatic Speech Recognition

In contrast to *offline* or *batch* ASR, *streaming* ASR tries to provide a good trade-off between latency (delay of receiving the input signal and providing the output sequence) and accuracy (transcription quality). This is motivated by the fact that our proposal of streaming ASR is focused on recognizing continuous audio streams without segmentation. Indeed, providing a continuous transcription is required, even if the shown transcription is not the final hypothesis, for the sake of response time. This unsegmented recognition allows the system not having to deal with segmentation algorithms that could impact the transcription quality (Rybach et al. 2009).

Regarding the latency/accuracy trade-off, this involves two crucial aspects: the speed of the system to keep the pace when processing the input signal, and a limited view of this input signal that will define the baseline delay, as the system cannot wait until the whole audio is provided. To address the first aspect, the speed, the decoding process has to leverage the best models in the first step of recognition. Indeed, as mentioned before in the offline ASR section, the multi-pass approach increases the response time of ASR systems compared to those based on one-pass decoders. In addition, search errors that cannot be fixed could

be propagated in posterior steps. To alleviate these cascaded errors, the neural LM has to be an active part of the first recognition step. As mentioned before, several approaches coped with this issue (Arisoy et al. 2014; Singh et al. 2017; Lee et al. 2018). However, not all of them are prepared to work under real-time conditions, meaning that moving to streaming conditions is not feasible.

On the other hand, along with using the best LM (the neural LM) during the first pass to improve the quality of the recognition, the speed of the whole system is a crucial aspect. For streaming decoding, the decoder has to work under real-time constraints, meaning that the decoding process should take no longer than the duration of the speech signal to be processed (or even less). For example, consider an input stream and a system with a fixed delay of t seconds gathering enough acoustic information. Thus, the decoder has to process these t seconds of audio necessarily in less than t seconds; otherwise, the processing time of the decoder will introduce more and more latency continuously, adding up to the fixed latency of t seconds, reducing the time performance drastically. Alternatively, if the decoder can work steadily under real-time conditions, the total latency will equal the fixed latency t plus a fraction of t at maximum. In this work other factors that will condition the global latency are not considered, such as the network or communication delays, as they are very heterogeneous. Clearly, in nowadays cloud-based environments, these factors have to be taken into account, but they fall outside the scope of this thesis.

Another critical aspect to consider in the streaming setting, in addition to the performance of the decoder itself, is how to deal with the partial acoustic input. As mentioned before, in the streaming setup, the user expects the output from the system as soon as possible (i.e., live captioning), involving that the system has to provide partial hypotheses from the input stream as it comes. To this purpose, it is required to include the aforementioned fixed latency, t, to gather acoustic information to perform the feature extraction and the acoustic processing. Traditionally, GMM did not require any additional context to provide the acoustic scores, processing the input as it comes without any further modification concerning the *offline* processing. However, with the neural-based models, the acoustic context became more and more relevant (Hinton et al. 2012). That is especially true with the recurrent models, i.e., bidirectional LSTM, where the model itself stores contextual information from both past and future context in their internal states (A. Graves and Schmidhuber 2005).

To alleviate the lack of full context on the streaming setup, there were several proposals in the literature based on different windowed/chunking approaches to adapt the bidirectional LSTMs to this environment. In (Mohamed, Seide, et al. 2015), it is compared the traditional windowed FFN mechanism with the same BLSTM approach, obtaining the scores for the center frames grouping consecutive frames, sharing the same underlying RNN state, outperforming the DNN also with truncated inputs. After that, in (Zeyer, R. Schlüter, and H. Ney 2016) this mechanism was extended to compute not only the center frames but all the frames

from the window, along with several scoring averaging methods to compute the scores from overlapping windows. The authors reported a similar recognition performance to the traditional offline setup. This approach, unlike (Mohamed, Seide, et al. 2015), does not require any specific training scheme, allowing the adaptation of the offline models to the streaming setup. In addition, this latter approach allowed the system to process the input with steps larger than one frame, speeding up the acoustic model computations. In parallel, in (K. Chen and Huo 2016) it is proposed another approach based on splitting the input sequence into chunks with appended contextual observations, in what is called context-sensitive chunks (CSCs). Differently from (Zeyer, R. Schlüter, and H. Ney 2016), where training and recognition are inconsistent, here, the consistency is enforced to speed up the training procedure as well, as the contextual frames are not considered when computing the loss and when the backpropagation step is performed. Another similar approach is presented in (Zhang et al. 2016), in this case alleviating the truncated fixed context from CSC replacing it by carrying the whole previous history with the hidden state, from the past context, during the computation of the next frames, copying the states directly from previous chunks. The future context remains as contextual information that generates no error signal. Again, this approach involves a consistent training-recognition scheme, and authors claimed that this proposal leads to faster training/recognition and, often, better accuracy than CSC (Zhang et al. 2016; Xue and Yan 2017). In general, these works illustrated that the gap in accuracy between the full and the truncated context is manageable, encouraging the development of new techniques to reduce this gap even more, to prepare the models for the streaming deployment with minimal WER loss.

Notwithstanding the several proposals, mainly focused on improving the WER, none of them are evaluated under real streaming conditions, these are, non-segmented long audio signals where, for example, normalization should be appropriately addressed. In addition, none of those approaches mentioned above incorporate the LSTM LM during the first pass of recognition, using the count-based model alone or following a two-steps approach. Finally, the latency of the system, a key factor in live captioning, it is, in the case where it is provided, only indicated in theoretical terms, and not measured under real conditions.

On the other hand, despite existing several proposals from the so-called end-to-end models (Alex Graves 2012; Chiu and Raffel 2018) for the streaming task (Raffel et al. 2017; Rao, Sak, and Prabhavalkar 2017), the performance of this kind of models, at the time this thesis started, was still far from the hybrid approach in complex tasks in terms of WER, conceivably due to not leveraging the vast text-only resources available (Prabhavalkar et al. 2017; Lüscher et al. 2019). For this reason, this work is mainly focused on the hybrid approach and how to adapt the pipeline to the streaming task.

3.7 Benchmarking

The proliferation of ASR systems that provide good quality results comes with the availability of open datasets and benchmarks with hundreds of hours to train and evaluate new approaches. Among these sets, the most popular one is LibriSpeech (Panayotov et al. 2015), due to its extension (∼1k training hours) and the academic community acceptance of their curated resources. Based on read speech, LibriSpeech provided a widely used benchmark to compare ASR models, being the basic workbench to validate new systems. However, LibriSpeech does not reflect the complexity of many real-world tasks ultimately. The lack of spontaneous speech poses some flaws in this dataset not reflecting, for example, hesitations, incomplete discourse, etc.

Another popular benchmark is TED-LIUM (Rousseau, Deléglise, and Estève 2012; Rousseau et al. 2014; Hernandez et al. 2018), an ASR corpus based on TED talks. Initially built for the International Conference on Spoken Language Translation (IWSLT) evaluation, this corpus has been evolving through different versions, extending its content. The second (Rousseau et al. 2014),and third (Hernandez et al. 2018) releases are the most widely used, providing around 200 and 450 hours, respectively, sharing the evaluation sets. In this case, the content is oriented to transcribe talks, where spontaneous speech is more common than in audiobooks, but it is still based on prepared discourse that does not cover all the variations from real-world tasks. In addition to this, it is essential to remark that the TED talks included in this dataset are pre-segmented, and all the published evaluation metrics are computed on these segmented version. This segmentation complicated the evaluation of streaming ASR systems, i.e.: truncating the context that the streaming system can benefit from and simplifying the task removing noisy segments. This segmentation encouraged us to reconsider the original corpus evaluation sets to adapt them to the task of live captioning during this thesis.

In summary, the typical ASR pipeline is devoted to the *offline* setup, which is why the majority of the benchmarks were thought to validate these approaches. These benchmarks involve short-span pre-segmented utterances, lasting not more than some seconds or minutes, at most. However, during this thesis, the main focus is to provide a production-ready and high-performing streaming ASR system. For this reason, there will be a particular focus on validating our approach under natural streaming conditions, meaning that the recognition will be performed not only on short utterances but long-span speeches or TV shows. This kind of content spans from minutes to hours. Therefore, we are considering well-known academic benchmarks and challenging real-world tasks with a wide variety of speakers, noise conditions, and topics. Both challenges, the long-span and the unconstrained conditions during the evaluation, are reflected as the third scientific goal.

3.8 Evaluation of ASR systems

To measure the improvements in ASR the most acknowledged and used metric is the Word Error Rate (WER). This metric compares the accuracy of the system's output (hypothesis) and the reference text (the ground truth). To obtain this WER value, previous to the evaluation step, the reference text is normalized; that is, punctuation and casing are removed. The WER is evaluated using the output of the system with the resulting normalized reference. For this evaluation, the following formula is considered:

$$WER = \frac{S + D + I}{N_r} \tag{1.14}$$

Where N_r is the total number of words in the reference, S, D, I are the minimum number of editing operations to perform in the reference text to obtain the system's output. In particular, the operations are: the number of substitutions (S) (i.e., cat -> cut), the number of deletions (D)(i.e., cat -> (nothing)) and the number of insertions (I)(i.e., (nothing) -> cat). The resulting value is an error ratio that indicates how close the reference is to the output.

Typically, from the computer-assisted transcription point of view, a WER of close to 15% involves a transcription effort in time equivalent to double the audio length to be transcribed. In comparison, a 35% WER involves an effort close to transcribing everything from scratch (Bain et al. 2005). Therefore, values around 15-20% are helpful for most of the ASR-related tasks, while values lower than 10-15% reflect an ASR system that can be used directly with almost no supervision (i.e., streaming of live events). From a practical point of view, automatic transcriptions of WER equal or less than 25% convey enough correct information to be useful (Munteanu et al. 2006), and professional stenographers prefer them to manually transcribing from scratch (Akita, Mimura, and Kawahara 2009).

There are other speed related measures that will be used during this thesis, such as the Real-Time Factor (RTF) or the latency, that measure the speed of the system working offline and on streaming, respectively. The RTF provides a factor to evaluate the time to transcribe the audio, computed with the time required to be transcribed divided by the length of the audio. For example, if two hours of audio are transcribed in one hour, the RTF is 0.5. Lower values indicate that the system works faster than higher ones. The second time measure, latency, gives an idea of the delay between the audio input and the provided output, commonly measured in seconds.

It is also important to remark that in a streaming setup, in which it is crucial to minimise the user-perceived system latency, the need of providing partial or intermediate outputs may arise. Regarding this, there will be a potential mismatch between those partial hypotheses providing a low latency feeling, and the

final output, with obviously a bit more delay. Empirically, we have checked that the correctness of the partial hypotheses differs very little from the final output, but measuring this mismatch falls outside of the scope of this work as this poses several difficulties (i.e.: how to decide which partial hypothesis to select at each time step).

To summarize, during this thesis, several approaches will be proposed, designed, implemented, and evaluated in terms of WER and RTF, to finally obtain a competitive streaming ASR system with low latency and high accuracy.

4 Framework

This work started with the Ph.D. program funded by the FPU (FPU14/03981) scholarship, working on theoretical models for general machine learning problems under streaming conditions, providing outputs for data input streams of diverse nature. After publishing some of these contributions being part of the Pattern Recognition and Human Language Technologies (PRHLT) group, there was a change in the direction of this thesis to work on ASR, in particular, on streaming ASR. To this end, the Ph.D. program continued as part of the Machine Learning and Language Processing (MLLP) group. Over more than a decade, the MLLP research group has achieved successful results on NLP technologies in research European projects, transLectures (2011-2014), EMMA (2014-2016), X5Gon (2017-2020); national projects, such as MORE (2016-2018) and Multisub (2019-2021); and technology transfer agreements such as Apptek (2017-2019), Tyris Software (2021); CERN (2020 and 2022-), À Punt (2020-), just to mention a few. Indeed, this accumulated in-depth theoretical and technical experience also came with broadened know-how to thoroughly accomplish technological transfer to meet the industry requirements. This application-oriented perspective stimulated the work developed on ASR during this thesis started in 2018, contributing to research projects and technology transfer agreements that expanded the scope of this work outside academia.

The main project that framed this thesis was the European project "X5Gon: Cross-Modal, Cross-Cultural, Cross-Lingual, Cross-Domain and Cross Site Global OER Network". X5Gon aspired to provide this collective OER repository from the worldwide sparse content available. The development of AI-based tools was a key factor during the whole project to promote the learning experience through the platform. Along with different sites collaborating through different modalities, this learning platform encourages the cross-cultural and cross-lingual aspects. The latter aspects are crucial to achieving the purpose of the project, and they are the entry point to include NLP technologies, such as ASR. To this end, the contributions to improving ASR systems were essential. More specifically, providing high-quality ASR tools to transcribe live lectures or classroom sessions was one of the motivations for the work done during this thesis. These systems were the

initial steps to include other tools such as recommendation or AI systems to guide the user through well-curated learning paths. Indeed, as mentioned in previous sections, live content is gaining traction in MOOC platforms. Due to this, providing systems with similar performance to the ones used for recorded content is required to preserve the quality of the learning experience. For this purpose, the work done for this thesis boosted the accessibility and cross-language education, enabling high-quality automatic transcription of OERs both in recorded and live content. In this project, two relevant repositories were selected as a case of study: VideoLectures.NET, a free and open access web portal that has published more than 20K educational videos, and PoliMedia, a high-quality multimedia educational repository developed by the UPV. It includes more than 15,000 Spanish video lectures lasting up to 10 minutes each, created by more than 1800 lecturers, summing up a total amount of about 3000 hours. In the context of X5Gon, several ASR systems were developed for OER transcription with these sets (VideoLectures.NET and PoliMedia) in 4 languages (En, Es, Sl, De). It is important to remark that these automatic transcriptions provided a sound basis for other advanced NLP tasks such as content translation and voice synthesis. All of these tasks were also successfully addressed by other members of the MLLP group, providing a fully developed speech-to-speech pipeline.

As referred above, two national projects also framed the context of this thesis as well: MORE and MultiSub. On the one hand, MORE, standing for Multilingual Open Resources for Education, was aimed to provide OER multilingual access and enable multilingual online communication in MOOC platforms. On the other hand, MultiSub targeted subtitling of classrooms and plenary sessions to further improve state of the art in ASR and Machine Translation for OER. Both projects included the results of the work of this thesis, developing ASR systems to transcribe classroom content and parliamentary debates.

About the technology transfer agreements, it is essential to highlight some of them that also framed the results obtained during this thesis. The most relevant is the R&D collaboration agreement between the Corporació Valenciana de Mitjans de Comunicació (À Punt) and the Universitat Politècnica de València (UPV) for real-time computer-assisted captioning/subtitling of audiovisual contents. As mentioned before, these professionals will benefit from high-performance ASR tools that help them to reduce the workload and stress, notably during broadcasting live events. This agreement allowed us to refine our research developments to convert them into products used intensively in a new real-world environment such as a broadcast channel. This conversion came with exciting challenges that were not considered previously and valuable end-user feedback to improve our tools and techniques.

Additionally, other transfer agreements also leveraged the developments on streaming ASR. Tyris Software, a technology company focused on applied AI solutions, uses our streaming and offline ASR technology to provide automatic transcription for its over-the-top (OTT) service, such as live football matches. Moreover,

the technology transfer contract to provide automatic speech transcription and translation services by UPV to the European Organization for Nuclear Research (CERN), using the latest ASR techniques to transcribe their formative content shared across the multi-lingual organization.

The developments of this thesis have contributed to achieving the goals of the preceding research projects by providing tools for building ASR systems and proposing new competitive models. More precisely, during all these projects and transfer agreements, there were developed ASR systems, both offline and streaming, general-purpose or adapted, in widely-spoken languages such as English or Spanish, as well as less widely-spoken languages, such as Slovenian or Catalan, where advanced ASR tools are not commonly available.

5 List of publications

5.1 Paper 1

Title	MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge
Authors	Jorge, Javier ; Martínez-Villaronga, Adrià ; Golik, Pavel ; Giménez, Adrià ; Silvestre-Cerdà, Joan Albert ; Doetsch, Patrick ; Císcar, Vicent Andreu ; Ney, Hermann ; Juan, Alfons ; Sanchis, Albert
Year	2018
Type	Workshop
DOI	10.21437/IberSPEECH.2018-54
Name	Proc. IberSPEECH 2018
Pages	257-261

This paper describes the Automatic Speech Recognition systems developed by the MLLP research group of Universitat Politècnica de València and the HLTPR research group of RWTH Aachen for the *IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge*. Our systems participated in both the closed and the open training conditions, with a closed dataset in the former and all the data that the participants had in the latter. Our best system built for the closed condition was our first version of the one-pass decoder, including neural network-based LMs, such as LSTM LMs. This decoder allowed us to provide a fast (under real-time speed) and accurate system, achieving a WER of 20.0% on the *dev2* set, and a 22.0% WER in the *test-2018* set, winning the competition in the closed track. For the open condition, the system from RWTH Aachen used approx. 3800 hours of training data from multiple sources and trained their RASR-RETURNN one-pass hybrid BLSTM-HMM ASR system. This system scored 15.6% WER on the *dev2* set, and 16.45% on *test-2018*, winning the open track. Finally, the highlights of

these systems include robust speech data filtering for acoustic model training and show-specific language modeling.

5.2 Paper 2

Title	Real-time One-pass Decoder for Speech Recognition Using LSTM Language Models
Authors	Jorge, Javier; Giménez, Adrià; Iranzo-Sánchez, Javier; Civera, Jorge; Sanchis, Albert; Juan, Alfons
Year	2019
Type	International Conference - Core A - CGS A
DOI	10.21437/Interspeech.2019-2798
Name	Proc. of the 20th Annual Conf. of the ISCA (Interspeech 2019)
Pages	3820-3824

This publication presented extensively the one-pass decoder with neural-based LMs to improve the traditional multi-pass rescoring hypotheses approach. The new architecture proposed in this work aimed to make the decoding efficient, leveraging new pruning techniques to reduce the real-time factor of the system. This architecture is based on the precomputation of static tables to compute the LM look-ahead score. These tables allow the system to drastically reduce the search effort and computation complexity. Along with these decoding improvements, several techniques were explored to alleviate the LSTM expensive computations, such as Variance Regularization and lazy evaluation. This system was evaluated in the well-known datasets LibriSpeech and TED-LIUM release 3. The final ASR system obtained competitive WERs with $\sim$0.6 RTFs. Finally, the one-pass decoder approach was compared with our decoupled two-pass decoder.

5.3 Paper 3

Title	LSTM-Based One-Pass Decoder for Low-Latency Streaming
Authors	Jorge, Javier; Giménez, Adrià; Iranzo-Sánchez, Javier; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons
Year	2020
Type	International Conference - CGS A
DOI	10.1109/ICASSP40776.2020.9054267
Name	Proc. of 45th ICASSP 2020
Pages	7814–7818

The main goal of this work was to extend the one-pass offline decoder proposed in the previous paper to be used under streaming conditions. These conditions involve several limitations, mainly for the acoustic signal processing, as the whole

utterance is unavailable in this scenario. Therefore, adapting the acoustic processing involved two steps: adapting the AM to work with a limited context and adapting the acoustic signal normalization scheme. For this purpose, starting from our decoder working faster than real-time (RTF < 1.0) keeping accuracy thanks to the neural LM integration, this paper covered these limitations that were not considered in the offline case. To adapt the neural AM, in particular a BLSTM model, instead of the whole input, the model processes the streaming input through an overlapping sliding window that will be averaged to obtain the acoustic scores. On the other hand, the normalization scheme was adapted to retain initial frames to compute the initial statistics, and then it is performed on the fly during recognition. Finally, this new version of our decoder was evaluated under a pure streaming setup on the well-known datasets LibriSpeech and TED-LIUM release 3, considering both WER and latency, obtaining competitive WER figures with low-latency outputs.

5.4 Paper 4

Title	Live Streaming Speech Recognition Using Deep Bidirectional LSTM Acoustic Models and Interpolated Language Models
Authors	Javier Jorge Adrià Giménez, Joan Albert Silvestre-Cerdà, Jorge Civera Albert Sanchis Alfons Juan
Year	2021
Type	Journal
DOI	10.1109/TASLP.2021.3133216
Name	IEEE/ACM Transactions on Audio, Speech, and Language Processing.
Pages	148 - 161

In this paper, most of the developments of previous works are wrapped up and extended to pose a production-ready streaming one-pass decoder using neural models. In addition to the LSTM LM, the decoder now includes a third LM based on the Transformer architecture, providing the possibility of a three-way interpolation among the count-based and the two neural-based models. The overlapping sliding window BLSTM was widely explained and extended with a new pruning approach named Acoustic Model Look-Ahead (AMLA) to reduce the search effort to improve the system's global speed. Additionally, the normalization scheme was revisited, and a new proposal was provided, reducing the global latency. In this case, along with the commonly used LibriSpeech and TED-LIUM release 2, two new tasks were used: the TED-LIUM release 2 evaluation sets in the original video format and the RTVE2018 sets. These two tasks enabled the proper evaluation of our streaming-oriented system under challenging real-world tasks without any previous manual or automatic segmentation. The final results

on these four sets showed that our system is a production-ready tool providing remarkable low WER with low-latency responses.

5.5 Paper 5

Title	MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge
Authors	Jorge, Javier; Giménez, Adrià; Baquero-Arnal, Pau; Iranzo-Sánchez, Javier; Pérez-González-de-Martos, Alejandro; Garcés Díaz-Munío, Gonçal V; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons
Year	2021
Type	Workshop
DOI	10.21437/IberSPEECH.2021-25
Name	Proc. IberSPEECH 2021
Pages	118-122

In this last paper, our previous streaming decoder was evaluated against other developments from international research and industry teams on the open-condition challenge IberSpeech2021. Our team submitted several hybrid ASR systems: two streaming systems with 0.6 and 1.5 seconds of future context and one setup with our traditional offline system. In the case of the streaming systems, the 1.5 used a linear interpolation of the three LMs (n-gram, LSTM LM, and Transformer LM) while the 0.6 setup, aimed to reduce the latency, used only the Transformer LM. The offline system consisted of a fast pre-recognition and voice activity detection step to detect speech/non-speech segments, followed by our real-time one-pass decoding using all three interpolated LMs.

Results showed that our streaming pipeline performed better in these TV broadcast tasks. Our best system (streaming with 1.5 seconds) provided a competition-winner WER of 16.0%, outperforming our offline system with an even outstanding 17.1% WER. Finally, in this challenging task, it is essential to remark that our fast system provided a remarkable WER of 16.9%, with a noticeable latency of $\sim$0.8 seconds, consolidating the system as a production-ready valuable tool to use directly in real-world environments.

6 References

Akita, Yuya, Masato Mimura, and Tatsuya Kawahara (Jan. 2009). "Automatic transcription system for meetings of the Japanese National Congress". In: *Proc. of InterSpeech 2009*, pp. 84–87 (cit. on p. 17).

Arisoy, E. et al. (2014). "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1, pp. 184–192 (cit. on pp. 12, 14).

Baevski, Alexei and Michael Auli (2019). "Adaptive Input Representations for Neural Language Modeling". In: *Proc. of ICLR 2019* (cit. on p. 11).

Bain, Keith et al. (Jan. 2005). "Accessibility, transcription, and access everywhere". In: *IBM Systems Journal* 44, pp. 589–604 (cit. on p. 17).

Baker, James K (1990). "Stochastic modeling for automatic speech understanding". In: *Readings in speech recognition*, pp. 297–307 (cit. on p. 7).

Baum, Leonard E and John Alonzo Eagon (1967). "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". In: *Bulletin of the American Mathematical Society* 73.3, pp. 360–363 (cit. on p. 7).

Bayes, Thomas (1763). "LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S". In: *Philosophical transactions of the Royal Society of London* 53, pp. 370–418 (cit. on p. 5).

Bengio, Yoshua et al. (2003). "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3, pp. 1137–1155 (cit. on p. 11).

Bourlard, Herve and Christian J Wellekens (1990). "Links between Markov models and multilayer perceptrons". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.12, pp. 1167–1178 (cit. on p. 10).

Bozheniuk, Vitalii et al. (2020). "A comprehensive study of Residual CNNs for acoustic modeling in ASR". In: *Proc. of ICASSP 2020* (cit. on p. 10).

Chan, William et al. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. of ICASSP 2016*, pp. 4960–4964 (cit. on p. 6).

Chen, Kai and Qiang Huo (2016). "Training deep bidirectional LSTM acoustic model for LVCSR by a Context-Sensitive-Chunk BPTT approach". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.7, pp. 1185–1193 (cit. on p. 15).

Chen, Stanley F. and Joshua Goodman (1999). "An empirical study of smoothing techniques for language modeling". In: *Computer Speech and Language* 13.4, pp. 359–394 (cit. on p. 10).

Chen, Xie et al. (2017). "Future word contexts in neural network language models". In: *Proc. of ASRU 2017*, pp. 97–103 (cit. on p. 13).

Chiu, Chung-Cheng and Colin Raffel (2018). "Monotonic Chunkwise Attention". In: *Proc. of ICLR 2018* (cit. on p. 15).

Dai, Zihang et al. (2019). “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Proc. of ACL 2019*, pp. 2978–2988 (cit. on p. 11).

Davis, S. and P. Mermelstein (1980). “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4, pp. 357–366 (cit. on p. 7).

Deng, Li et al. (May 2013). “Recent Advances in Deep Learning for Speech Research at Microsoft”. In: *Proc. of ICASSP 2013* (cit. on p. 3).

Graves, A. and J. Schmidhuber (2005). “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *Neural networks* 18.5-6, pp. 602–610 (cit. on p. 14).

Graves, Alex (2012). “Sequence transduction with recurrent neural networks”. In: *CoRR* (cit. on p. 15).

Hernandez, François et al. (2018). “TED-LIUM 3: Twice as Much Data and Corpus Repartition for Experiments on Speaker Adaptation”. In: *Proc. of SPECOM 2018*, p. 198 (cit. on p. 16).

Hinton, Geoffrey et al. (2012). “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal processing magazine* 29.6, pp. 82–97 (cit. on pp. 10, 14).

Hochreiter, S. and J. Schmidhuber (Nov. 1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 11).

Hori, Takaaki et al. (2007). “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, pp. 1352–1365 (cit. on p. 12).

Irie, Kazuki et al. (2019). “Language Modeling with Deep Transformers”. In: *Proc. of InterSpeech 2019*, pp. 3905–3909 (cit. on p. 11).

Lee, Kyungmin et al. (2018). “Accelerating recurrent neural network language model based online speech recognition system”. In: *Proc. of ICASSP 2018*, pp. 5904–5908 (cit. on pp. 12, 14).

Lüscher, Christoph et al. (2019). “RWTH ASR systems for LibriSpeech: Hybrid vs Attention”. In: *Proc. of InterSpeech 2019*, pp. 231–235 (cit. on p. 15).

Mikolov, Tomas et al. (2010). “Recurrent neural network based language model.” In: *Proc. of InterSpeech 2010*, pp. 1045–1048 (cit. on p. 11).

Mohamed, Abdel-rahman, George Dahl, Geoffrey Hinton, et al. (2009). “Deep belief networks for phone recognition”. In: *Proc. of NIPS 2009*. Vol. 1. 9. Vancouver, Canada, p. 39 (cit. on p. 3).

Mohamed, Abdel-rahman, George E. Dahl, and Geoffrey Hinton (2012). “Acoustic Modeling Using Deep Belief Networks”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1, pp. 14–22 (cit. on p. 3).

Mohamed, Abdel-rahman, Frank Seide, et al. (2015). “Deep bi-directional recurrent networks over spectral windows”. In: *Proc. of ASRU 2015*, pp. 78–83 (cit. on pp. 14, 15).

Mohri, Mehryar and Michael Riley (1999). "Integrated context-dependent networks in very large vocabulary speech recognition". In: *Proc. of ECSCT 1999* (cit. on p. 12).
– (2001). "A weight pushing algorithm for large vocabulary speech recognition". In: *Proc. of ECSCT 2001* (cit. on p. 12).
Munteanu, Cosmin et al. (Apr. 2006). "The effect of speech recognition accuracy rates on the usefulness and usability of webcast archives". In: *Proc. of Conference on Human Factors in Computing Systems 2006*. Vol. 1, pp. 493–502 (cit. on p. 17).
Ney, Hermann (1984). "The use of a one-stage dynamic programming algorithm for connected word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2, pp. 263–271 (cit. on p. 11).
Ney, Hermann and Stefan Ortmanns (2000). "Progress in dynamic programming search for LVCSR". In: *Proc. of IEEE 2000* 88.8, pp. 1224–1240 (cit. on p. 12).
Nolden, David (Apr. 2017). "Progress in Decoding for Large Vocabulary Continuous Speech Recognition". PhD thesis. Computer Science Department RWTH Aachen University Aachen (Germany): RWTH Aachen University (cit. on p. 12).
O'Shaughnessy, Douglas (2008). "Invited paper: Automatic speech recognition: History, methods and challenges". In: *Pattern Recognition* 41.10, pp. 2965–2979 (cit. on p. 2).
Ogawa, A. et al. (2018). "Rescoring N-Best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model". In: *Proc. of ICASSP 2018*, pp. 6099–6103 (cit. on p. 3).
Panayotov, V. et al. (2015). "Librispeech: an ASR corpus based on public domain audio books". In: *Proc. of ICASSP 2015*, pp. 5206–5210 (cit. on p. 16).
Park, Daniel S. et al. (2019). "SpecAugment: A Simple Augmentation Method for Automatic Speech Recognition". In: *Proc. of InterSpeech 2019* (cit. on p. 7).
Povey, Daniel et al. (2011). "The Kaldi Speech Recognition Toolkit". In: *Proc. of ASRU 2011* (cit. on p. 12).
Prabhavalkar, Rohit et al. (2017). "A Comparison of Sequence-to-Sequence Models for Speech Recognition". In: *Proc. of InterSpeech 2017*, pp. 939–943 (cit. on p. 15).
Radford, Alec et al. (2019). *Language Models are Unsupervised Multitask Learners* (cit. on p. 11).
Raffel, Colin et al. (2017). "Online and Linear-Time Attention by Enforcing Monotonic Alignments". In: *Proc. of ICML 2017*, pp. 2837–2846 (cit. on p. 15).
Rao, Kanishka, Haşim Sak, and Rohit Prabhavalkar (2017). "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer". In: *Proc. of ASRU 2017*, pp. 193–199 (cit. on p. 15).
Rousseau, Anthony et al. (2014). "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks." In: *Proc. of LREC 2014*, pp. 3935–3939 (cit. on p. 16).
Rousseau, Anthony, Paul Deléglise, and Yannick Estève (May 2012). "TED-LIUM: an Automatic Speech Recognition dedicated corpus". In: *Proc. of LREC 2012*. Istanbul, Turkey, pp. 125–129 (cit. on p. 16).

Rybach, David et al. (Apr. 2009). "Audio segmentation for speech recognition using segment features". In: pp. 4197–4200. DOI: 10.1109/ICASSP.2009.4960554 (cit. on p. 13).

Sainath, Tara N et al. (2015). "Deep convolutional neural networks for large-scale speech tasks". In: *Neural Networks* 64, pp. 39–48 (cit. on p. 10).

Schlüter, Ralf et al. (2007). "Gammatone Features and Feature Combination for Large Vocabulary Speech Recognition." In: *Proc. of ICASSP 2007*, pp. 649–652 (cit. on p. 7).

Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cit. on p. 10).

Seide, Frank, Gang Li, and Dong Yu (2011). "Conversational speech transcription using context-dependent deep neural networks". In: *Twelfth annual conference of the international speech communication association* (cit. on p. 3).

Shannon, C. E. (1948). "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3, pp. 379–423 (cit. on p. 10).

Shannon, Claude Elwood (2001). "A mathematical theory of communication". In: *ACM SIGMOBILE mobile computing and communications review* 5.1, pp. 3–55 (cit. on p. 6).

Si, Yujing et al. (2013). "Prefix tree based n-best list re-scoring for recurrent neural network language model used in speech recognition system." In: *Proc. of InterSpeech 2013*, pp. 3419–3423 (cit. on p. 3).

Singh, Mittul et al. (2017). "Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition". In: *Proc. of InterSpeech 2017*, pp. 2720–2724 (cit. on pp. 12, 14).

Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney (2012). "LSTM neural networks for language modeling". In: *Proc. of InterSpeech 2012* (cit. on p. 11).

Sundermeyer, Martin, Zoltán Tüske, et al. (2014). "Lattice decoding and rescoring with long-span neural network language models". In: *Proc. of the InterSpeech 2014* (cit. on p. 13).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS 2017*, pp. 5998–6008 (cit. on p. 11).

Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269 (cit. on pp. 9, 11).

Wald, Mike et al. (2007). "Correcting automatic speech recognition captioning errors in real time". In: *International Journal of Speech Technology* 10.1, pp. 1–15 (cit. on p. 2).

Xu, H. et al. (2018). "A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition". In: *Proc. of ICASSP 2018*, pp. 5929–5933 (cit. on pp. 3, 13).

Xue, S. and Z. Yan (2017). "Improving latency-controlled BLSTM acoustic models for online speech recognition". In: *Proc. of ICASSP*, pp. 5340–5344 (cit. on p. 15).

Yang, Peter (2020). "The Cognitive and Psychological Effects of YouTube Video Captions and Subtitles on Higher-Level German Language Learners". In: *Technology and the Psychology of Second Language Learners and Users*, pp. 83–112 (cit. on p. 2).

Young, S. J., J. J. Odell, and P. C. Woodland (1994). "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proc. of Workshop on Human Language Technology 1994*, pp. 307–312 (cit. on p. 9).

Zeyer, A., R. Schlüter, and H. Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Proc. of InterSpeech 2016*, pp. 3424–3428 (cit. on pp. 14, 15).

Zhang, Yu et al. (2016). "Highway long short-term memory RNNs for distant speech recognition". In: *Proc. of ICASSP 2016*, pp. 5755–5759 (cit. on p. 15).

Chapter 2

Selected Papers

1 MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge

Jorge, Javier; Martínez-Villaronga, Adrià; Golik, Pavel; Giménez, Adrià; Silvestre-Cerdà, Joan Albert; Doetsch, Patrick; Císcar, Vicent Andreu; Ney, Hermann; Juan, Alfons; Sanchis, Albert

Proc. of IberSPEECH 2018: 10th Jornadas en Tecnologías del Habla and 6th Iberian SLTech Workshop, pp. 257–261

Barcelona (Spain)

DOI 10.21437/IberSPEECH.2018-54

21-23 November 2018

MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge

Jorge, Javier; Martínez-Villaronga, Adrià; Golik, Pavel; Giménez, Adrià; Silvestre-Cerdà, Joan Albert; Doetsch, Patrick; Císcar, Vicent Andreu; Ney, Hermann; Juan, Alfons; Sanchis, Albert

Abstract

This paper describes the Automatic Speech Recognition systems built by the MLLP research group of Universitat Politècnica de València and the HLTPR research group of RWTH Aachen for the *IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge*. We participated in both the closed and the open training conditions.

The best system built for the closed condition was an hybrid BLSTM-HMM ASR system using one-pass decoding with a combination of a RNN LM and show-adapted n-gram LMs. It was trained on a set of reliable speech data extracted from the *train* and *dev1* sets using MLLP's transLectures-UPV toolkit (TLK) and TensorFlow. This system achieved 20.0% WER on the *dev2* set.

For the open condition we used approx. 3800 hours of out-of-domain training data from multiple sources and trained a one-pass hybrid BLSTM-HMM ASR system using open-source tools RASR and RETURNN developed at RWTH Aachen. This system scored 15.6% WER on the *dev2* set. The highlights of these systems include robust speech data filtering for acoustic model training and show-specific language modeling.

1.1 Introduction

This paper describes the joint collaboration between the *Machine Learning and Language Processing* (MLLP) research group from the *Universitat Politècnica de València* (UPV) and the *Human Language Technology and Pattern Recognition* (HLTPR) research group from the *RWTH Aachen University* for the participation in the IberSpeech-RTVE 2018 Speech-to-Text transcription challenge, that will be held during the IberSpeech 2018 conference in Barcelona, Spain. Our participation consisted of the submission of three systems: one primary and one contrastive for the closed system training condition, and one primary for the open condition.

The rest of the paper is structured as follows. First, Section 1.2 describes the RTVE database that was provided by the organizers of the challenge. Second, in Section 1.3 we describe the ASR systems we developed under the closed training conditions. Next, Section 1.4 details the ASR system that participated in the open training conditions. Finally, Section 1.5 provides a summary of the work and gives some concluding remarks.

1.2 RTVE database

The RTVE (*Radio Televisión Española*) database consists of a collection of 15 different TV shows broadcast by the Spanish public TV station between 2015 and 2018. It comprises 569 hours of audio data, from which 460 hours are provided with subtitles, and 109 hours have been human-revised. In addition, the database includes 3 million sentences extracted from the subtitles of 2017 broadcast news at the RTVE 24H Channel.

The database is provided in five partitions: *subs-C24H*, the 3M text dataset from the RTVE 24H channel; *train*, that comprises 463 hours of audio data with non-verbatim subtitles from 16 TV shows; and *dev1*, *dev2*, *test*, consisting of 57, 15 and 40 hours from 5, 2 and 8 different TV shows, respectively. *dev1* and *dev2* sets were provided with manual corrected transcripts, while the *test* set was used to gauge the performance of the participant systems. It is important to note that there is a certain overlap between *dev1*, *dev2*, *test* and the *train* set in terms of TV-shows.

In order to have available as many training data as possible during the development stage of the closed-condition system, we decided to split *dev1* into two subsets: *dev1-train*, comprising 43 hours of raw audio to be used for acoustic and language model training; and *dev1-dev*, 15 hours to be used internally as development data to optimize different components of the system. This split was done at the file level, trying to satisfy that *dev1-dev* have similar size to *dev2*, and that both *dev1* subsets include files from the same TV shows. We used *dev2* as a test set to measure the performance of the system on unseen data.

1.3 Closed-condition system

Speech data filtering

Under the closed training conditions, it is extremely important to make the most of the provided training data, specially when it is scarce and/or noisy. This is the case of the RTVE database: on the one hand, the *train* set comprises only 463 hours of audio, which is not much compared with the amount of data used to train current state-of-the-art systems (Chiu et al. 2018; Xiong et al. 2018). On the other hand, training data is not provided with verbatim transcripts but with approximate subtitles. This becomes a major concern when using recurrent neural networks for acoustic modeling, as the accuracy drops significantly when using noisy training data. Therefore, a robust speech data filtering procedure becomes a key point to achieve high ASR performance.

After examining some random samples from *train*, *dev1* and *dev2* sets, we first-hand checked that the provided subtitles are far from being verbatim transcripts, but also noted the presence of 1) subtitle/transcription gaps, 2) subtitle files with no timestamps, 3) audio files considerably larger than their corresponding subtitle files, 4) subtitle files covering timestamps that exceed the length of their corresponding audio file, or 5) human transcription errors in *dev1*, among others.

For all these reasons, we applied the following speech data filtering pipeline. As subtitle timestamps 1) are not reliable in the *train* set, 2) are not given in *dev1*, and 3) are given only at speaker-turn level in *dev2*, we first force-aligned each audio file to its corresponding subtitle/transcript text. We did this using a preexisting hybrid CD-DNN-HMM ASR (Hinton et al. 2012) system in which the search space was constrained to recognize the exact text (no language model was involved in this procedure), with the only freedom of exploring the different word pronunciations given by the lexicon model and of using an optional silence phoneme at the beginning of each word. In this way we computed the best alignment between the input frame sequence to the sequence of HMM states inferred from the subtitle/transcript text. Then, we applied a heuristic post-filtering based on state-level frame occupation and word-level alignment scores: if either an HMM state is aligned to more frames than the observed average state frame occupation + two times the observed standard deviation, or a word whose average alignment score is lower than a given threshold, then the corresponding word alignment is considered noisy and the word is removed. Next, we completely discarded those files in which more than two thirds of the words were filtered out in the previous step. Finally, we built a clean training corpus by joining words into segments whose boundaries were delimited by large-enough silences and deleted words.

Table 2.1 shows the result of applying this speech data filtering pipeline to the *train*, *dev1* and *dev2* sets. The second column shows the raw audio length in hours of each set. The third refers to the amount of raw hours that could be

Table 2.1: Number of raw, aligned raw, aligned speech, and filtered speech hours as a result of applying the speech data filtering pipeline to the whole RTVE database.

	Raw	Aligned		Filtered
		Raw	Speech	Speech
train	463	438	252	187
dev1-train	43	31	24	18
dev1-dev	14	12	9	7
dev2	15	12	9	6
Overall	535	493	294	218

aligned to the corresponding subtitle/transcript text by our alignment system. It must be noted that there were some audio files that the system was not capable to align. This happens when none of the active hypotheses can reach the final HMM state at the last time step, due to an excessive histogram pruning or due to an non-matching transcript. For this reason, 42 hours of audio data could not be aligned. The fourth column gives the total amount of aligned speech data after removing non-speech events that were aligned to the silence phoneme. Surprisingly, the original 438 raw hours from the *train* set were reduced to 252 hours of speech data, i.e. we detected 186 hours of non-speech events. After some manual analysis of the alignments, we found that a significant portion of these 186 hours is explained by non-subtitled speech, whose corresponding audio frames were in practice aligned to the silence phoneme. Finally, the fifth column shows the number of hours of clean speech data after applying the described heuristic post-filtering procedure and after discarding files that shown a high word rejection rate. Starting with the original 535 hours of raw audio, we aligned 294 hours of speech, from which we rejected 76 hours of noisy data, ending up with 218 hours of speech suitable for acoustic training.

Acoustic modeling

The acoustic models (AM) used during the development of the *MLLP-RWTH_c1-dev_closed* system were trained using filtered speech data from *train* and *dev1-train* sets, that is, 205 hours of training speech data. We extracted 16-dim. MFCC features augmented by the full first and second time derivatives, resuling in 48-dim. features.

Our acoustic models were based on the hybrid approach (Bourlard and Wellekens 1989; Hochreiter and Schmidhuber 1997). We first trained a conventional context-dependent Gaussian mixture model hidden Markov model (CD-GMM-HMM) with three left-to-right states. The state-tying schema was estimated following a phonetic decision tree approach (Young, Odell, and Woodland 1994), resulting in 8.9K tied states. The GMM acoustic model was used to force align the training data. We then trained a context-dependent feed-forward DNN-HMM using a context window of 11 frames, six hidden layers with ReLU activation functions

Table 2.2: Corpus statistics of the text data used for LM training.

	Sentences	Running words	Vocabulary
train	340K	4.3M	80K
subs-C24H	3.1M	57M	160K
RNN-train	1.8M	35M	176K
dev1-dev	9.9K	160K	13K
dev2	7.7K	150K	12K

and 2048 units per layer. We used the transLectures-UPV toolkit (TLK) (M. del-Agua et al. 2014) to train both GMM and DNN acoustic models.

Apart from the feed-forward model, we also trained a BLSTM-HMM model (Hochreiter and Schmidhuber 1997). The DNN was used to refine the alignment between input acoustic features and HMM states. We then trained the BLSTM-HMM model using the open source toolkit TensorFlow (Abadi, Agarwal, et al. 2015) and TLK. The BLSTM network consisted of four bidirectional hidden layers with 512 LSTM cells per layer and per direction.

In order to increase the amount of training data, the final submitted system (*MLLP-RWTH_p-final_closed*) was retrained on a total of about 218 hours from sets *train*, *dev1* and *dev2*.

Language modeling

Our language model (LM) for the closed condition consists of a combination of several n-gram models and a recurrent neural network (RNN) model. Also, since TV shows of each audio file are known in advance, we performed an LM adaptation at the n-gram model level.

First, we extracted sentences from all `.srt` and `.trn` files. Then we applied a common text processing pipeline to normalize capitalization, remove punctuation marks, expand contractions (i.e. *sr.* $\rightarrow$ *señor*) and transliterate numbers. As already mentioned, we split *dev1* into two subsets, *dev1-train* and *dev1-dev*, in order to include *dev1-train* in training.

Thus, in this section, we will refer to the combination of *train* and *dev1-train* simply as *train*. For LSTM LM training, we concatenated the *train* and *subs-C24H* sets into a single training file and removed redundancy by discarding repeated sentences. Also, sentences were shuffled after each epoch to allow better generalization. To carry out TV-show LM adaptation experiments, we randomly extracted 500 sentences of each TV show from the *train* set to be used as validation data in the adaptation process. Table 2.2 provides corpus statistics after normalization.

Second, to define our closed-condition system's vocabulary, we first computed the vocabulary of both *train* and *subs-C24H* sets, and then removed singletons, so that language models can properly model unknown word probabilities. After applying these two steps, the resulting vocabulary had 132K words. The out-of-vocabulary ratios of *dev1-dev* and *dev2* sets were 0.36% and 0.53%, respectively.

Third, we trained two standard Kneser-Ney smoothed 4-gram LMs on the *train* and *subs-C24H* sets using the SRILM toolkit (Stolcke 2002). Rows (a) and (b) of Table 2.3 show the perplexities obtained with these models on the *dev1-dev* and *dev2* sets. In addition to these two general n-gram LMs, we trained one n-gram LM for each TV show. Row (c) of Table 2.3 shows the averaged perplexity of the corresponding TV-show-specific LM for each file.

Next, we trained a RNN LM using the Variance Regularization (VR) criterion (X. Chen et al. 2015). This criterion reduces the computational cost during the test phase. Our models were trained on GPU devices using the CUED-RNNLM toolkit (Xie Chen, Liu, et al. 2016). The network setup was optimized to minimize perplexity on the *dev1-dev* set. It consisted of a 1024-unit embedding layer and a hidden LSTM layer of 1024 units. The output layer is a 132K-unit softmax, whose size corresponds to the vocabulary size. The perplexities obtained with this network are depicted in Row (f) of Table 2.3.

Then, the combination of the LMs was done in two steps. Firstly, we performed a linear interpolation of n-gram models. For the general, non-adapted models, we interpolated the LMs estimated on the *train* and the *subs-C24H* sets by minimizing the perplexity on *dev1-dev* (Jelinek and Mercer 1980). Row (d) of Table 2.3 shows the perplexities for this particular LM combination. For each show-specific LM, we performed a three-way interpolation: the individual show-specific LM, the *train* LM and the *subs-C24H* LM. In this case, interpolation weights were optimized individually for each TV show so that the perplexity was minimized on the corresponding 500-sentence show-specific validation set, similarly to the approach followed in (Martínez-Villaronga, Agua, et al. 2013; Martínez-Villaronga, M. A. del-Agua, et al. 2014). Secondly, we combined the interpolated n-gram LMs with the RNN LM. Other than the static interpolation of n-gram LMs, the result of this step is not a new monolithic model, but a set of interpolation weights to be used on-the-go by the ASR decoder during search. Perplexities for the combination of the RNN LM with the general and the adapted n-gram LMs can be found in Rows (g) and (h) of Table 2.3.

Finally, to take the most of the provided data, the final submitted system (*MLLP-RWTH_p-final_closed*) was trained using the same hyper-parameters values estimated during the development stage, but using also *dev1-dev* and *dev2* sets as part of the training data.

Table 2.3: Perplexities of the different LM components.

	dev1-dev	*dev2*
(a) N-gram *train*	139.6	183.0
(b) N-gram *subs-C24H*	161.2	193.4
(c) N-gram show-specific	184.0	294.3
(d) N-gram general (a+b)	107.0	147.5
(e) N-gram adapt (a+b+c)	99.5	139.1
(f) RNN	92.3	110.7
(g) RNN+N-gram general (d+f)	78.2	101.8
(h) RNN+N-gram adapt (e+f)	68.9	99.2

Experiments and results

In this section we describe the experiments carried out to determine the best closed training condition system. Our experiments were devoted to assess three components of the system: acoustic models, language models and voice activity detection (VAD) modules. In all cases we used the TLK toolkit decoder (M. del-Agua et al. 2014) for recognizing test data using a one-pass decoding setup.

First, we compared the performance of the CD-FFDNN-HMMs and BLSTM-HMMs acoustic models described in Section 1.3. In both cases we used the general n-gram language model described in Section 1.3. Grammar scale factor and search pruning parameters were optimized on the *dev1-dev* set. Table 2.4 shows the results on both *dev1-dev* and *dev2* sets. As expected, the BLSTM acoustic model outperformed the feed-forward model by 12.2% relative.

Table 2.4: Comparison of the CD-FFDNN-HMM and BLSTM-HMM acoustic models using the general n-gram language model. Results in WER % and relative WER % improvement.

	dev1-dev	*dev2*	
	WER	WER	ΔWER
FFDNN	29.7	27.1	-
BLSTM	26.5	23.8	12.2

Next, we analyzed the contribution of different LM combinations during search, leaving fixed the acoustic model to the best BLSTM neural network. Specifically, we carried out recognition experiments using (1) the general n-gram LM, (2) the RNN LM, (3) the interpolation of the RNN LM with the general n-gram LM, and (4) the interpolation of the RNN LM with the adapted, show-specific n-gram LMs. Table 2.5 shows perplexities and WERs for the *dev1-dev* and *dev2* sets over these four different LM setups.

The best results were obtained with the combination of RNN and n-gram models, showing a consistent 6% relative improvement in both sets over the baseline general n-gram LM. It is worth noting that in terms of WER, the improvement from using adapted models does not translate to *dev2*. As *dev1-dev* and *dev2*

Table 2.5: Comparison of different language model combinations using the BLSTM-HMM acoustic model in terms of perplexity, WER % and relative WER % improvement.

	dev1-dev		*dev2*		
	PPL	WER	PPL	WER	ΔWER
n-gram general	107	26.5	148	23.8	-
RNN	92	26.2	111	23.0	3.4
RNN + n-gram general	78	25.3	102	22.4	5.9
RNN + n-gram adapt	69	24.8	99	22.4	5.9

contain different shows with strongly varying amounts of show-specific text data available for training, not all shows benefit from adaptation equally. Anyway, since the adaption does not degrade the system performance, and given the good improvement seen on *dev1-dev*, we decided to use the combination of RNN LMs plus adapted n-gram LMs for the final system.

Looking at the system outputs, after carrying out error analysis, we realized that our VAD module (Silvestre-Cerdà et al. 2012) was discarding a significant amount of speech regions in the audio files. This significantly affected the WER by increasing the number of deletions. For this reason, we decided to explore other audio segmentation approaches and compare its performance in terms of WER. Concretely, we compared the following approaches: (1) our baseline MLLP-UPV VAD system, based on a speech/non-speech GMM-HMM classifier that ranked second in the Albayzin-2012 audio segmentation challenge (Silvestre-Cerdà et al. 2012); (2) The LIUM Speaker Diarization Tools, a VAD system based on Generalized Likelihood Ratio between speech/non-speech Gaussian models (Meignier and Merlin 2010); (3) The well-known CMUseg audio segmentation system using the standard configuration (Siegler et al. 1997); (4) Apply a fast pre-recognition step to segment the audio file by the recognized silences, using the best CD-FFDNN-HMM acoustic model and a pruned version of the general n-gram LM; and (5) Use the segments generated in (4), and apply VAD the system (1) to classify those segments into speech/non-speech. It is important to note that (3) and (4) are not VAD systems but just audio segmenters, so all detected segments are considered speech, i.e. all audio is passed through to the ASR. Table 2.6 shows the WER for each of the five audio segmentation/VAD techniques, including the ratio of discarded audio that is dropped by the VAD prior to decoding.

As we expected, the baseline VAD system (1) was discarding too much segments, as it was too aggressive compared to other techniques. With either (2) or (3) we obtained a consistent improvement. It was further increased up to 8% by using (4). We decided then to combine this segmentation with our baseline VAD system (1), which led us to achieve an 11% relative WER improvement. In absolute terms, we got a 2.4 WER points gain in *dev2*, with a final WER of 20.0%. This setup constituted our contrastive closed-condition system (*MLLP-RWTH_c1-dev_closed*), whilst our primary system (*MLLP-RWTH_p-final_closed*) was the result of re-

Table 2.6: Comparison of different audio segmentation/VAD techniques using the BLSTM-HMM acoustic model and the combination of the RNN LM + adapted n-gram LM. Results in WER % and relative WER % improvement and the ratio of dropped audio.

	dev1-dev		*dev2*		
	% drop.	WER	% drop.	WER	ΔWER
MLLP-UPV (1)	10.9	24.8	5.9	22.4	-
LIUM (2)	7.1	23.7	3.9	20.8	7.1
CMUseg (3)	0	23.2	0	20.9	6.7
Pre-Recognition (4)	0	22.9	0	20.6	8.0
+ MLLP-UPV (5)	3.2	22.3	3.3	20.0	10.7

training the same acoustic and language models with all available data, as stated in Sections 1.3 and 1.3.

Finally we analyzed the speed of submitted system in terms of Real Time Factor (RTF). We studied how tightening the pruning parameters affects the RTF and the WER. Also, to assess the speed of a fast pre-recognition step to segment the audio signal, we also did this comparison using the LIUM VAD system. Results of this analysis are shown in Table 2.7.

Table 2.7: Speed analysis in terms of RTF an its effect on the WER% over the *dev2* set, either with the submitted system and removing the pre-precognition step, using LIUM VAD instead.

	RTF	WER
Submitted system (1)	1.5	20.0
+ inc. prune	0.8	20.3
(1) with LIUM VAD	1.0	20.9
+ inc. prune	0.4	21.3

First, a more aggressive pruning resulted in a system 1.88 times faster while degrading the WER by 0.3% absolute. Next, if we replace the pre-recognition step on the submitted system by the LIUM VAD module, we get a speed-up of 50% at the cost of 0.9 points WER. Finally, we could afford a system 3.75 times faster if we tighten the prune parameters when using LIUM VAD, with a WER loss of 1.3 absolute points, although it would still be a competitive system, scoring 21.3% WER points on *dev2*.

1.4 Open-condition system

The main motivation for participating in the open-condition track was the desire to evaluate a system developed in the recent months for a different purpose, not related to the IberSpeech challenge. In order to achieve this goal, we decided to keep the amount of parameter optimization as low as possible. This system is based on the software developed at RWTH Aachen University: RASR (Rybach et al. 2011; Wiesler et al. 2014) and RETURNN (Doetsch et al. 2017; Zeyer, Alkhouli, and Ney 2018).

The ASR system is based on a hybrid LSTM-HMM acoustic model. It was trained on a total of approx. 3800 hours of transcribed speech from several sources, covering a variety of domains and acoustic conditions. The collection consists of subtitled videos crawled from Spanish and Latin American websites.

We used a pronunciation lexicon with a vocabulary size of 325k with one or more pronunciation variants. The acoustic model takes 80-dim. MFCC features as input and estimates state posterior probabilities for 5000 tied triphone states. The state tying was obtained by estimating a classification and regression tree (CART) on all available training data. Acoustic modeling was done using a bi-directional LSTM network with four layers and 512 LSTM units in each layer. About 30% of activations are dropped in each layer for regularization purpose (Srivastava et al. 2014). During training we minimized the cross-entropy of a network generated distribution in the softmax output layer at aligned label positions using a Viterbi alignment defined over the 5000 tied triphone states of the CART. We used the Adam learning rate schedule (Kingma and Ba 2015) with integrated Nesterov momentum and further reduced the learning rate following a variant of the Newbob scheme. We split input utterances into overlapping chunks of roughly 10 seconds and perform an L2 normalization of the gradients for each chunk. With the normalized gradients the network is updated in a stochastic gradient descent manner where batches containing up to 50 chunks are distributed over eight GPU devices and recombined into a common network after roughly 500 chunks have been processed by all devices.

The language model for the single-pass HMM decoding is a 5-gram count model trained with Kneser-Ney smoothing on a large body of text data collected from multiple publicly available sources. Its perplexity on *dev1-dev* and *dev2* is 173.5 and 173.2 respectively. This open-track system has reached a WER of 18.3% and 15.6% on *dev1-dev* and *dev2* without any speaker or domain adaptation or model tuning.

1.5 Conclusions

In this paper we have presented the description of the three systems that participated in the IberSpeech-RTVE 2018 Speech-to-Text transcription challenge. Two of them, one primary (*MLLP-RWTH_p-final_closed*) and one contrastive (*MLLP-RWTH_c1-dev_closed*), were submitted to the closed training conditions, while the other one (*MLLP-RWTH_p-prod_open*) participated in the open training track. On the one hand, our best development closed-condition ASR system (*MLLP-RWTH_c1-dev_closed*), consisting of a BLSTM-HMM acoustic model trained on a reliable set of 205 hours of training speech data, and a combination of both RNN and TV-show adapted n-gram language models, achieved a competitive mark of 20.0% WER on the *dev2* set. Our final, primary closed-condition ASR system (*MLLP-RWTH_p-final_closed*) should offer a similar or even better performance as it followed the same system design setup but trained

with all available data, including both development sets. On the other hand, our general-purpose open-condition ASR system (*MLLP-RWTH_p-prod_open*), without carrying out any speaker, domain nor model adaptation of any kind, scored 15.6% WER on the *dev2* set.

References

Abadi, Martín, Ashish Agarwal, et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (cit. on p. 37).

Bourlard, H. and C. J. Wellekens (1989). "Links between Markov Models and Multilayer Perceptrons". In: *Advances in Neural Information Processing Systems I*, pp. 502–510 (cit. on p. 36).

Chen, X. et al. (2015). "Improving the training and evaluation efficiency of recurrent neural network language models". In: *Proc. of ICASSP 2015*, pp. 5401–5405 (cit. on p. 38).

Chen, Xie, Xunying Liu, et al. (2016). "CUED-RNNLM – An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *Proc. of ICASSP 2016*. Shanghai, China, pp. 6000–6004 (cit. on p. 38).

Chiu, Chung-Cheng et al. (Apr. 2018). "State-of-the-art Speech Recognition With Sequence-to-Sequence Models". In: *Proc. of ICASSP 2018*. Calgary, Canada, pp. 4774–4778 (cit. on p. 35).

del-Agua, M.A. et al. (Nov. 2014). "The translectures-UPV toolkit". In: *Proc. of Advances in Speech and Language Technologies for Iberian Languages 2014*, pp. 269–278 (cit. on pp. 37, 39).

Doetsch, Patrick et al. (Mar. 2017). "RETURNN: The RWTH extensible training framework for universal recurrent neural networks". In: *Proc. of ICASSP 2017*. New Orleans, LA, USA, pp. 5345–5349 (cit. on p. 41).

Hinton, G. et al. (Nov. 2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97 (cit. on p. 35).

Hochreiter, S. and J. Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on pp. 36, 37).

Jelinek, Frederick and Robert L. Mercer (Apr. 1980). "Interpolated estimation of Markov source parameters from sparse data". In: *Proc. Workshop on Pattern Recognition in Practice 1980*. Amsterdam, Netherlands, pp. 381–397 (cit. on p. 38).

Kingma, Diederik P. and Jimmy Ba (May 2015). "Adam: A Method for Stochastic Optimization". In: *Proc. of the Int. Conf. on Machine Learning*. San Diego, CA, USA (cit. on p. 42).

Martínez-Villaronga, A., M. A. del Agua, et al. (May 2013). "Language Model Adaptation for Video Lectures Transcription". In: *Proc. of ICASSP 2013*. Vancouver, Canada, pp. 8450–8454 (cit. on p. 38).

Martínez-Villaronga, A., M. A. del-Agua, et al. (Nov. 2014). "Language model adaptation for lecture transcription by document retrieval". In: *Proc. of IberSpeech 2014* (cit. on p. 38).

Meignier, Sylvain and Teva Merlin (Mar. 2010). "LIUM SpkDiarization: an open source toolkit for diarization". In: *Proc. of CMU SPUD Workshop 2010*. Dallas, TX, USA (cit. on p. 40).

Rybach, David et al. (Dec. 2011). "RASR - The RWTH Aachen University Open Source Speech Recognition Toolkit". In: *Proc. of ASRU 2011*. Honolulu, HI, USA (cit. on p. 41).

Siegler, Matthew A. et al. (1997). "Automatic Segmentation, Classification and Clustering of Broadcast News Audio". In: *Proc. of DARPA Speech Recognition Workshop 1997*, pp. 97–99 (cit. on p. 40).

Silvestre-Cerdà, Joan Albert et al. (Nov. 2012). "Albayzin Evaluation: The PRHLT-UPV Audio Segmentation System". In: *Proc. of IberSpeech 2012*. Madrid, Spain, pp. 596–600 (cit. on p. 40).

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958 (cit. on p. 42).

Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech 2002*, pp. 901–904 (cit. on p. 38).

Wiesler, Simon et al. (May 2014). "RASR/NN: The RWTH neural network toolkit for speech recognition". In: *Proc. of ICASSP 2014*. Florence, Italy, pp. 3313–3317 (cit. on p. 41).

Xiong, Wayne et al. (Apr. 2018). "The Microsoft 2017 Conversational Speech Recognition System". In: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. Calgary, Canada, pp. 5934–5938 (cit. on p. 35).

Young, S. J., J. J. Odell, and P. C. Woodland (1994). "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proc. of Workshop on Human Language Technology 1994*, pp. 307–312 (cit. on p. 36).

Zeyer, Albert, Tamer Alkhouli, and Hermann Ney (July 2018). "RETURNN as a Generic Flexible Neural Toolkit with Application to Translation and Speech Recognition". In: *Proc. of ACL*. Melbourne, Australia (cit. on p. 41).

2 Real-time one-pass decoder for speech recognition using LSTM language models

Jorge, Javier; Giménez, Adrià; Iranzo-Sánchez, Javier; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Proc. of the 20th Annual Conf. of the ISCA (Interspeech 2019), pp. 3820–3824

Graz (Austria)

DOI 10.21437/Interspeech.2019-2798

Core A - GGS A

15-19 September 2019

Real-time one-pass decoder for speech recognition using LSTM language models

Jorge, Javier; Giménez, Adrià; Iranzo-Sánchez, Javier; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Abstract

Recurrent Neural Networks, in particular Long Short-Term Memory (LSTM) networks, are widely used in Automatic Speech Recognition for language modelling during decoding, usually as a mechanism for rescoring hypothesis. This paper proposes a new architecture to perform real-time one-pass decoding using LSTM language models. To make decoding efficient, the estimation of look-ahead scores was accelerated by precomputing static look-ahead tables. These static tables were precomputed from a pruned n-gram model, reducing drastically the computational cost during decoding. Additionally, the LSTM language model evaluation was efficiently performed using Variance Regularization along with a strategy of lazy evaluation. The proposed one-pass decoder architecture was evaluated on the well-known LibriSpeech and TED-LIUMv3 datasets. Results showed that the proposed algorithm obtains very competitive WERs with ∼0.6 RTFs. Finally, our one-pass decoder is compared with a decoupled two-pass decoder.

2.1 Introduction

Recurrent Neural Networks (RNNs) and particularly Long Short-Term Memory (LSTM) networks are widely used to build Language Models (LMs) for Large-Vocabulary Continuous Speech Recognition (LVCSR) (Hochreiter and Schmidhuber 1997; Mikolov et al. 2010; Jozefowicz et al. 2016). An initial recognition step is first applied on the basis of an n-gram LM, from which a set of best hypotheses is produced (e.g. an N-best list or a lattice). Then, a second recognition step is carried out in which an LSTM-based LM is used for hypothesis rescoring (Kombrink et al. 2011; Si et al. 2013; Sundermeyer et al. 2014; Chen, Liu, et al. 2017; Xu et al. 2018; Ogawa et al. 2018). The use of this two-steps approach instead of a more direct, one-pass decoding, is needed to overcome the high computational cost associated with the LSTM-LM. That is, by applying an LSTM-LM to a limited set of best hypotheses, we take advantage of its high accuracy while keeping the decoding time under reasonable levels.

Although the above two-step approach is still the preferred way to develop fast yet accurate ASR systems, we think that it will be soon replaced by one-pass decoding alone, not only to avoid cascade errors but, more importantly, to leverage the full potential of state-of-the-art ASR at real-time. To our knowledge, the direct use of Neural LMs during decoding was first explored in (Shi et al. 2014), where the authors proposed the use of a Variance Regularization (VR) term along with caching strategies to speed-up this process. Despite using (feed-forward) Neural LMs in decoding early was seen as a big challenge at that time, empirical results showed significant relative improvements both in speed and accuracy. Other relevant contributions addressing this challenge have focused on applying heuristics to reduce model's queries and caching the network's states (Huang et al. 2014), proposing alternative one-pass decoding strategies such as on-the-fly rescoring (Hori, Kubo, and Nakamura 2014), improving CPU-GPU communications (K. Lee et al. 2015) and, more recently, combining Gated Recurrent Units with more efficient objective functions, such as Noise Contrastive Estimation (Kyungmin Lee et al. 2018). On the other hand, and certainly different from these contributions, other authors have explored the idea of converting Neural LMs, either recurrent or not, into n-gram models that can thus be smoothly integrated into the conventional recognition pipeline (Arisoy et al. 2014; Singh et al. 2017).

This work follows the idea of directly using Neural LMs in decoding and, as in the pioneering work by Shi et al. (Shi et al. 2014), we advocate the use of the one-pass decoding strategy instead of the conventional two-step approach. It is worth noting, however, that significant progress has been made in ASR since the publication of this pioneering work, and thus the work reported here, based on current ASR technology, differs greatly from it. Generally speaking, we propose the direct use of LSTM-based LMs during one-pass decoding based on a History Conditioned Search (HCS) strategy (Nolden 2017). To alleviate the computational cost entailed by the use of LSTM-LMs, three main ideas are exploited: static look-ahead tables; accelerated LSTM-LM computation by variance regularization and lazy evaluation; and two new pruning techniques. Results are reported on two standard tasks showing that these ideas are really useful for real-time one-pass decoding using LSTM-LMs.

2.2 One-pass decoder architecture

As previously mentioned, we propose the direct use of LSTM-LMs in one-pass HCS-based decoding. This proposal derives from the large capacity LSTM-LMs have, in contrast to n-gram LMs, to deal with histories of unlimited length (Kombrink et al. 2011). This makes HCS-based decoding perfectly suited for its use with LSTM-LMs, as HCS decoders group hypotheses by history, and thus large decoding sub-networks can be safely removed during search, thereby lowering memory requirements.

Although the HCS approach allows us to deal with potentially infinite LM histories during decoding, LSTM-LMs present other challenges which need to be overcome in order to get a real-time decoder. The most important one is the high computational cost required by these models. In particular, the calculation of the Softmax layer is very expensive for large vocabularies. In what follows we describe the solutions we implemented in our decoder to the problems we encountered.

Static look-ahead tables

LM look-ahead is a well-known and widespread pruning technique. Basically, this technique consists of adjusting the LM score for each hypothesis h and time t by also taking into account every possible word w to follow (Nolden 2017). In terms of computational cost, it requires a separate set of look-ahead scores, often referred to as the look-ahead table, for each new history h; that is, it requires the computation of $p(w \mid h)$ for each history h and word w. Moreover, the cost of this computation, which is already high for conventional n-gram LMs, is even exacerbated when LSTM-LMs are used instead.

A common technique for HCS decoders to keep the look-ahead complexity at a reasonable level is to build look-ahead tables from simplified LMs; that is, if an n-gram LM is being used as the "big" reference LM, look-ahead tables are built from m-gram LMs with $m < n$. The only exception is that, whenever a word-end node is reached, the look-ahead score is replaced by the probability given by the big LM. This way, the number of different look-ahead tables and queries to the big LM are greatly reduced, which is particularly convenient for our LSTM-based one-pass decoder. That is, for fast computation of look-ahead tables, we propose the use of n-gram LMs.

Although the above trick for fast computation of look-ahead tables is really effective, it is worth noting that it can be refined even further, in a straightforward manner. To this end, consider, as we do here, the use of a small n-gram LM such as a pruned 4-gram LM. Then, all look-ahead tables can be precalculated before the actual recognition process begins, and thus the look-ahead complexity during decoding becomes negligible. This is done here, and in order to fit all look-ahead tables in memory, we use an approach similar to the Sparse LM Look-Ahead strategy described in (Nolden 2017).

Variance regularization and lazy evaluation

As commented before, one of the main drawbacks of using LSTM-LMs is the high computational cost of the Softmax layer. This high cost is mainly due to the estimation of the normalization term. Following the idea posed in (Shi et al. 2014), including a Variance Regularization (VR) term reduces drastically this computation. In this technique the normalization term is approximated by a constant. Therefore, the probability of a word can be approximated as

$$p(w \mid h) = \frac{\exp(v_L(h)^T \cdot a_w)}{Z(h)} \approx \frac{\exp(v_L(h)^T \cdot a_w)}{D}\,, \tag{2.1}$$

where h denotes the current history, L is the number of hidden layers, $v_L(h)$ and $Z(h)$ are respectively the input vector to the Softmax layer and the normalization term related to h, a_w is the weight vector for word w, and D is a constant.

In order to speed up the evaluation of the LSTM-LM during decoding, the models were trained based on the VR technique in conjunction with a lazy evaluation strategy in the decoding process. The basic idea behind the VR technique is to avoid computing the full Softmax, while the lazy strategy delays the evaluation of LSTM-LM as much as possible. More precisely, during decoding each LSTM history h is represented as a tuple $h = (w, h', V)$, where w is the last word of h, $h' = (w', h'', V')$ is the state of the previous history (implemented as a pointer), and V is either $\emptyset$ (empty) or $v_1^L(h)$. The term $v_1^L(h)$ refers to the LSTM hidden state for h and can be calculated as $v_1^L(h) = \mathrm{RNN}(w, v_1^L(h'))$. Therefore, during decoding each time a word-end node (w) is reached for a given history $h' = (w', h'', V')$ the following steps are executed:

1. If $V' = \emptyset$ then $V' = \mathrm{RNN}(w', v_1^L(h''))$
2. Estimate $p(w \mid h') = \frac{\exp(v_L(h')^T \cdot a_w)}{D}$
3. Create new state as $h = (w, h', \emptyset)$

Using this approach new histories are created at negligible cost, since the forward step in the model is carried out only when a word-end node is reached for the first time. Once the hidden state is calculated, it is cached in the current state. In practice most of these new histories will be pruned before any hypothesis reaches a word-end node, saving a significant amount of computations. In addition, each time $p(w \mid h')$ is required, it is approximated using Eq. 2.1 that replaces the full Softmax calculation.

Novel pruning techniques

Apart from the conventional pruning methods, two new pruning techniques were implemented for the one-pass decoder. On the one hand, in some situations the lack of LM history recombination in conjunction with the histogram pruning (upper bound for the maximum number of active hypothesis at each time frame) resulted in a decrease of performance. More precisely, for some long sentences, most part of the active hypotheses were similar, except for some long term differences in the LM history. Thus, in this case, the decoder behaves similarly to a greedy decoder. In order to avoid this behaviour a LM history recombination (LMHR) parameter was introduced. More precisely, for a given N two different LM histories w_1^M and $\hat{w}_1^L$ are recombined if $w_{M-N+1}^M = \hat{w}_{L-N+1}^L$. This recombination forces the decoder to consolidate word prefixes, and thus, it focuses on the current time frame. It is worth noting that the history length for the LSTM-LM is not limited. Each hypothesis still keeps a reference to the real LSTM state. A similar technique was introduced in (Huang et al. 2014), although the motivation was different.

Furthermore, the maximum number of new LM histories that can be created at each time frame is limited according to a parameter that we refer to as LM histogram pruning (LMHP).

Additional remarks

Although our approach is based on an HCS decoder, note that it could be reinterpreted as an on-the-fly composition as other authors have proposed in the past (Hori et al. 2007; Sak et al. 2010). Indeed, static look-ahead tables resembles the WFST approach, since the information stored in the set of look-ahead tables is similar to that represented in a WFST. Thus, the memory requirements are alike and the most important difference is that in the look-ahead tables the information is organized according to the LM histories.

Regarding this interpretation, this could be seen as performing the composition of a WFST (look-ahead tables) with a LSTM-LM. Nevertheless, it is important to remark that we are still using an HCS decoder. Thus, the structure of the small LM used for look-ahead is not introducing any kind of hypothesis recombination during the search. Its impact is limited to the look-ahead score computation.

Since our approach can be referred to as an on-the-fly composition of a small n-gram model and a big LM, it is possible to use other types of LMs rather than LSTMs. Indeed, an on-the-fly interpolation of n-gram and LSTM-LMs was implemented.

2.3 Experiments

Experimental settings

The proposed approach has been evaluated on the LibriSpeech ASR corpus (Panayotov et al. 2015), and the third version of the TED-LIUM corpus (Hernandez et al. 2018). Statistics for these datasets are shown in Table 2.8. The provided vocabulary for LibriSpeech includes 200K words, while TED-LIUM's vocabulary comprises 153K. Regarding the partitions, we have used the **-other* for LibriSpeech and the **-legacy* ones for TED-LIUM.

Table 2.8: Statistics of the corpora.

	LibriSpeech		TED-LIUM	
	Dur.(h)	Words	Dur.(h)	Words
Train	961	884M	452	258M
Dev	5.3	50K	1.59	17K
Test	5.1	52K	2.61	27K

Our acoustic models were based on the hybrid approach (Bourlard and Wellekens 1989). First, we trained a context-dependent feed-forward DNN-HMM with three left-to-right states. State-tying schema follows a phonetic decision tree approach (S. J. Young, J. J. Odell, and Woodland 1994), resulting in 8.3K and 10.8K tied states for LibriSpeech and TED-LIUM respectively. We used the transLectures-UPV toolkit (TLK) to train both acoustic models (del-Agua et al. 2014).

The DNN-HMM model was then used to bootstrap a Bidirectional LSTM-HMM model (Zeyer et al. 2017), using TLK and TensorFlow (Abadi, Agarwal, et al. 2015). The BLSTM network was composed of eight bidirectional hidden layers with 512 LSTM cells per layer and per direction. We limited the previous history to perform back propagation through time to a window size of 50 frames.

Regarding language modelling, we used n-gram and LSTM-LMs separately and in combination through linear interpolation. For LibriSpeech, we used the 4-gram ARPA LM (*fglarge*) that is provided with the dataset, while for TED-LIUM we trained a standard Kneser-Ney smoothed 4-gram LM model with the same data as it is indicated in (Hernandez et al. 2018) using SRILM (Stolcke 2002). The OOV ratio on dev sets was 0.57% and 0.17%, while was 0.54% and 0.09% on test sets, for LibriSpeech and TED-LIUM respectively. A pruned version for both models was used to estimate the static look-ahead tables.

LSTM-LMs were trained using Noise Contrastive Estimation (NCE) (Mnih and Teh 2012) and VR (Shi et al. 2014) criterion, in order to reduce the computational cost during both, training and test phases. Training was performed on GPU using the CUED-RNNLM toolkit (Chen, Liu, et al. 2016). We selected those models

that provided the lowest perplexity on the dev sets: *dev-other* and *dev-legacy* for LibriSpeech and TED-LIUM, respectively. Both models consisted of a 256-unit embedding layer and a hidden LSTM layer of 1024 units. The output layer is a 200K units Softmax layer in the case of LibriSpeech, while in TED-LIUM the intersection between the provided vocabulary and words in the training set resulted in an output layer of 144K units. As our decoder can combine any number of models during decoding, an offline evaluation of the best linear interpolation of both n-gram and LSTMs using SRILM was performed. The interpolation weights were $w_{ngram} = 0.15, w_{lstm} = 0.85$ for LibriSpeech and $w_{ngram} = 0.22, w_{lstm} = 0.78$ for TED-LIUM.

Regarding the hardware setup, experiments were conducted on an Intel Xeon(R) CPU E5-1620@3.50GHz, and a GPU GTX1080Ti with 12GB. The estimation of the acoustic model scores was performed on GPU, while the estimation of the LM score and the rest of the decoding was carried out on CPU.

Experimental results

In this section, the impact of the LMHR parameter presented in Section 2.2 is evaluated. After this, a comparative experiment is shown using different LM during decoding. Then, the LMHP parameter is studied to assess the trade-off between Word Error Rate (WER) and Real-Time Factor (RTF). Finally, a comparison between one-pass and two-pass decoders is performed.

Figure 2.1 shows WER curves as a function of the LMHR parameter for the dev sets of LibriSpeech and TED-LIUM. As mentioned before, the LMHR controls the number of words that were kept during the decoding to perform hypothesis recombination. The LM employed in these figures is a linear interpolation between a large 4-gram LM and a VR-trained LSTM-LM described in Section 2.3.

As observed in both datasets, there is an optimum WER at around a history length of 10. This means a decrease of 0.5 and 0.3 WER points in LibriSpeech and TED-LIUM, respectively, w.r.t. a history length of 2. History lengths above 20 did not provide further improvements. In what remains, we have determined the optimal LMHR value for each dataset of 10.

Regarding the impact of using different LMs, Table 2.9 shows WER figures, relative improvement of WER ($\Delta\%$) and perplexity (PPL) for LibriSpeech and TED-LIUM test sets on systems that differ in their LM. From top to bottom, the baseline LM is the pruned (small) 4-gram LM used to perform look-ahead, then the large 4-gram LM, next the VR-trained LSTM-LM and finally the interpolated LM mentioned above. Decoding hyperparameters were tuned on the dev set.

As shown in Table 2.9, one-pass decoding systems including LSTM-LM present relative improvements of $\sim$17% for LibriSpeech and $\sim$13% for TED-LIUM, compared with large 4-gram-based systems. In addition, interpolated LM systems

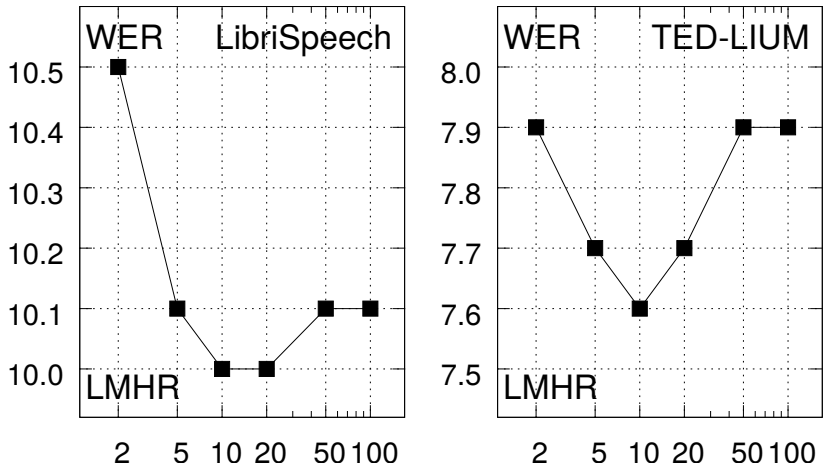

Figure 2.1: Evaluation of the impact of the LMHR parameter in terms of WER [%] for LibriSpeech and TED-LIUM.

provide an additional ∼1% improvement over only-LSTM LM systems. In terms of perplexity w.r.t. large 4-gram LMs, interpolated LMs offer a significant reduction of ∼40% across datasets, stressing the consistent relation between perplexity and WER. This interpolated LM was the default model for posterior experiments.

Table 2.9: Comparison of WER, relative improvement of WER w.r.t the baseline, and perplexity results using different LM models on LibriSpeech and TED-LIUM test partitions.

	LibriSpeech			TED-LIUM		
	WER	Δ%	PPL	WER	Δ%	PPL
small 4-gram	14.4	-	222.1	10.4	-	176.4
+ 4-gram	12.3	14.6	146.2	9.6	7.7	148.7
+ LSTM	10.2	29.2	89.2	8.3	20.2	91.1
+ interp.	10.1	29.9	86.4	8.2	21.1	88.0

Figure 2.2 shows WER/RTF curves for LibriSpeech and TED-LIUM dev sets as a function of several selected values for the LMHP parameter (LMHP = 30, 60, 100, *unlimited*), varying the beam width. As remarked in Section 2.2, LMHP allows us to control the number of new LM histories that will be expanded, reducing the queries to the LSTM-LM.

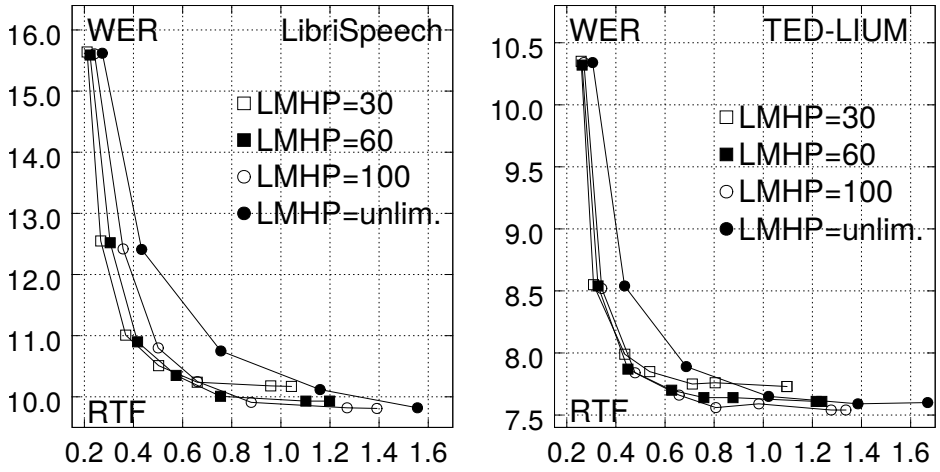

Figure 2.2: Comparison of the impact of different values for the *LM-histogram-pruning* (LMHP) parameter in terms of WER and RTF for LibriSpeech and TED-LIUM.

As shown in Figure 2.2, the LMHP parameter has a strong impact in RTF, allowing us to adjust the WER/RTF trade-off on demand. To this purpose, real-time performance (RTF$\sim$1) can be reached at almost no cost in terms of WER, as differences among LMHP values show. In particular, limiting LMHP to 100 allows us to obtain RTF results below 1 without WER reduction in both datasets.

Table 2.10 shows WER figures, RTF and relative decrease of RTF ($\Delta\%$) on LibriSpeech and TED-LIUM test sets comparing the best performing LMHP value and *unlimited* using the same pruning parameters. As observed, RTF was drastically reduced while maintaining similar WER figures.

Table 2.10: Comparison of WER, RTF and relative increase of RTF w.r.t to LMHP=100 on LibriSpeech and TED-LIUM test sets.

	LibriSpeech			TED-LIUM		
LMHP	WER	RTF	$\Delta\%$	WER	RTF	$\Delta\%$
unlimited	10.12	1.56	-	8.18	1.38	-
100	10.20	0.88	43.6	8.19	0.87	37.0

As mentioned in Section 2.2, our decoder can be considered as an on-the-fly composition of the pruned LM and the LSTM/n-gram interpolation. This motivates another set of experiments that assesses the impact of performing this on-the-fly composition in contrast to the decoupled approach. For the decoupled approach, the standard two-pass decoding strategy was adopted based on generating lat-

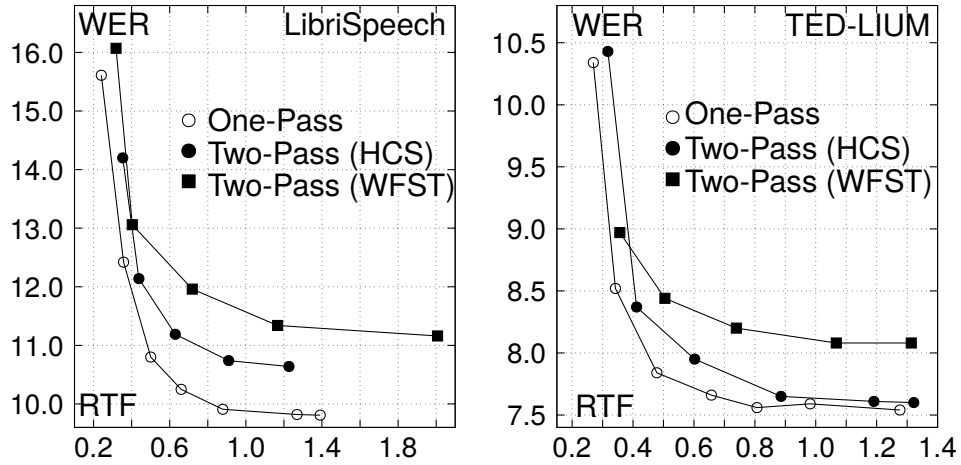

Figure 2.3: Comparison of the one-pass HCS decoder and the two-pass HCS and WFST decoders in terms of WER and RTF for LibriSpeech (left) and TED-LIUM (right).

tices and then rescoring them with the LSTM/n-gram interpolated LM. For lattice generation two options were considered: using the pruned (small) n-gram model, already used for static look-ahead tables estimation, and the large LM. In the case of the small n-gram, a WFST decoder was used. In the second option, the large LM was used in our HCS decoder replacing the LSTM-LM to generate lattices. The posterior lattice rescoring was carried out in both cases using the CUED-RNNLM toolkit, which imports this function from HTK (S. Young et al. 2002). It is important to remark that this software was extended to include the VR normalization. The same acoustic and language models were used in both cases.

Figure 2.3 compares the WER/RTF curves for one-pass using LMHR=10 and LMHP=100 versus both two-pass approaches on LibriSpeech (left) and TED-LIUM (right) datasets.

Results show that the one-pass decoder produces significant improvements in WER compared to WFST, especially when RTF is greater than 0.4. Considering a very similar RTF performance, the one-pass decoding approach achieves relative improvements in WER $\sim$12% and $\sim$6% in LibriSpeech and TED-LIUM, respectively. Comparing HCS decoders, one-pass shows a consistent improvement in RTF, reducing WER ($\sim$6%) in the case of LibriSpeech and obtaining a similar accuracy in TED-LIUM.

Table 2.11 shows WER figures for similar RTFs on test partitions, considering the aforementioned decoders, one-pass and two-pass HCS, and the two-pass WFST

decoder. As shown in development sets, the one-pass HCS significantly improves over the WFST approach. Finally. the one-pass HCS obtains a better WER/RTF trade-off than the two-pass HCS.

Table 2.11: Comparison of WER for similar RTFs between one-pass HCS decoder and the two-pass HCS and WFST decoders on LibriSpeech and TED-LIUM test partitions.

	LibriSpeech		TED-LIUM	
	WER	RTF	WER	RTF
one-pass HCS	10.20	0.88	8.19	0.87
two-pass HCS	10.95	0.90	8.46	0.88
two-pass WFST	11.39	1.16	8.70	1.06

2.4 Conclusions and future work

A novel one-pass decoder that seamlessly combines the use of static look-ahead tables and LSTM-LMs has been presented. This decoder has been evaluated on reference ASR datasets, such as LibriSpeech and TED-LIUM, obtaining competitive WER/RTF results. Indeed, RTF figures are reported well below one that makes this decoder specially suitable for a real-time streaming setup.

Moreover, two new pruning parameters, LMHR and LMHP, were introduced allowing us to adjust the trade-off between WER and RTF according to our requirements. In addition, the one-pass and two-pass decoders were directly compared to demonstrate how the one-pass decoder clearly benefits from the integration of all LMs in the first stage of the decoding process.

As future work, the current decoder will be evaluated on a real streaming scenario providing recognition hypothesis as the audio signal data is ingested. In addition, the current LSTM-LM evaluation will be moved from CPU to GPU in order to further alleviate the computational cost.

References

Abadi, Martín, Ashish Agarwal, et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (cit. on p. 52).

Arisoy, E. et al. (2014). "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1, pp. 184–192 (cit. on p. 48).

Bourlard, H. and C. J. Wellekens (1989). "Links between Markov Models and Multilayer Perceptrons". In: *Advances in Neural Information Processing Systems I*, pp. 502–510 (cit. on p. 52).

Chen, Xie, Xunying Liu, et al. (2016). "CUED-RNNLM – An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *Proc. of ICASSP 2016*. Shanghai, China, pp. 6000–6004 (cit. on p. 52).

Chen, Xie, Xunying Liu, et al. (2017). "Future word contexts in neural network language models". In: *Proc. of ASRU 2017*, pp. 97–103 (cit. on p. 47).

del-Agua, M.A. et al. (Nov. 2014). "The translectures-UPV toolkit". In: *Proc. of Advances in Speech and Language Technologies for Iberian Languages 2014*, pp. 269–278 (cit. on p. 52).

Hernandez, François et al. (2018). "TED-LIUM 3: Twice as Much Data and Corpus Repartition for Experiments on Speaker Adaptation". In: *Proc. of SPECOM 2018*, p. 198 (cit. on p. 52).

Hochreiter, S. and J. Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 47).

Hori, Takaaki et al. (2007). "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, pp. 1352–1365 (cit. on p. 51).

Hori, Takaaki, Yotaro Kubo, and Atsushi Nakamura (2014). "Real-time one-pass decoding with recurrent neural network language model for speech recognition". In: *Proc. of ICASSP 2014*. IEEE, pp. 6364–6368 (cit. on p. 48).

Huang, Z. et al. (2014). "Cache based recurrent neural network language model inference for first pass speech recognition". In: *Proc. of ICASSP 2014*, pp. 6354–6358 (cit. on pp. 48, 51).

Jozefowicz, R. et al. (2016). "Exploring the limits of language modeling". In: *arXiv preprint arXiv:1602.02410* (cit. on p. 47).

Kombrink, Stefan et al. (2011). "Recurrent Neural Network based language modeling in meeting recognition". In: *Proc. of Interspeech 2011*, pp. 2877–2880 (cit. on pp. 47, 48).

Lee, K. et al. (2015). "Applying GPGPU to recurrent neural network language model based fast network search in the real-time LVCSR". In: *Proc. of InterSpeech 2015*, pp. 2102–2106 (cit. on p. 48).

Lee, Kyungmin et al. (2018). "Accelerating recurrent neural network language model based online speech recognition system". In: *Proc. of ICASSP 2018*, pp. 5904–5908 (cit. on p. 48).

Mikolov, Tomáš et al. (2010). "Recurrent neural network based language model". In: *Proc. of InterSpeech 2010*. ISCA, pp. 1045–1048 (cit. on p. 47).

Mnih, Andriy and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models". In: *Proc. of ICML* (cit. on p. 52).

Nolden, David (Apr. 2017). "Progress in Decoding for Large Vocabulary Continuous Speech Recognition". PhD thesis. Computer Science Department RWTH Aachen University Aachen (Germany): RWTH Aachen University (cit. on pp. 48, 49).

Ogawa, A. et al. (2018). "Rescoring N-Best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model". In: *Proc. of ICASSP 2018*, pp. 6099–6103 (cit. on p. 47).

Panayotov, V. et al. (2015). “Librispeech: an ASR corpus based on public domain audio books”. In: *Proc. of ICASSP 2015*, pp. 5206–5210 (cit. on p. 52).

Sak, Haşim et al. (2010). “On-the-fly lattice rescoring for real-time automatic speech recognition”. In: *Proc. of the international speech communication association 2010* (cit. on p. 51).

Shi, Yongzhe et al. (2014). “Efficient One-Pass Decoding with NNLM for Speech Recognition”. In: *IEEE Signal Processing Letters* 21.4, pp. 377–381 (cit. on pp. 48, 50, 52).

Si, Yujing et al. (2013). “Prefix tree based n-best list re-scoring for recurrent neural network language model used in speech recognition system.” In: *Proc. of InterSpeech 2013*, pp. 3419–3423 (cit. on p. 47).

Singh, Mittul et al. (2017). “Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition”. In: *Proc. of InterSpeech 2017*, pp. 2720–2724 (cit. on p. 48).

Stolcke, Andreas (2002). “SRILM - an extensible language modeling toolkit.” In: *Proc. of Interspeech 2002*, pp. 901–904 (cit. on p. 52).

Sundermeyer, Martin et al. (2014). “Lattice decoding and rescoring with long-span neural network language models”. In: *Proc. of the InterSpeech 2014* (cit. on p. 47).

Xu, H. et al. (2018). “A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition”. In: *Proc. of ICASSP 2018*, pp. 5929–5933 (cit. on p. 47).

Young, S. J., J. J. Odell, and P. C. Woodland (1994). “Tree-based State Tying for High Accuracy Acoustic Modelling”. In: *Proc. of Workshop on Human Language Technology 1994*, pp. 307–312 (cit. on p. 52).

Young, Steve et al. (2002). “The HTK book”. In: *Cambridge university engineering department* 3, p. 175 (cit. on p. 56).

Zeyer, Albert et al. (2017). “A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition”. In: *Proc. of ICASSP 2017*. IEEE, pp. 2462–2466 (cit. on p. 52).

3 LSTM-based one-pass decoder for low latency streaming

Jorge, Javier; Giménez, Adrià; Iranzo-Sánchez, Javier; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Proc. of 45th Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2020), pp. 7814–7818

Barcelona (Spain)

DOI 10.1109/ICASSP40776.2020.9054267

GGS A

4-8 May 2020

LSTM-based one-pass decoder for low latency streaming

Jorge, Javier; Giménez, Adrià; Iranzo-Sánchez, Javier; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Abstract

Current state-of-the-art models based on Long Short-Term Memory (LSTM) networks have been extensively used in ASR to improve performance. However, using LSTMs under a streaming setup is not straightforward due to real-time constraints. In this paper we present a novel streaming decoder that includes a bidirectional LSTM acoustic model as well as an unidirectional LSTM language model to perform the decoding efficiently while keeping the performance comparable to that of an off-line setup. We perform a one-pass decoding using a sliding window scheme for a bidirectional LSTM acoustic model and an LSTM language model. This has been implemented and assessed under a pure streaming setup, and deployed into our production systems. We report WER and latency figures for the well-known LibriSpeech and TED-LIUM tasks, obtaining competitive WER results with low-latency responses.

3.1 Introduction

On-line or streaming automatic speech recognition (ASR) poses additional challenges to the off-line setup when state-of-the-art neural-based models are involved. The main practical challenge is that speech recognition must be performed under real-time constraints as the audio stream becomes available. This implies that the complete audio stream is not fully available when decoding is performed up to a certain point in time. This constraint, combined with the essential requirement of low latencies, is especially demanding when state-of-the-art, neural-based acoustic and language models are used.

On the one hand, acoustic modeling based on Bidirectional Long Short-Term Memory (BLSTM) networks has shown to be the current state-of-the-art in off-line ASR systems (Graves and Schmidhuber 2005). However, BLSTM network training requires to consider some future (right) context with respect to the current frame to compute its acoustic score. This means that the output of the ASR

system must be delayed up to the point in which *enough* frames from the future have become available for decoding. In this sense, to facilitate streaming ASR, an in-depth study of using a sliding window that captures the left (past) and right (future) context with respect to the current frame was carried out in (Mohamed et al. 2015). models outperform deep neural network (DNN) approaches in the streaming setup. This work was extended in (A. Zeyer, Schlüter, and Ney 2016) with a definition of a theoretical framework for the sliding window approach and a weighting scheme for overlapping frames.

On the other hand, language modeling based on LSTMs networks has become the dominant approach, providing the best results so far in many ASR tasks (Jozefowicz et al. 2016). While these models are commonly used as part of the rescoring process using the n-best hypothesis or lattices (Xie Chen, Liu, et al. 2017; Ogawa et al. 2018; Kombrink et al. 2011), there is an increasing interest in providing an efficient way to integrate LSTM-based language models (LMs) into the decoding process, thus enabling their use in streaming ASR. Indeed, many approaches have been proposed in this regard, such as converting recurrent models into n-grams (Arisoy et al. 2014), using cache strategies (Huang et al. 2014), and reducing the computational complexity of the Softmax function (Shi et al. 2014). It is worth noting, however, that these approaches were only focused on language modeling, that is, the decoding problem was not considered as a whole for (low-latency) streaming.

Recently, we proposed the use of LSTM-based LMs in the first pass of the decoding process, without any approximation or transformation in (Jorge et al. 2019). This is achieved by keeping the state of the LSTM LM network as part of the search structure, along with the Variance Regularization term (Shi et al. 2014) used to reduce the Softmax computational complexity. This allowed us to achieve a competitive performance while keeping the real time factor (RTF) below one. In this way, we paved the way to the use of LSTM LMs in a streaming setup, in which tight real-time constraints and low latency are critical. This is done in this work, that is, we extend the one-pass decoding approach described in (Jorge et al. 2019) by using LSTM LMs in a streaming setup. As in (A. Zeyer, Schlüter, and Ney 2016), acoustic modeling is based on BLSTMs, though in this work important requirements for streaming ASR are also considered, such as the normalization of the input features or the system architecture. More generally, this work can be seen as a thorough study of the effect that moving from an off-line to a streaming setup has on WER. It is shown that it highly depends on the feature normalization context and the availability of future acoustic context. In addition, given the tight real-time constraints of the streaming setup, detailed results on time latency are reported on reference tasks widely used in the literature such as LibriSpeech (Panayotov et al. 2015) and TED-LIUM (Hernandez et al. 2018).

3.2 Streaming decoder

One-pass decoder review

We followed the decoder structure proposed in (Jorge et al. 2019). The main aspects of this decoder will be commented briefly in this section. In this decoder, hypotheses are organized by their history, in a similar way as it is done in (Nolden 2017).

Therefore, the computation of the lookahead score should be carried out dynamically during decoding, involving several queries to the language model. To reduce the impact of that, in this approach it is proposed the use of the static lookahead tables, a structure that is precomputed in advance, providing the lookahead score efficiently during decoding. This structure is obtained from a pruned n-gram model, enabling the use of less memory during recognition.

On the other hand, when a word-end node is reached, we replaced the score that we got so far by the score from the LSTM LM. In order to compute this efficiently, we used the Variance Regularization term as a self-normalized function, avoiding the computation of the whole Softmax (Shi et al. 2014). Along with this function, we followed a lazy strategy, when a language model node is created, to postpone the computation of the LM score as much as possible. A more detailed description of these steps can be found in (Jorge et al. 2019).

We included two additional parameters in this one-pass decoder to control the WER/RTF trade-off: the Language Model Histogram Recombination (LMHR) and the Language Model Histogram Pruning (LMHP). The LMHR controls the size of the previous history considered to combine two hypotheses. The LMHP limits the number of new LM histories to be expanded, and thus the number of queries to the LSTM LM. As shown in (Jorge et al. 2019), these parameters are really effective to reduce the RTF while keeping a good WER performance.

Streaming adaptation

Under streaming conditions, we cannot see the complete context for a given frame. Therefore, when using BLSTM networks for acoustic modeling, we have to assume a delay between the input and the output, that allows us to consider this temporal gap as a short-term context in order to obtain the acoustic score. In order to work with this acoustic lookahead, we have followed a similar strategy to (A. Zeyer, Schlüter, and Ney 2016). It is based on the use of a sliding window over the sequence, where for each frame, we have a context of the $n_{\text{lookahead}}$ following seconds. These seconds, or equivalently, frames, were used in order to compute the forward and the backward steps over the sequence, obtaining a score for each frame within the window, using a BLSTM network as in (Mohamed et al. 2015). We have done this frame by frame, meaning that there will be an overlap equal

to the size of the window. Regarding the overlapping frames, we used a uniform weighted average of the acoustic scores to obtain the final score that will be provided to the decoder. In the extreme cases where the utterance is shorter than the window or the window goes beyond the sequence, we introduced zero padding up to the length of the window.

Regarding normalization of the features, we have included an initial delay that will be used to gather statistics to initialize the mean and the variance. We have a parameter, n_{norm}, that indicates the number of seconds that will be used to compute the statistics. Once these n_{norm} seconds have been accumulated, the normalization will be applied from the first frame to the last delayed frame, and then the recognition starts updating mean and variance frame by frame without including any additional delay. There is, indeed, an initial latency to obtain the first result from the decoder of n_{norm} seconds. However, in a real streaming setup, if this initial delay is small, it will not harm the global latency nor the performance of the system.

Our streaming setup follows a client-server architecture, where the server can process different requests at the same time.

In order to manage that, in the server side, we have a pool of recognizers that will be paired with clients, serving requests in a separated way, while the recognizers share the same models, reducing the required CPU and GPU memory. This allowed us to accommodate different systems at the same time. This architecture is used in our TTP platform[1], currently available under registration, in English, Spanish and Catalan, and in the PoliSubs service[2] provided by the Universitat Politècnica de València.

3.3 Experiments

Experimental setup

Table 2.12 provides some basic statistics of the corpora used to assess our approach: corpus (Panayotov et al. 2015), TED-LIUM corpus (Hernandez et al. 2018). we have used the provided 200K words for LibriSpeech, and for TED-LIUM's we have selected 153K words. Regarding the partitions, we have used the **-other* for LibriSpeech and the **-legacy* ones for TED-LIUM.

Following the hybrid approach (Bourlard and Wellekens 1989), we trained a context-dependent feed-forward DNN-HMM with three left-to-right states. We have obtained 8.3K and 10.8K tied states (senones) for LibriSpeech and TED-LIUM respectively after applying a phonetic decision tree (Young, Odell, and

[1] `https://ttp.mllp.upv.es/`
[2] `https://apps.upv.es/` - `https://polisubs.upv.es/`

Table 2.12: Statistics of the corpora.

	LibriSpeech		TED-LIUM	
	Dur.(h)	Words	Dur.(h)	Words
Train	961	884M	452	258M
Dev	5.3	50K	1.59	17K
Test	5.1	52K	2.61	27K

Woodland 1994), following a subphonetic modeling. To train our systems we have used the transLectures-UPV toolkit (TLK) (del-Agua et al. 2014).

We bootstrapped a bidirectional LSTM-HMM model using the previous DNN-HMM, as in (Albert Zeyer et al. 2017). For the BLSTM model, we used both TLK and TensorFlow (Abadi, Agarwal, et al. 2015), and an architecture of eight bidirectional hidden layers with 512 LSTM cells per layer and direction. Following (Albert Zeyer et al. 2017), we performed chunking during training by considering a context to perform back propagation through time to a window size of 50 frames.

On the language modeling side, we used n-grams and LSTM LMs, combining them through linear interpolation. Apart from the 4-gram model provided for LibriSpeech, we trained a 4-gram Kneser-Ney smoothed LM for TED-LIUM, using the same data as in (Hernandez et al. 2018) and SRILM (Stolcke 2002). We obtained OOV ratios of 0.57% and 0.12% for LibriSpeech and TED-LIUM, respectively, on the dev sets. To compute the static lookahead tables, a pruned version of these n-gram models was computed.

The CUED-RNNLM toolkit (Xie Chen, Liu, et al. 2016) was used to train LSTM LMs. Noise Contrastive Estimation (NCE) criterion (Mnih and Teh 2012) was used to train faster, and the normalization constant learnt from training was used during recognition (X. Chen et al. 2015). Based on the lowest perplexity models on dev, we selected the final models with 256-unit embedding layer and a hidden LSTM layer of 1024 units. Finally, the interpolation weights to combine the n-gram and the LSTM LM were $w_{ngram} = 0.15, w_{lstm} = 0.85$ for LibriSpeech and $w_{ngram} = 0.22, w_{lstm} = 0.78$ for TED-LIUM. Table 2.13 shows the perplexity on the dev partitions of these LM.

Table 2.13: Perplexity results on development partitions.

	LibriSpeech	TED-LIUM
4-gram	140.6	125.8
LSTM LM	86.2	88.0
LSTM LM + 4-gram int.	83.7	78.7

It is worth noting that we used the same models and pruning parameters for both the off-line and the streaming approach, in order to perform a fair comparison

between the two setups. Following the experimental analysis in (Jorge et al. 2019), we used the values of $LMHR = 10$ and $LMHP = 100$ for the one-pass decoder to provide best results in the development sets for both datasets. We carried out the computations related to both acoustic and language models in GPU, whereas the decoder was run in CPU, using a Intel Xeon(R) CPU E5-1620@3.50GHz with a GTX1080Ti with 12GB.

Assessment of normalization and lookahead contexts

This section is to assess the impact of the context for normalization (n_{norm}) and the impact of the lookahead context ($n_{\text{lookahead}}$) in terms of WER.

Table 2.14 shows the WER on the dev partition, for LibriSpeech and TED-LIUM, as a function of n_{norm} (in secs.). As discussed in Section 3.2, n_{norm} indicates the number of seconds that the system is allowed to compute statistics for feature normalization. For the results in this Table, we considered a value $n_{\text{lookahead}} = 0.5$ seconds, which is the value we used for chunking during training.

Table 2.14: Impact of normalization context on WER, on LibriSpeech and TED-LIUM.

n_{norm} (sec)	*LibriSpeech*	*TED-LIUM*
0	15.6	9.7
1	11.0	8.2
2	10.0	8.1
4	9.6	7.9
8	9.4	7.7
∞	9.4	7.6

From the results in Table 2.14, it is clear that normalization helps in improving performance. As expected, the best WER is achieved when the whole utterance is considered. Indeed, broadly speaking, with more information, the mean and variance are better estimated. However, it goes without saying an optimal value for n_{norm} should be in between extreme values. On our experience, an appropriate value for this parameter could be around 2 seconds. To us, this value is a sort of minimum for the system to collect meaningful statistics and respond after a reasonable waiting time.

Once this initial delay for the normalization is fixed to 2 seconds, we have performed an analysis of the impact on the WER of the parameter $n_{\text{lookahead}}$, which indicates the number of seconds of acoustic lookahead. Table 2.15 summarizes the results for these experiments, with the WER obtained for each corpus considering $\{0.125, 0.25, 0.5, 1, 2\}$ seconds of $n_{\text{lookahead}}$.

Results shown that the best WER is obtained with a 1 second and 2 seconds window length for LibriSpeech and TED-LIUM, respectively. While for TED-LIUM

Table 2.15: Impact of lookahead context on LibriSpeech and TED-LIUM on WER.

$n_{\text{lookahead}}$ (sec)	*LibriSpeech*	*TED-LIUM*
0.125	17.1	10.5
0.250	11.6	8.8
0.500	10.0	8.1
1.000	9.9	7.9
2.000	10.2	7.8

using the biggest considered size helped the performance, there is a degradation on WER for LibriSpeech, which could be due to the additional padding that we had to introduce for the small segments, very common in this dataset. In addition, results shown that there is an important gap in performance when using more than 0.250 seconds to compute the acoustic score, as a WER improvement of $\sim 14\%$ and $\sim 8\%$ could be achieved using 0.5 seconds to compute the sliding window, for LibriSpeech and TED-LIUM, respectively.

Latency assessment

Table 2.16 shows the average latency for LibriSpeech and TED-LIUM, as a function of $n_{\text{lookahead}}$ and n_{norm}. Here, latency refers to time elapsed between the end of the current hypothesis and the point in time at which it is actually delivered. So, for instance, if a hypothesis is delivered 2 seconds after its (acoustic) end, then the latency is 2 seconds. In contrast to Table 2.15, where WER is studied as a function of $n_{\text{lookahead}}$, here we focus on the average latency as a function of $n_{\text{lookahead}}$ and two values for n_{norm}, 2 and 0 seconds, with 0 seconds meaning that no time is devoted to gather statistics before recognition starts. While the setup with $n_{\text{norm}} = 2''$ was the one that we used to compute the WER in the previous Section, the results with $n_{\text{norm}} = 0''$ reflects the case in which sequences are long enough for normalization not to harm the global latency. Clearly, this gives an idea of how the decoder behaves in a real streaming setup.

It is worth noting that these measurements were taken in the server side, not considering the network latency.

As can be seen in Table 2.16, the results for TED-LIUM are really good. In the TED-LIUM task, we get small latencies for both $n_{\text{norm}} = 0''$ and $n_{\text{norm}} = 2''$, up to $n_{\text{lookahead}} = 0.5$, from which we get reasonable WER figures. In the LibriSpeech task, however, a slight degradation of the average latency is observed, especially for $n_{\text{norm}} = 2''$. This is to the relative shorter in LibriSpeech ($\sim$6.5 seconds), as compared to TED-LIUM ($\sim$11.3 seconds), which results in the decoder not being able to fully recover from the initial delay. If we restrict ourselves to the realistic case of $n_{\text{norm}} = 0''$, then we can conclude that a good value for $n_{\text{lookahead}}$

Table 2.16: Impact of lookahead context on LibriSpeech and TED-LIUM on the latency ($n_{\text{l}} = n_{\text{lookahead}}$, $n_{\text{n}} = n_{\text{norm}}$).

	LibriSpeech		*TED-LIUM*	
n_{l} (sec)	$n_{\text{n}} = 2''$	$n_{\text{n}} = 0''$	$n_{\text{n}} = 2''$	$n_{\text{n}} = 0''$
0.125	1.8 ± 0.5	0.6 ± 0.3	0.7 ± 0.5	0.3 ± 0.1
0.250	1.7 ± 0.5	0.6 ± 0.2	0.8 ± 0.4	0.5 ± 0.1
0.500	1.6 ± 0.5	0.8 ± 0.2	0.9 ± 0.3	0.8 ± 0.1
1.000	2.2 ± 0.5	1.4 ± 0.2	1.4 ± 0.2	1.3 ± 0.1
2.000	2.6 ± 0.6	2.9 ± 0.7	2.4 ± 0.3	2.3 ± 0.3

is 0.5 seconds, which leads to a latency $\sim$ 1 second and a competitive WER in LibriSpeech and TED-LIUM.

Off-line/Streaming comparison

With the values set for $n_{\text{norm}} = 2$ seconds and $n_{\text{lookahead}} = 0.5$ seconds, that involves a latency of $\sim$0.8 seconds, we have performed a final comparison between the off-line and the streaming systems, considering the *test* partitions. It is important to remark that, for the off-line setup, we have used the whole normalized sequence, while the streaming setup has just a limited context for both the acoustic features and normalization. Table 2.17 shows these results, reflecting how we can provide an streaming decoder with a similar performance in terms of WER, working with a low-latency setup.

Table 2.17: WER results on test sets for LibriSpeech and TED-LIUM.

	LibriSpeech	*TED-LIUM*
Off-line setup	10.2	8.2
Streaming setup	10.7	8.7

3.4 Conclusions and future work

In this work we materialized the streaming decoder that was proposed in (Jorge et al. 2019), using LSTM-based models for the acoustic and language modeling. We studied the impact of the normalization during the on-line decoding, as well as the impact of the acoustic future context in the WER and the latency. We obtained a system that provides a similar performance in terms of WER but working in a full streaming setup, with a latency of $\sim$1 second. We evaluated our approach in LibriSpeech and TED-LIUM, obtaining a competitive WER under a streaming regime.

As future work, we want to study the impact of using the same $n_{\text{lookahead}}$ window for training and decoding. Additionally, we want to consider alternative

approaches for dealing with the limited future context, for example, considering all the past context or using faster models such as feed forward neural networks for the future context.

References

Abadi, Martín, Ashish Agarwal, et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (cit. on p. 67).

Arisoy, E. et al. (2014). "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1, pp. 184–192 (cit. on p. 64).

Bourlard, H. and C. J. Wellekens (1989). "Links between Markov Models and Multilayer Perceptrons". In: *Advances in Neural Information Processing Systems I*, pp. 502–510 (cit. on p. 66).

Chen, X. et al. (2015). "Improving the training and evaluation efficiency of recurrent neural network language models". In: *Proc. of ICASSP 2015*, pp. 5401–5405 (cit. on p. 67).

Chen, Xie, Xunying Liu, et al. (2016). "CUED-RNNLM – An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *Proc. of ICASSP 2016*. Shanghai, China, pp. 6000–6004 (cit. on p. 67).

Chen, Xie, Xunying Liu, et al. (2017). "Future word contexts in neural network language models". In: *Proc. of ASRU 2017*, pp. 97–103 (cit. on p. 64).

del-Agua, M.A. et al. (Nov. 2014). "The translectures-UPV toolkit". In: *Proc. of Advances in Speech and Language Technologies for Iberian Languages 2014*, pp. 269–278 (cit. on p. 67).

Graves, A. and J. Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks* 18.5-6, pp. 602–610 (cit. on p. 63).

Hernandez, François et al. (2018). "TED-LIUM 3: Twice as Much Data and Corpus Repartition for Experiments on Speaker Adaptation". In: *Proc. of SPECOM 2018*, p. 198 (cit. on pp. 64, 66, 67).

Huang, Z. et al. (2014). "Cache based recurrent neural network language model inference for first pass speech recognition". In: *Proc. of ICASSP 2014*, pp. 6354–6358 (cit. on p. 64).

Jorge, Javier et al. (2019). "Real-time One-pass Decoder for Speech Recognition Using LSTM Language Models". In: *Proc. of InterSpeech 2019*, pp. 3820–3824 (cit. on pp. 64, 65, 68, 70).

Jozefowicz, R. et al. (2016). "Exploring the limits of language modeling". In: *arXiv preprint arXiv:1602.02410* (cit. on p. 64).

Kombrink, Stefan et al. (2011). "Recurrent Neural Network based language modeling in meeting recognition". In: *Proc. of Interspeech 2011*, pp. 2877–2880 (cit. on p. 64).

Mnih, Andriy and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models". In: *Proc. of ICML* (cit. on p. 67).
Mohamed, A. et al. (2015). "Deep bi-directional recurrent networks over spectral windows". In: *Proc. of ASRU 2015*, pp. 78–83 (cit. on pp. 64, 65).
Nolden, David (Apr. 2017). "Progress in Decoding for Large Vocabulary Continuous Speech Recognition". PhD thesis. Computer Science Department RWTH Aachen University Aachen (Germany): RWTH Aachen University (cit. on p. 65).
Ogawa, A. et al. (2018). "Rescoring N-Best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model". In: *Proc. of ICASSP 2018*, pp. 6099–6103 (cit. on p. 64).
Panayotov, V. et al. (2015). "Librispeech: an ASR corpus based on public domain audio books". In: *Proc. of ICASSP 2015*, pp. 5206–5210 (cit. on pp. 64, 66).
Shi, Yongzhe et al. (2014). "Efficient One-Pass Decoding with NNLM for Speech Recognition". In: *IEEE Signal Processing Letters* 21.4, pp. 377–381 (cit. on pp. 64, 65).
Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech 2002*, pp. 901–904 (cit. on p. 67).
Young, S. J., J. J. Odell, and P. C. Woodland (1994). "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proc. of Workshop on Human Language Technology 1994*, pp. 307–312 (cit. on p. 66).
Zeyer, A., R. Schlüter, and H. Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Proc. of InterSpeech 2016*, pp. 3424–3428 (cit. on pp. 64, 65).
Zeyer, Albert et al. (2017). "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition". In: *Proc. of ICASSP 2017*. IEEE, pp. 2462–2466 (cit. on p. 67).

4 Live Streaming Speech Recognition Using Deep Bidirectional LSTM Acoustic Models and Interpolated Language Models

Jorge, Javier; Giménez, Adrià; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons

IEEE/ACM Transactions on Audio, Speech, and Language Processing, VOL. 30, 2022

DOI 10.1109/TASLP.2021.3133216

IF 3.919 - Ranking 4/31 Acoustics (Q1)

2021

Live Streaming Speech Recognition Using Deep Bidirectional LSTM Acoustic Models and Interpolated Language Models

Jorge, Javier; Giménez, Adrià; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Abstract

Although Long Short-Term Memory (LSTM) networks and deep Transformers are now extensively used in offline ASR, it is unclear how best offline systems can be adapted to work with them under the streaming setup. After gaining considerable experience on this regard in recent years, in this paper we show how an optimized, low-latency streaming decoder can be built in which bidirectional LSTM acoustic models, together with general interpolated language models, can be nicely integrated with minimal perfomance degradation. In brief, our streaming decoder consists of a one-pass, real-time search engine relying on a limited-duration window sliding over time and a number of ad hoc acoustic and language model pruning techniques. Extensive empirical assessment is provided on truly streaming tasks derived from the well-known LibriSpeech and TED talks datasets, as well as from TV shows on a main Spanish broadcasting station.

4.1 Introduction

Live video streaming services over the Internet have increased dramatically in recent years because of higher user demand and bandwidth speeds. This has resulted in a growing need by live video streaming platforms to provide high-quality automatic speech transcriptions. However, the application of state-of-the-art neural-based Automatic Speech Recognition (ASR) models to video streaming is a highly complex and challenging task due to real-time and low-latency decoding constraints.

At this time, state-of-the-art ASR systems are based on the hybrid Hidden Markov Model (HMM) and neural network approach (Yu and Deng 2014). In particular, deep Bidirectional Long Short-Term Memory (BLSTM) networks have proven to be a powerful architecture for acoustic modeling in a wide range of ASR tasks (Graves and Schmidhuber 2005; K. Chen and Huo 2016; Albert Zeyer

et al. 2017). In the same way, Transformer-based architectures have recently reached very promising results for language modeling (Irie et al. 2019), though LSTM recurrent neural networks (LSTM-RNN) are still broadly used (Jozefowicz et al. 2016). It goes without saying that end-to-end systems are attracting great attention, and this includes a number of proposals for operation under low-latency streaming decoding (Moritz et al. 2020; Miao et al. 2020; Nguyen et al. 2020). However, despite their simplicity and promising prospects, it is still unclear whether or not they will soon surpass state-of-the-art hybrid systems combining independent models trained from vast amounts of data.

Two main challenges need to be addressed so as to properly adapt hybrid ASR systems to the streaming setup. The first one is due to the fact that BLSTM acoustic models can no longer be applied in their full extent, over the whole input signal. Instead, they need to be time-limited within a window sliding over time in which only a small fraction of non-decoded signal (right context) can be captured for the system to respond quickly after the incoming audio stream. This adaptation of BLSTM acoustic models to deal properly with the incoming audio stream also implies to dynamically carry out acoustic mean normalization as opposed to full normalization over the whole signal. An additional issue to be considered is to adapt well-known pruning techniques as acoustic look-ahead (David Nolden 2017; Berlin Chen et al. 2004) to work with BLSTM acoustic models to speed up the decoding process. The second one is that Transformer and LSTM-RNN language models (LMs) cannot likewise be applied as they use to be, by rescoring n-best hypotheses or lattices in a two-pass decoding approach (Xie Chen, Liu, et al. 2017; Ogawa et al. 2018; Kombrink et al. 2011). In all these cases, efficient techniques are required for on-the-fly scoring under a real-time one-pass decoding scheme.

The use of BLSTM acoustic models under streaming conditions has been explored in several recent works. In (Mohamed et al. 2015), a finite sliding window was applied to approximate the acoustic posterior probability of the center frame. This approach was improved in (A. Zeyer, Schlüter, and H. Ney 2016) by using a more accurate weighting scheme of overlapping windows. Under this approach, BLSTM-based models outperformed deep neural networks (DNNs) under the streaming setup, also showing that a right context of limited duration suffices to reach a performance similar to that of the offline setup. In contrast to using a sliding window over the incoming signal, a different approach consists in splitting it into overlapping chunks with appended (past and future) contextual observations. This approach was followed in (K. Chen and Huo 2016), where the so-called Context-Sensitive-Chunk (CSC) method was proposed to speed up BLSTM training for low-latency decoding by just adding some delay in between consecutive chunks. This method can be accelerated by simply avoiding computations on the left context, as done with the Latency-Controlled BLSTMs proposed in (Zhang et al. 2016), which in turn can be further improved as shown in (Xue and Yan 2017). However, in all these previous works, empirical evaluations were not performed under genuine streaming conditions, that is, dealing with the speech signal as

an incoming audio stream and, therefore, on-the-fly mean normalization was not considered at all. Moreover, basic n-grams LMs were used in experiments which greatly helped to improve the responsiveness of the system in the evaluation of system latencies as were reported only in the case of the CSC approach.

Regarding the use of neural LMs, to our knowledge, the direct use of this technique during decoding was first explored in (Shi et al. 2014), where the authors proposed the use of a Variance Regularization term together with caching strategies for fast decoding. Despite using feed-forward neural LMs in decoding, empirical results showed significant relative improvements both in speed and accuracy. Other relevant contributions addressing one-pass decoding with neural LMs have focused on heuristics to reduce the number of queries to the model and catching network states (Z. Huang et al. 2014), alternative one-pass decoding strategies such as on-the-fly rescoring (Hori, Kubo, and Nakamura 2014), improving CPU-GPU communications (K. Lee et al. 2015) and, more recently, combining Gated Recurrent Units with more efficient objective functions, such as Noise Contrastive Estimation (Kyungmin Lee et al. 2018). Certainly different from these contributions, other authors have explored the idea of converting neural LMs, either recurrent or not, into n-gram models that can thus be smoothly integrated into a conventional decoder (Arisoy et al. 2014; Singh et al. 2017). It is worth noting, however, that these approaches were only focused on language modeling and not on the low-latency streaming decoding problem.

This work takes as a starting point a novel architecture for real-time one-pass decoding with LSTM-RNN LMs proposed in (Jorge, Giménez, Iranzo-Sánchez, Civera, et al. 2019). In it, one-pass decoding was accelerated by estimating look-ahead scores using precomputed static look-ahead tables. Moreover, LSTM-RNN LM probabilities were efficiently computed using Variance Regularization and lazy evaluation. Later on, in (Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020), this architecture for real-time one-pass decoding was extended to include BLSTM acoustic models within a time sliding window, also used as a window for time-constrained, on-the-fly acoustic feature normalization. Not surprisingly, empirical assessment of this extended architecture under strict streaming conditions proved it was really effective, indeed keeping the pace with non-streaming (offline) systems. The most recent refinement in connection to this research line has consisted in replacing streaming-adapted LSTM-RNN LMs with Transformer LMs (Baquero-Arnal et al. 2020). In doing so, empirical results on the well-known LibriSpeech (Panayotov et al. 2015) and TED-LIUM (Rousseau et al. 2014) tasks have shown that this refinement leads to top, state-of-the-art recognition rates and latencies under streaming conditions. In short, it has been shown that hybrid one-pass ASR systems built in this way can work under both, offline and streaming conditions with no significant differences in quality.

This work is intended to provide a complete, detailed reference of the main contributions made along the research line described above, also including a number of new additional enhancements and a new and extensive empirical evaluation un-

der streaming conditions. In particular, the following important novel algorithmic enhancements are provided:

- The sliding window framework proposed in (A. Zeyer, Schlüter, and H. Ney 2016) is revisited to include and adapt necessary concepts for proper streaming decoding.
- Also for proper streaming decoding, novel methods for acoustic feature normalization are explored.
- Along with these two streaming-oriented enhancements, and to improve the general performance of the decoder, new pruning techniques for fast decoding are also considered. More precisely, a new approach for the acoustic look-ahead is provided, together with a more efficient pruning to speed up the use of interpolated neural LMs.

Only after including these enhancements, a fully-fledged streaming ASR system can be effectively deployed into production. For empirical evaluation, apart from the conventional LibriSpeech and TED-LIUM tasks considered in previous work, two genuine streaming tasks also posed for streaming benchmarking: a video-based version of TED-LIUM with unsegmented talks, and a set of full-length videos from a Spanish TV broadcaster.

The paper is organized describing separately the two main components which generally speaking deserve special attention in the deployment of a streaming decoder. In particular, the use of deep BLSTM acoustic models for streaming is described in Section 4.1. On the other hand, the efficient pruning technique for fast one-pass decoding using interpolated neural LMs is presented in Section 4.2. All these components are empirically assessed in Section 4.3 with emphasis on the key adaptation parameters required for finding an appropriate (task-dependent) trade-off between accuracy and latency. Finally, the main conclusions drawn from on this research line are summarized in Section 4.4.

Deep Bidirectional LSTM Acoustic Models for Streaming

In this section, all the issues concerning the use of deep BLSTM acoustic models for streaming are described. Firstly, the sliding window framework proposed in (A. Zeyer, Schlüter, and H. Ney 2016) is revisited in Section 4.1 to include and adapt necessary concepts for proper streaming decoding using BLSTM acoustic models. Secondly, a novel approximation for computing acoustic look-ahead scores is proposed in Section 4.1. Lastly, several acoustic mean normalization methods for streaming are proposed in Section 4.1.

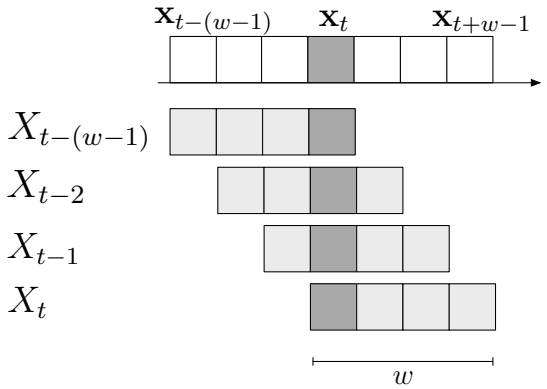

Figure 2.4: Frame sequence at the top and just below a sliding window of $w = 4$ frames at all steps embracing frame t, $\vec{x}_t$.

Streaming decoding using BLSTM acoustic models

Let $\vec{x}_1^{\infty}$ be an unbounded sequence of frames computed from the incoming audio stream, which is being processed by application of a sliding window of w frames shifting one frame to the right at each step (though a step size of more than one frame can be also used if convenient). Thus, frames t to $t + w - 1$ are covered by the sliding window at step t, $X_t = \vec{x}_t^{t+w-1}$. This is illustrated in Figure 2.4, where the sliding window is also depicted at all the $w - 1$ preceding steps embracing frame t. $X_{t-(w-1)}^{t-1}$.

For each acoustic state a, we assume that a BLSTM acoustic model is available to compute the posterior probability of the n-th frame within the sliding window at step t, $p_n(a \mid X_t)$. As frame t falls into position $w - i + 1$ of the sliding window at step $t-(w-i)$, $1 \leq i \leq w$, it gets w different posteriors from which its acoustic score is computed by just (weighted) averaging:

$$q(\vec{x}_t, a) = \frac{1}{w} \sum_{i=1}^{w} p_{w-i+1}(a \mid X_{t-(w-i)}) \tag{2.2}$$

For this score to be efficiently computed, we assume that the recurrent state of each BLSTM does not depend on its previous states, and hence the posterior it provides for frame t only depends on (its position in) the window context considered. This assumption enables fast computation of posteriors, not only by running independent BLSTM queries in parallel, but also by avoiding repeated posterior computations during consecutive step batches. Figure 2.5 shows how this looks like in a simple example where the acoustic scores for $b = 3$ consecutive

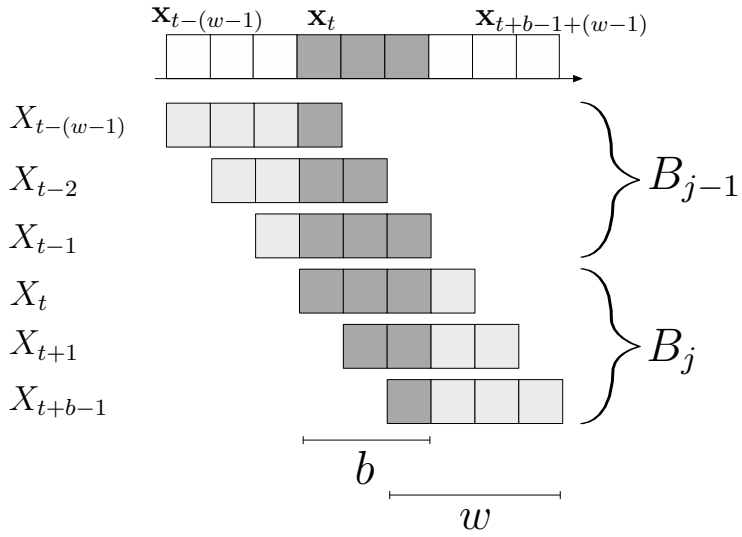

Figure 2.5: Computing the acoustic scores for $b = 3$ consecutive frames starting at t, $\vec{x}_t^{t+b-1}$, within a sliding window of size $w = 4$ during two consecutive b-step batches, B_{j-1} and B_j.

frames starting at t, $\vec{x}_t^{t+b-1}$, are computed within a sliding window of size $w = 4$ during two consecutive b-step batches, B_{j-1} and B_j.

As shown in Figure 2.5, scoring the frame subsequence $\vec{x}_t^{t+b-1}$ can be efficiently done by keeping all precomputed posteriors for it and just running b independent BLSTM parallel queries in B_j to get the posteriors still required. In the example given, for each position i ($i = 1, 2, 3$), we have $w - i$ posteriors available and i posteriors to be computed. In general, for any $b \geq 1$, a carousel may be used at each position i, $1 \leq i \leq \min(b, w - 1)$, to keep track of $w - i$ precomputed posteriors and fill in the remaining i from the b parallel BLSTM calls in batch B_j. Clearly, if $b \geq w$, no precomputed posteriors are involved from position w to b.

It is obvious that the window size w is a key adaptation parameter for streaming as it controls the duration of the acoustic context, both in the *past* ($\vec{x}_{t-(w-1)}^{t-1}$) and the *future* ($\vec{x}_{t+1}^{t+w-1}$), that is used when scoring the *current* frame $\vec{x}_t$. Needless to say, we are assuming that the most relevant acoustic context at each frame occurs within a time window of a handful tenths of second. Also, as we need the complete future context to be available for (exact) scoring, time windows longer than that may prevent the system to respond after a reasonable latency of, say, one second. This is of course a topic to be explored empirically. In this regard, note that we are limiting ourselves to symmetrical time windows of fixed

duration (w) and exact scoring, as defined in Eq. (2.2). However, if convenient, more general schemes for acoustic context management and scoring can be also devised, such as asymmetrical time windows of variable duration and approximate scoring.

As w, the batch size b is also a key adaptation parameter for streaming though, in contrast to w, its effect is only computational. In principle, we may want b as large as possible, for maximum parallelism, but also small enough for the additional future context (of $b-1$ frames) it requires not to become the dominant factor in the observed system latency. This is easily understood from Figures 2.4 and 2.5. In Figure 2.4, we have $b = 1$ and thus, apart from the future $w-1$ frames required to complete the sliding window at t, X_t, no additional frame is needed to score $\vec{x}_t$. Instead, in Figure 2.5, we have $b = 3$, and hence 2 additional frames are needed before running a 3-step batch of parallel BLSTM calls. There are also other hardware-dependent factors such as (GPU) memory bandwidth that may add up significantly to the observed latency as the batch size increases. Therefore, as with w, this is best studied empirically.

Acoustic Model Look-ahead

The acoustic look-ahead refers to the best acoustic score (emission and transition probabilities) that can be reached from a given frame $\vec{x}_t$ and an acoustic state a. More precisely, following a similar notation to (David Nolden 2017), the exact acoustic look-ahead $l(t, a)$ is defined as

$$l(t,a) = \max_{a_0^L : a_0 = a} \sum_{\tau=1}^{L} q(\vec{x}_{t+\tau}, a_\tau) + q(a_\tau, a_{\tau-1}) \,, \tag{2.3}$$

being L the number of remaining frames until the end of the speech signal, and $q(\vec{x}_t, a_t) = \log \frac{p(a_t|\vec{x}_t)}{p(a_t)}$ and $q(a_t, a_{t-1}) = \log p(a_t \mid a_{t-1})$ the emission and transition probabilities, respectively.

During decoding, for a given partial hypothesis a_1^t, an upper bound of the acoustic score $\hat{q}(a_1^t)$ is computed by adding the acoustic look-ahead score as

$$\hat{q}(a_1^t) = q(a_1^t) + l(t, a_t) \tag{2.4}$$

where

$$q(a_1^t) = \sum_{\tau=1}^{t} q(\vec{x}_\tau, a_\tau) + q(a_\tau, a_{\tau-1}) \tag{2.5}$$

It is worth noting that $\hat{q}(a_1^t)$ is an optimistic estimation of the acoustic score, since it is not only considering the score until instant t, but also the score of the best future path that could be reached from that instant according to the acoustic model. This estimated score leads to a more guided beam search, which in turn, could lead to speed up the decoding process.

Therefore, the challenge resides in efficiently computing the acoustic look-ahead score $l(t, a)$ and how to speed up the search of the best future path (see Eq. (2.3)). In fact, let us to refer previous works in which exact look-ahead scores were approximated by limiting the future context to a few frames and/or simplifying the emission models used for look-ahead calculation (David Nolden 2017; Berlin Chen et al. 2004). A major issue when acoustic look-ahead was applied over Gaussian HMMs lied in the cost of estimating the emission scores. However, in current hybrid systems based on neural networks this is not a problem anymore since, for each frame, the neural network estimates the scores for all HMM states, and usually this is performed in batch mode using GPUs. In current state-of-the-art hybrid system based on triphonemes the number of different states that must be considered during the search of an acoustic alignment (without considering the LM) may vary between half a million and several millions, depending on the total number of HMMs and the vocabulary size. This makes unfeasible to directly apply exact acoustic look-ahead estimation.

As alternative to circumvent all these drawbacks, we propose to approximate the search space by a bigram model using HMM states as tokens. In the bigram model we only consider those transitions that are allowed in the original search space. Additionally, all transitions scores are set to zero, i.e., we only focus on emission scores. Using this approach the acoustic look-ahead can be estimated at low cost using dynamic programming, and without the need of limiting the number of future frames or simplifying emission models. More precisely, for a given frame $\vec{x}_t$ and state a, the proposed acoustic look-ahead approximation $\hat{l}(t, a)$ is estimated as

$$\hat{l}(t,a)=\begin{cases}0 & t=T\\ \max\limits_{a'\in A}\hat{q}(a,a')+q(\vec{x}_{t+1},a')+\hat{l}(t+1,a') & t<T\end{cases} \tag{2.6}$$

where T is the total number of frames, A is the set of HMM acoustic states, and $\hat{q}(a, a')$ is a function that returns 0 if (a, a') is a non-zero probability bigram transition or $-\infty$ otherwise. During decoding, the look-ahead based acoustic score for a hypothesis can be incrementally updated as

$$\begin{aligned}\hat{q}(a_1^t) = \hat{q}(a_1^{t-1}) - l(t-1, a_{t-1}) + \\ + q(a_t, a_{t-1}) + q(\vec{x}_t, a_t) + l(t, a_t).\end{aligned} \tag{2.7}$$

In an offline setup the proposed acoustic look-ahead scores could be precomputed before decoding without limitation on the number of future frames. However, the streaming setup requires an on-the-fly estimation of acoustic look-ahead scores. More precisely, as described before and was illustrated in Figure 2.5, every b frames the BLSTM is queried with $b + w - 1$ frames, and outputs the emission score for the first b frames. In this setup, every time the BLSTM is queried the acoustic look-ahead scores are estimated for the first b frames of the query. In order to take full advantage of the available data, when querying the BLSTM we also retrieve partially averaged emission scores for the frames of the future context, therefore, the acoustic look-ahead score for the first frame of the batch will be estimated considering $b+w-2$ future frames, while in the last frame of the batch only $w - 1$ frames will be considered. It is worth noting, that in every BLSTM query a computational overhead of w is introduced when compared with an offline scenario. Consequently, the larger the batch size, the smaller the computational overhead and the future context limitation. However, a large batch size also means higher latencies. Therefore, this is a trade-off that must be taken into account when applying the proposed technique for streaming as will be appropriately evaluated in Section 4.3.

Acoustic Feature Normalization for Streaming

Under the offline scenario, Full Sequence Normalization (FSN) is usually performed beforehand applying mean normalization to the whole speech utterance. However, since FSN is not feasible under streaming conditions, we propose different alternatives to carry out on-the-fly sequence normalization.

The first alternative, called Dynamic Threshold Normalization (DTN), consists on the initialization of the mean by considering an initial delay of n_{norm} frames. Afterwards, the mean is dynamically updated for every new frame. In previous works, we proved that two seconds of initial delay should be enough to achieve similar performance to FSN (Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020; Baquero-Arnal et al. 2020). Although, two seconds of delay could be reasonable in a continuous streaming setup, it could be not so suitable for short utterances such as voice commands.

To overcome this limitation and taking advantage of the sliding window technique introduced in Section 4.1, we propose in this work a novel normalization scheme called Weighted Moving Average (WMA), in which mean normalization is performed using the frames of the sliding window. In this way, WMA is applied over a batch B_j of frames as

$$\hat{B}_j = B_j - \hat{\mu}_j \tag{2.8}$$

where

$$\hat{\mu}_j = \frac{\vec{f}_{j-1} + \sum_{t=1}^{b+w} B_{j,t}}{n_{j-1} + b + w} \tag{2.9}$$

being $\mathbf{f}_{j-1}$ the accumulated values of previous frames until batch B_{j-1}, $B_{j,t}$ the t-th frame in batch B_j, n_{j-1} the number of frames until batch B_{j-1}, and b and w the batch and window sizes, respectively.

The accumulated values $\mathbf{f}_j$ and n_j are updated by weighting the contribution of previous batches using a parameter α as

$$\mathbf{f}_j = \alpha \cdot \mathbf{f}_{j-1} + \sum_{t=1}^{b} B_{j,t} \tag{2.10}$$

$$n_j = \alpha \cdot n_{j-1} + b \tag{2.11}$$

Unlike FSN, WMA dynamically adapts the normalization of the speech signal to local changes, and differently from DTN, without introducing an initial delay. Therefore, it can be used without affecting the global latency even from the beginning of the utterance. Although an initial mean could be precomputed from the training set, WMA has been evaluated in this work starting from scratch on each sample and using only the information that comes from the audio stream as expected in streaming conditions. Similarly to the acoustic look-ahead, the batch size has also an impact regarding the amount of frames used to compute the mean. This impact will be evaluated during the experimental section.

4.2 Efficient One-pass Decoding using Interpolated Neural LMs

The direct use of neural LMs in one-pass decoding takes full advantage of its ability to deal with histories of unlimited length in contrast to n-gram LMs (Kombrink et al. 2011). This makes History Conditioned Search (HCS) decoders perfectly suited for its use with neural LMs, as HCS technique group hypotheses by its history allowing potentially an unlimited representation of continuous contexts (Hermann Ney and Ortmanns 2000). However, in practice, the integration of neural LMs in one-pass HCS decoders for streaming presents some relevant difficulties which need to be solved. For instance, the very efficient computation of LM look-ahead scores and word neural LM probabilities or the use of specific LM pruning parameters to reduce the search space. In the following, all these decoding issues are discussed and how they have been efficiently addressed in this proposal.

Language Model Look-ahead

There are many techniques to deal with look-ahead LM scores as the use of cache strategies, perfect hashing and precomputation of scores (Cardenal-López et al. 2002), computing look-ahead scores bottom-up from back-off LM (Langzhou Chen and Chin 2008) or leveraging the LM sparseness to partially compute look-ahead tables (D. Nolden et al. 2011). Some of these approaches use a lower order n-gram to obtain look-ahead scores, as higher n-gram orders are not feasible due to memory and computation requirements. In the case of (D. Nolden et al. 2011), 3-grams and 4-grams are also evaluated, but in this case look-ahead scores are dynamically computed.

In HCS-based decoders, LM look-ahead scores are dynamically computed every time a word-end node is reached during decoding. In order to do this efficiently for streaming, we propose to compute beforehand all these look-ahead scores in static look-ahead tables. With this purpose, a heavily pruned version of the n-gram model can be used to represent this model in a compact structure that can be used during decoding efficiently. This is a critical part of the search when it comes to speed, as this score is queried many times during search to fill the search network structure. Therefore, reducing the n-gram model to this static structure and following a cascade structure of look-ahead tables similar to that proposed in (D. Nolden et al. 2011), we apply an efficient technique to compute the look-ahead scores during decoding. It is worth noting that this pruned n-gram does not constrain the search space in any way, as this is only used to compute the look-ahead scores. This allows the decoder to consider very long word contexts in search hypotheses leveraging the benefits of using neural LMs.

Neural LM integration

During decoding, when word-end nodes are reached, look-ahead table scores are replaced with those computed from the real LM (i.e. n-grams or neural LMs). In the case of neural LM probabilities, this is an important drawback since it involves computational issues, reducing the speed of the decoder as they are usually more complex models than count-based ones. For this reason, neural LMs are typically applied in a second step of recognition using n-best or lattice rescoring. To alleviate this drawback we propose to apply the Variance Regularization technique (Shi et al. 2014) reducing the complexity of the computation of the output layer where the softmax function is computed. This technique involves a regularization term during training that aims to reduce the variance of the denominator of the softmax adjusting it to a constant. This constant is kept and then used during decoding when computing neural LM probabilities, instead of computing the denominator. As opposed to LSTM-RNN LMs which store the previous context in an internal vector, Transformer LMs need to compute all the previous history when a new word comes in for the attention to work properly. To deal with long audio streams (possibly hours of continuous speech) we should

limit the history to the n previous words, where n is a parameter provided in decoding. It is important to remark that the Transformer LM training enforced no history restriction, indeed there could be a training-decoding mismatch regarding history length that may harm the performance, as sentences of different lengths are devoted to training and the history length is not adjusted beforehand.

LM pruning parameters

In order to further speed up the decoding process, specific LM pruning parameters had to be incorporated to the one-pass decoder, to reduce the search space or the number of queries in the computation of neural LM probabilities (Jorge, Giménez, Iranzo-Sánchez, Civera, et al. 2019). One of these parameters is the Language Model History Recombination (LMHR) which defines the number of words to be considered before performing hypothesis recombination during decoding. LMHR parameter is needed to control the length of histories since, in HCS decoders, hypotheses are grouped according to their history, meaning that without enforcing any back-off recombination previous histories of active hypotheses tend to grow without any limitation. However, this effect that could be considered a feature turned to be a problem when long histories are considered, as active hypotheses cluster over similar contexts that differ only in words far from the current frame. This parameter aims to reduce the uncertainty of having these long and very similar hypotheses making pruning less effective. This is achieved by combining them when they share a given number of words, and that is indeed what this parameter defines, the number of previous words evaluated to combine active hypotheses. The second pruning parameter, named Language Model Histogram Pruning (LMHP), limits the number of hypotheses that will query the neural LM after reaching new word-end nodes during decoding. This is particularly effective in reducing the costly neural LMs computation, as active hypotheses are pruned before performing any computation. Unlike global histogram pruning applied to thousands of hypotheses after each decoding step, LMHP affects tens or hundreds of hypotheses.

LM interpolation

It is worth stressing that the proposed one-pass HCS-based decoder enables the use of linearly interpolated count-based and/or neural LMs which to our knowledge is unprecedented in streaming ASR.

Table 2.18: Basic statistics of dev and tests sets in the evaluation tasks: Duration in hours, number of samples (segments or videos), average duration of samples in seconds plus-minus standard deviation ($d_\mu \pm \sigma$), and running words (RW) in thousands (K).

Task	Set	Hours	#Samples	$d_\mu \pm \sigma$	RW(K)
LS	dev-other	5.3	2864	6.4 ± 4.3	51
(Panayotov et al. 2015)	test-other	5.1	2939	6.5 ± 4.4	52
TDs	dev-legacy	1.6	507	11.3 ± 5.6	18
(Rousseau et al. 2014)	test-legacy	2.6	1155	8.1 ± 4.3	28
TDv	dev	1.7	8	771 ± 401	18
	test	3.1	11	1004 ± 404	28
RTVE	dev1-dev	11.9	10	4267 ±1549	120
(Lleida et al. 2018)	test	39.3	59	2395 ±1673	377

4.3 Experiments

Evaluation Datasets

The proposed ASR system for streaming was evaluated on LibriSpeech (LS) and TED-LIUM release 2 speech corpus. In the case of the TED-LIUM corpus, we defined a new evaluation task referred to as TDv, in which complete video talks were transcribed without any previous segmentation in order to simulate a streaming scenario. This is, to the best of our knowledge, the first time that the TED-LIUM corpus is considered at the talk level and it could be useful to assess streaming ASR systems in future works. In order to do that, we used the complete audio track for each talk along with the STM files provided in the dataset to evaluate the WER. The conventional segment-based TED-LIUM task is referred to as TDs in this work. Additionally, we used the RTVE2018 dataset which comprises a collection of complete TV shows drawn from diverse genres and broadcasted by the public Spanish national television from 2015 to 2018 (Lleida et al. 2018). Table 2.18 summarizes the basic statistics for the dev and test sets of the tasks mentioned above. In the case of RTVE2018, an internal partition of the provided dev1 set (dev1-dev) was created for development purposes, reserving the test set for evaluation.

Training setup

In order to build the English and Spanish hybrid ASR systems, a context-dependent feed-forward DNN-HMM with three left-to-right states using MFCC 16 plus first and second derivatives (48-dim) was initially trained with our own transLectures-UPV ASR toolkit (TLK) (del-Agua et al. 2014). Then, a BLSTM-HMM acoustic model was trained following the procedure described in (Albert Zeyer et al. 2017) using filter bank 85-dimensional features and the previous DNN-HMM alignments.

The architecture of the BLSTM model has eight bidirectional hidden layers with 512 LSTM cells per layer and direction trained using both, TLK and TensorFlow (Abadi, Agarwal, et al. 2015). Following (Albert Zeyer et al. 2017), we performed chunking during training by considering a context to perform back propagation through time to a window size of 50 frames. Additionally, SpecAugmentation was applied by means of time and frequency distortions (Park et al. 2019). Finally, a final step of sequence discriminative training was performed using our in-house implementation of lattice-based MMI to adjust the transition scores and the weights of the softmax layer (Giménez et al. 2014).

English acoustic models were trained on 961 and 207 hours of training speech corpus for LibriSpeech and TED-LIUM release 2, respectively. After applying a phonetic decision tree (Young, Odell, and Woodland 1994), 8.3K and 10.8K tied-states (or senones) were obtained for LibriSpeech and TED-LIUM, respectively. On the other hand, Spanish acoustic models were trained using the 208 hours provided in the RTVE2018 dataset plus about 3.7k hours of internal resources. The Spanish ASR system comprises 10K tied-states.

Regarding the LM training, we used the approximately 800M words of text provided for LibriSpeech to train neural LMs, as the ngram model is provided with the corpus (*fglarge*), whereas for TED-LIUM we trained the LMs with the six provided subsets plus the TED-LIUM training audio transcriptions with up to 230M running words. Vocabularies were restricted to 200K and 153K words for LibriSpeech and TED-LIUM, respectively. In the case of the Spanish system, text resources were obtained from internal sources and other public repositories shown in Table 2.19. The vocabulary size was over 254K words and a 1-gigaword random subset of the LM data was selected to train the Spanish neural LMs. To train neural LMs when the vocabulary is defined in advance, we decided to obtain the vocabulary as the intersection between the provided vocabulary and that of the training data. In this way, the model avoids having null-word probabilities for words that are in the vocabulary but not in the training set. We take this into account when computing perplexities by renormalizing the unknown-word score accordingly.

As LMs, we used n-grams, LSTM-RNN LMs and Transformer LM (TLM), combining them through a linear interpolation. Count-based models were trained using SRILM (Stolcke 2002). Apart from the 4-gram model provided for LibriSpeech, we trained a 4-gram Kneser-Ney smoothed LM for TED-LIUM using the same data as (Rousseau et al. 2014). To compute the static look-ahead tables, a pruned version of these n-gram models was computed for each task. We obtained OOV ratios of less than 0.6% in all tasks.

The CUED-RNNLM toolkit (Xie Chen, Liu, et al. 2016) was used to train LSTM-RNN LMs with Noise Contrastive Estimation (NCE) criterion (Mnih and Teh 2012), and the normalization constant learned from training was used during decoding (X. Chen et al. 2015). Based on the lowest perplexity on the dev sets,

Table 2.19: Statistics of Spanish text resources used for language modeling. S=Sentences, RW=Running words, V=Vocabulary. Units are in thousands (K).

Corpus	S(K)	RW(K)	V(K)
Internal: TV, entertainment	4799	59235	307
Internal: education	87	1526	35
Internal: politics	1361	35170	126
Opensubtitles (Lison and Tiedemann 2016)	212635	1146861	1576
UFAL (*UFAL Medical Corpus* 2017)	92873	910728	2179
Wikipedia (*Wikipedia* 2020)	32686	586068	3373
UN (Callison-Burch et al. 2012)	11196	343594	381
News Crawl (*News Crawl corpus (WMT workshop)* 2015)	7532	198545	648
eldiario.es (*Eldiario.es* 2020)	1665	47542	247
El Periódico (*ElPeriodico.com* 2020)	2677	46637	291
Common Crawl (*CommonCrawl* 2014)	1719	41792	486
News Commentary (*News Crawl corpus (WMT workshop)* 2015)	207	5448	83
TOTAL	369434	3423146	5785

we selected as final models those with 256-unit embedding layer and two hidden LSTM-RNN layer of 2048 units.

The training of TLMs was carried out using our own customized version of the FairSeq toolkit (Ott, Edunov, Baevski, et al. 2019) using a 24-layer network with 768 units per layer, 4096-unit FFN, 12 attention heads, and an embedding of 768 dimensions. These models were trained until convergence with batches limited to 512 tokens, 512 sentences, and 512 words per sentence. Parameters of these models were updated every 32 batches. During inference, Variance Regularization was also applied to speed up the computation of the TLM score.

Table 2.20 shows the perplexity of LMs on the development sets for all tasks. When comparing single LM performance, neural models outperformed count-based models on every task, with enough margin, almost halving the perplexity for LS and RTVE in the case of the LSTM-RNN. These results were further improved with TLM, reducing the perplexity in approximately 25% for LS, 15% for TDs, 32% for TDv, and 35% for RTVE, with respect to the LSTM-RNN LM. Model interpolation had diverse impact depending on the combination, but in

Table 2.20: Perplexity (PPL) and weight (W) figures on development sets, considering single models and two-way and three-way interpolation of n-gram (N), LSTM-RNN LM (L) and Transformer LM (T). Interpolation weights were optimized by minimizing PPLs of the interpolated models.

	LS		TDs		TDv		RTVE	
	PPL	W(%)	PPL	W(%)	PPL	W(%)	PPL	W(%)
N	140.6		117.5		126.1		179.5	
L	72.5		84.0		94.2		98.4	
T	54.2		71.0		64.0		63.3	
L+N	71.6	91+ 9	71.4	73+27	77.1	70+30	93.2	85+15
T+N	53.9	96+ 4	60.5	74+26	55.6	78+22	61.6	94+ 6
T+L	53.5	86+13	64.1	62+38	61.0	78+22	60.7	87+13
T+L+N	53.4	86+12+ 2	58.1	56+25+18	54.8	70+12+18	59.5	85+10+ 5

general using all three models provided the best perplexity for each task. Consistently with the single performance, the TLM obtains the highest weights in the different LM combinations for LS and RTVE (∼85-95%), while LSTM-RNN and ngram models still have an important weight for TED-LIUM tasks, ranging from 22% to 38% when are combined with TLM. When considering the three-way interpolation, again LS and RTVE perplexities show a similar behavior to that of the two-way interpolation with high weights for the TLM, while TLM reduces its weight in favor of LSTM-RNN and ngram in TED-LIUM tasks. TLM history limitation was optimized for best perplexity in each case using the same history size when TLM was interpolated with other LMs.

Experiments on acoustic modeling for streaming

The use of BLSTM acoustic models under streaming conditions was evaluated in the following way. First, we studied the effect of the window size presented in Section 4.1 in the performance of the decoder considering that Full Sequence Normalization (FSN) is performed beforehand. In this way, the optimal window size was fixed for each task in order to be used in the following experiments. Then, the impact of the acoustic look-ahead was gauged to prove its pruning effectiveness, and the different methods for acoustic feature normalization proposed in Section 4.1 were also assessed. In all these experiments, only count-based LMs were used in order to isolate the effect of the proposed acoustic-related techniques on the decoding.

Figure 2.6 shows Word Error Rate (WER) as a function of the window size (w) in seconds from 0.1 (or 10 frames) to 1 second (or 100 frames) for each task. It is worth noting that in this experiment the acoustic models were the same and only the window size was varied during decoding. In LibriSpeech and the TED-LIUM tasks, more context means better performance up to the point at which

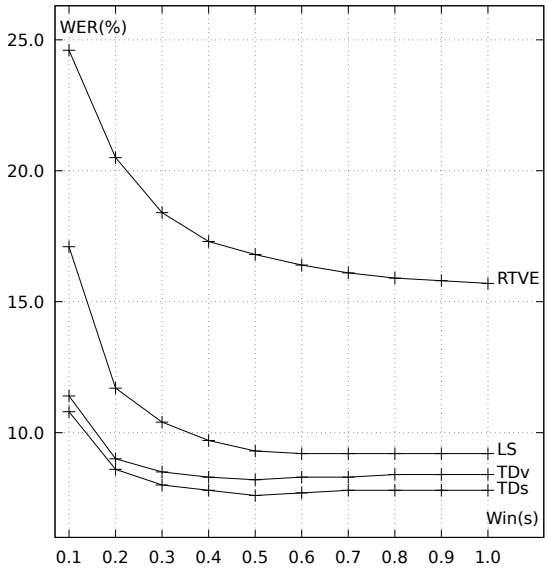

Figure 2.6: WER vs. window size in seconds for all tasks.

the windows size is equal to the chunk size used during acoustic training. Beyond this point there is no improvement by increasing the future context leading to more fixed latency without any decrease in WER. Differently from these tasks, RTVE obtains slight improvements after increasing the window size up to about 1 second of future context. This different behavior can be explained because of RTVE is composed of real TV shows with heterogeneous conditions and, therefore, further improvements can be expected considering larger contexts to better deal with changing audio conditions. Based on these empirical results, the optimal window sizes were fixed to 0.5 seconds for LibriSpeech and TED-LIUM tasks and, considering that 0.6 for RTVE pays off in WER, we selected this value to include similar fixed delays between all tasks.

In the following experiments, the trade-off between WER and latency is evaluated as it is a critical factor in streaming systems. In all these experiments, latency is measured as the time elapsed between the instant at which an acoustic frame is generated and it is fully processed by the decoder. The final latency for a sample (segment or video) is estimated as the average of the latencies at frame level. These measurements were run on an Intel i7-3820 CPU @ 3.60GHz, with 64GB of RAM and a RTX 2080 Ti GPU card. For simplicity, the time required to transform raw audio into filter bank was not included in our measures since this time is negligible and complicates the procedure used to estimate latencies.

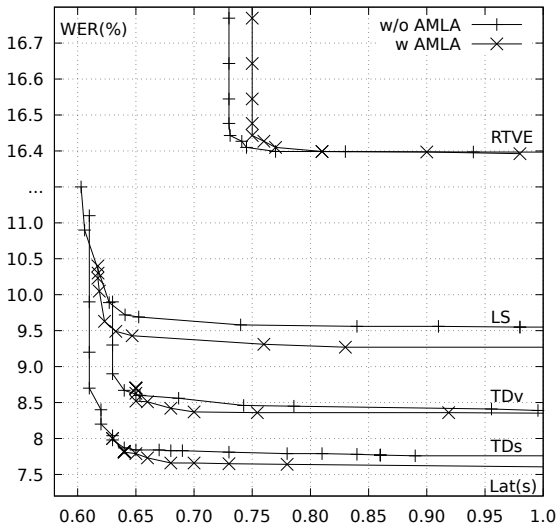

Figure 2.7: WER vs. latency in seconds with or without AMLA enabled for each task.

Figure 2.7 shows WER as a function of latency with or without Acoustic Model Look-Ahead (AMLA) enabled, using $b = 20$. As observed, AMLA is effective to decrease the search effort by exploring more promising paths and, consequently, better performance can be achieved at the same level of latency. This is mainly observed for LibriSpeech and TED-LIUM tasks, but not in RTVE. In RTVE, the number of active hypotheses is less than in LibriSpeech or TED-LIUM, meaning that the reduction in the number of active hypotheses does not compensate for the computational overhead of AMLA.

As stated in Section 4.1, the batch size directly impacts on both, latency and WER when AMLA is enabled, as the set of windows in the batch will be used to compute not only the acoustic scores, but also the acoustic look-ahead with partial acoustic scores. Regarding this impact, we explored batch sizes of 20 and 40 with AMLA enabled, observing that more context to compute AMLA scores lead to similar accuracy for segment-based tasks, such as LibriSpeech and the conventional TED-LIUM, but slightly better WERs in video-based tasks, such as TED-LIUM videos and RTVE. This is explained by the limited context of short segments of the former tasks compared to the latter tasks. In the case of RTVE, as very similar performance is achieved with or without AMLA enabled, the effect of batch size seems to be negligible. Finally, as expected, higher latencies are obtained with larger batch sizes in all tasks, as longer delays are introduced to gather enough frames to complete the batch. According to these results, in the

remaining experiments, AMLA was enabled for LibriSpeech and both TED-LIUM tasks, but not for RTVE.

Figures 2.8 and 2.9 depict for segment-based and video-based tasks, respectively, the effect in WER for the acoustic feature normalization schemes described in Section 4.1. In the case of the WMA scheme, WER is also shown as a function of the batch size (b) and the parameter α used to weight the importance of frames in previous batches. As observed, FSN and DTN provided similar WER in all tasks with the exception of LibriSpeech where higher improvements were achieved when FSN is applied.

Nevertheless, WMA clearly outperforms FSN and DTN when decoding long sequences, as shown in Figure 2.9 for TED-LIUM videos and RTVE tasks. The capacity of WMA to partially forget the previous context and adapt to new acoustic conditions seems to improve the performance of the recognition as more acoustic variations are likely to appear in long sequences. This is not the case of segment-based tasks shown in Figure 2.8, in which WMA did not outperform FSN, since sequences are shorter and acoustic conditions more stable (i.e. usually one sentence with a single speaker). When looking into the parameter α of WMA, it is observed that values close to 1.0 (equally weighting the previous and current batches) benefit segment-based tasks, as this is close to consider the complete sequence for normalization. However, values of α close to 0.9 provide better results in video-based tasks.

Additionally, we assessed the impact of the batch size in the normalization context. In this regard for normalization, unlike AMLA, using a batch size of 40 instead of 20 provided consistently better results along all tasks. Despite of this, taking the WER-latency trade-off into consideration, a batch size of 20 was selected in the following experiments to keep the latency as low as possible.

Experiments on language modeling for streaming

As shown in Table 2.20, the TLM provided the best performance measured in terms of PPL for all tasks. This is in line with previous results reported in (Irie et al. 2019). For this reason, the following experiments are focused on performing a comprehensive evaluation of the TLM behavior in streaming conditions.

As introduced in Section 4.2, the previous history of word sequences should be limited in order to keep the performance of the streaming decoder. This enforces a limitation in the number of words to consider when computing the LM probabilities that matches the history limitation of the TLM. In addition, the LMHR parameter controls the LM previous context to decide whether hypothesis recombination is performed. Both parameters are interrelated in streaming decoding as shown in the following experiments.

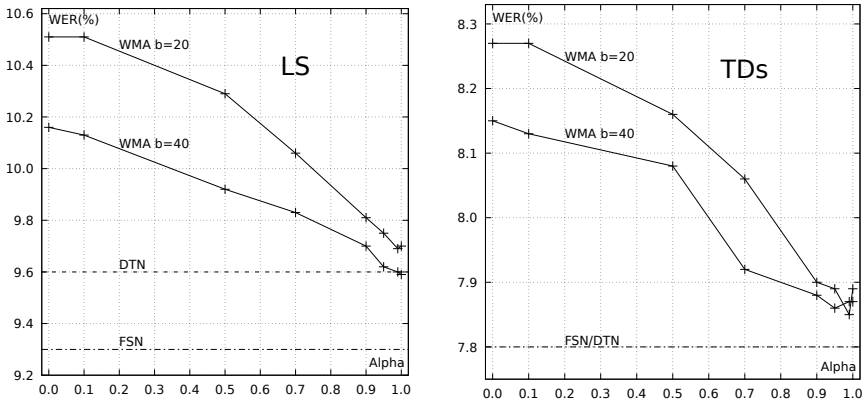

Figure 2.8: FSN, DTN and WMA normalization (with different α values) schemes evaluated on WER for segment-based tasks.

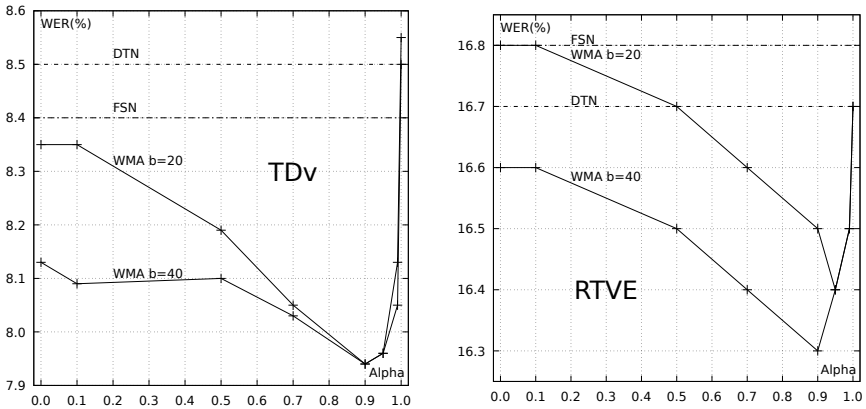

Figure 2.9: FSN, DTN and WMA normalization (with different α values) schemes evaluated on WER for video-based tasks.

Figures 2.10 and 2.11 plot WER as solid curves (left y-axis) and PPL as a dashed curve (right y-axis), as a function of the TLM history limitation in number of words using different LMHR values for all tasks. To better understand the values of the LMHR parameter, for instance, a LMHR of 3 would indicate that hypothesis recombination is performed at 4-gram level.

Similarly to previous experiments related to acoustic modeling, segment-based tasks in Figure 2.10 show a different trend compared to video-based tasks in Figure 2.11. Figure 2.10 shows that LMHR curves behaved similarly when increasing the TLM history limitation. This limit reached the best operating point in 60 words for LibriSpeech and TED-LIUM matching up with the lowest perplexity in both cases. The best performance was achieved with LMHR values of 12 and 9 for LibriSpeech and TED-LIUM, respectively, while higher values provided slightly higher WERs. This would indicate that for these segment-based tasks, longer histories have very similar contexts that make pruning less effective.

In the case of the video-based tasks, Figure 2.11 shows that PPL and WER figures increase beyond a TLM history of about 40 words. It is worth noting that in both TED-LIUM tasks the trained TLM was the same, that is, it was trained from full sentences not complete videos. This would explain why the PPL increased on video-based tasks when more than 60 words were considered for the TLM history. This fact was also reflected in WER, since the best results were consistently achieved using a LMHR value of 9 for both TED-LIUM tasks. The aforementioned optimum operating point of 60 words became on 40 words in video-based tasks as the speech input now is not as structured as in the segment-based tasks, and the beginning and end of sentences can be mixed during decoding, a situation that was not considered when training the TLM.

Regarding the RTVE task, a more stable behavior was observed in performance using a broader range of the TLM history. In this case, the performance only degraded when very high values of TLM history of about 140 words were considered. This might be explained by the fact that this LM was trained with a huge variety of text resources with very different sentence lengths and contexts. However, the performance degradation as a result of longer histories highlights the need of training specific models that take into account intra and inter sentence contexts considering complete videos or documents. Finally, the LM seemed not to play a crucial role in the RTVE task as can be interpreted by the very similar performance achieved by different LMHR values with a slight improvement when using a LMHR value of 9 and a TLM history of 40 words.

As introduced in Section 4.2, the LMHP parameter limits the number of active hypotheses that can query the neural LM to obtain its probability score reducing in this way the computational cost. However, the LMHP has a direct impact on WER as it limits the number of hypotheses to be considered during the rescoring step in decoding.

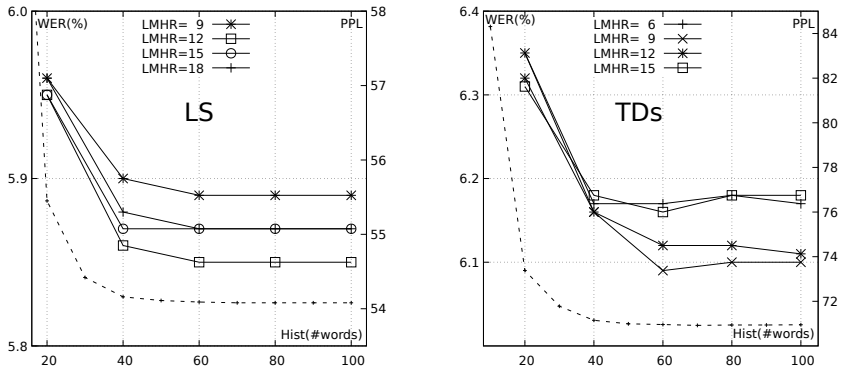

Figure 2.10: WER (left y-axis) and PPL (right y-axis) as a function of TLM history limitation and different LMHR values for segment-based tasks. Solid curves represent WER, while the dashed curve is PPL.

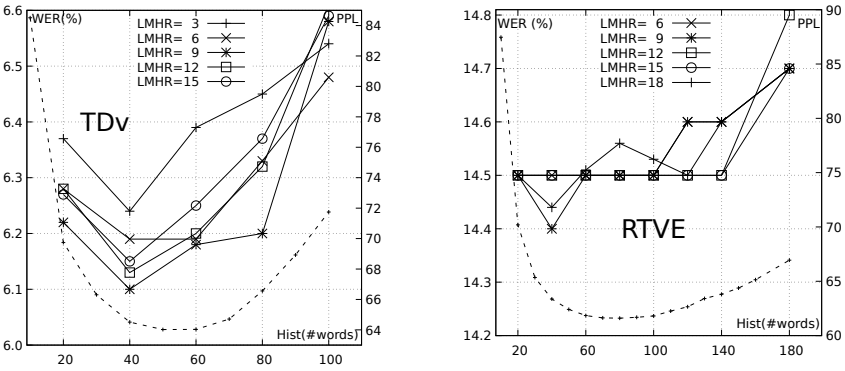

Figure 2.11: WER (left y-axis) and PPL (right y-axis) as a function of TLM history limitation and different LMHR values for video-based tasks. Solid curves represent WER, while the dashed curve is PPL.

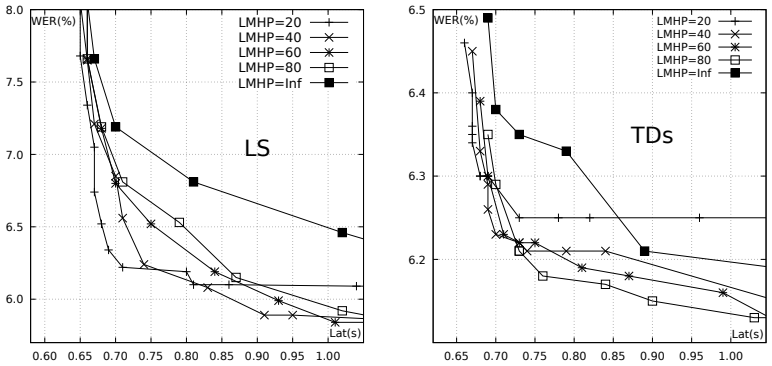

Figure 2.12: WER vs. latency in seconds varying LMHP values for segment-based tasks.

Figures 2.12 and 2.13 depict WER as a function of latency for segment-based and video-based tasks, respectively. The values of the parameter LMHP represent the number of active hypotheses, being LMHP=Inf an unlimited number of active hypotheses. As shown, the use of the LMHP pruning technique achieved an overall reduction in the system latency as can be observed by the left-shifting of the LMHP curves in almost all the LMHP values with respect to an unlimited number of active hypotheses. On the other hand, higher LMHP values translate into better WERs.

Nonetheless, the trade-off between WER and latency is very task dependent. In LibriSpeech, a LMHP value of 20 allows low latencies (about 0.7 seconds) but this limits the best WER to about 6.0%. However, allowing more LM queries (e.g. 40) leads to latencies about 0.9 seconds and WERs of about 5.8%. This behavior is different for TED-LIUM in both versions, where a LMHP value of 20 means an important increase in WER and not so much improvement in latency. This would mean that more queries in this case seems to help the decoder during the search to find best paths. Beyond this LMHP value, competitive WERs were obtained for low latencies getting better results when coming closer to latencies of about one second. In the RTVE task, a LMHP value of 20 provided a very good operating point when latency is closed to 0.8 seconds. Again, in this task the LM did not provide much information so limiting the number of queries only helped the performance of the system in terms of speed. This is specially helpful in this kind of long-recognition tasks where using conservative parameters allows us to discard a high number of active hypotheses during the search speeding up the decoding. These results show how the streaming decoder can be adapted very easily to our needs just adjusting the LMHP parameter in order to obtain the desired trade-off between WER and latency.

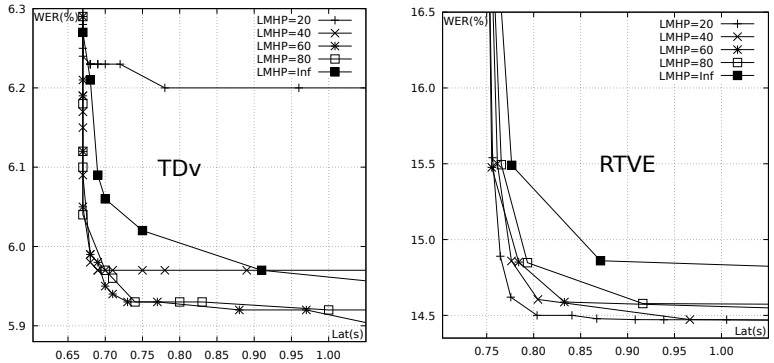

Figure 2.13: WER vs. latency in seconds varying LMHP values for video-based tasks.

As described in Section 4.2, a relevant feature of the proposed streaming decoder is its capability to interpolate on-the-fly count-based and neural LMs. Extensive experiments were carried out with the aim of evaluating the performance of different LM interpolations combining n-grams, LSTM-RNN LMs and TLMs.

Figures 2.14 and 2.15 show WER as a function of latency applying different combination of LMs for all tasks. Similarly to previous experiments, different behaviors depending on the task can be observed in these figures. In LibriSpeech, the single use of the TLM provided the best result when considering the trade-off between WER and latency. This could be the expected behavior based on the interpolation weights reported in Table 2.20 for the TLM (86-96%). The slight improvements in PPL achieved by the interpolation seemed not to have an influence on WER within the considered range of latencies. In the case of TED-LIUM, the weights were distributed in a different way, since n-gram and LSTM-RNN LMs weights were from about 22% to 38% when combined one-on-one with TLM and 30-40% when all the LMs were combined. In this case, while the combination of TLM with n-gram or with LSTM-RNN provided similar performance, the combination of the three LMs consistently provided the best WERs for all the considered range of latencies and for both, segment-based and video-based tasks. In RTVE, no significant differences were found in performance across different LM combinations. As in LibriSpeech, the TLM weight in the interpolation was between 85-94% for RTVE and this seemed to be the reason why the LM interpolation did not have a significant effect on WER.

As a final experiment, the performance of the streaming decoder was evaluated on the test set of all tasks using the hyperparameters optimized on the development sets in the previous experiments. Hyperparameters were optimized aiming at minimizing WER while the average latency was close to 1 second. Table 2.21 reports WER and latency figures on the test sets comparing them with the best

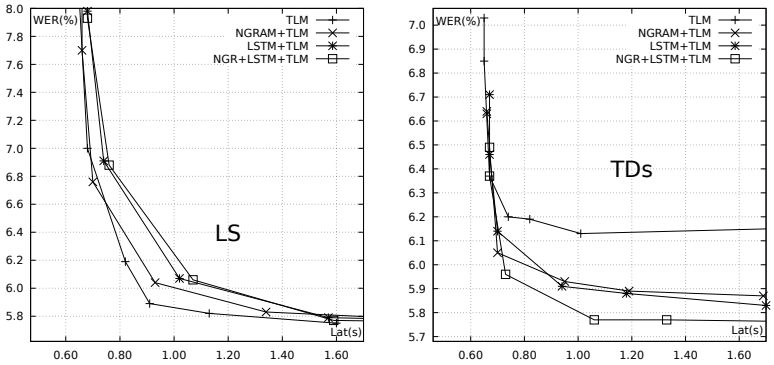

Figure 2.14: WER vs. latency in seconds considering different interpolation schemes with TLM for each segmented task.

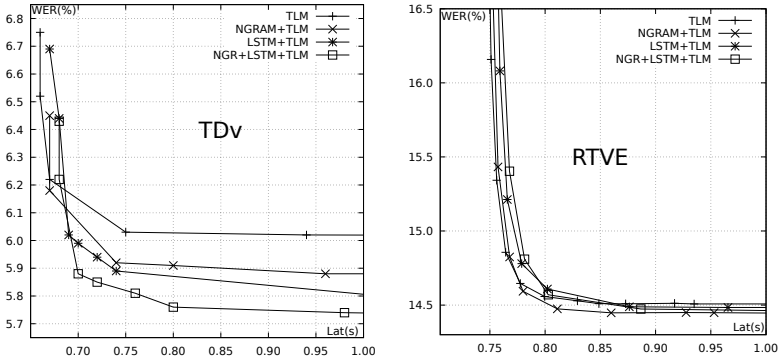

Figure 2.15: WER vs. latency in seconds considering different interpolation schemes with TLM for each video task.

Table 2.21: WER and latency in seconds on the test sets using the optimized streaming systems for all the evaluation tasks compared to previous works.

Task	Set	WER	Latency
LibriSpeech			
This work	test-other	6.3	0.93±0.62
CSC		6.8	1.37±0.63
LC		6.7	0.98±0.35
Moritz et al. (Moritz et al. 2020)		7.3–7.6	1.47–2.19
Moritz et al. (Moritz et al. 2020) (offline)		6.1	-
TDNN-F (Han, Pan, et al. 2021) (offline)		8.9	-
TED-LIUM segments (TDs)			
This work	test-legacy	6.4	0.94±0.45
Zhou et al. (Zhou et al. 2020) (offline)		5.6	-
TDNN-F (Han, J. Huang, et al. 2019) (offline)		8.7	-
TED-LIUM videos (TDv)			
This work	test-legacy	6.2	0.70±0.08
CSC		7.6	0.91±0.11
LC		7.6	0.72±0.12
RTVE			
This work	test	12.4	0.81±0.09
Jorge et al. (Jorge et al. 2018)		16.4	-

WER reported in previous works. In the case of CSC (K. Chen and Huo 2016) and LC (Zhang et al. 2016), the setup recommended by the authors was properly adapted to our framework. As observed, our streaming decoder offers competitive WERs even compared with offline decoders, demonstrating its applicability to real-world streaming applications.

Regarding latency figures, segment-based tasks, such as LS and TDs, showed a greater variability. This is explained by the fact that pruning was not so aggressive in these tasks in order to minimize WER, leading to latency peaks in some samples in which the decoder could not catch up before the sample ends. However, pruning was easier to adjust to stabilize latency in video-based tasks, as observed in TDv and RTVE.

4.4 Conclusion and future work

In this work an improved decoder based on the conventional hybrid ASR approach was proposed by adapting state-of-the-art models to the streaming setup. In particular, deep BLSTM acoustic models were adapted to the streaming conditions by using a sliding window of future context. Other techniques such as on-the-fly normalization of acoustic features and the improvement of pruning techniques related to acoustic and language models were also addressed.

The proposed decoder was evaluated by carrying out a comprehensive experimentation on well-known academic datasets and real-world challenging tasks. As reported, this decoder presented a very competitive performance being easily adapted to the task by tuning the desired trade-off between WER and latency.

Even so, the streaming setup opens some interesting challenges to be further investigated. For instance, in our experiments the same acoustic models were used independently from the window size employed at decoding time in order to alleviate the computational cost of training acoustic models. In order to address this mismatch between training and decoding conditions, the same window size in both training and decoding is desirable to better capture the nature of the task (i.e., segment-based or video-based). Moreover, according to the requirements of the latency, the window size could be dynamically adjusted in the decoding phase. On the other hand, real streaming tasks involve the recognition of long recordings across sentences, in this sense it would be very interesting to evaluate the performance of TLMs taking into account larger contexts.

References

Abadi, Martín, Ashish Agarwal, et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (cit. on p. 88).

Arisoy, E. et al. (2014). "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1, pp. 184–192 (cit. on p. 77).

Baquero-Arnal, Pau et al. (2020). "Improved Hybrid Streaming ASR with Transformer Language Models". In: *Proc. of InterSpeech 2020*, pp. 2127–2131 (cit. on pp. 77, 83).

Berlin Chen et al. (2004). "Lightly supervised and data-driven approaches to Mandarin broadcast news transcription". In: *Proc .of ICASSP*. Vol. 1, pp. I–777 (cit. on pp. 76, 82).

Callison-Burch, Chris et al. (2012). "Findings of the 2012 Workshop on Statistical Machine Translation". In: *Proc. of WMT*, pp. 10–51 (cit. on p. 89).

Cardenal-López, A. et al. (2002). "Fast LM look-ahead for large vocabulary continuous speech recognition using perfect hashing". In: *Proc. of ICASSP*. Vol. 1, pp. I-705–I-708 (cit. on p. 85).

Chen, Kai and Qiang Huo (2016). "Training deep bidirectional LSTM acoustic model for LVCSR by a Context-Sensitive-Chunk BPTT approach". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.7, pp. 1185–1193 (cit. on pp. 75, 76, 100).

Chen, X. et al. (2015). "Improving the training and evaluation efficiency of recurrent neural network language models". In: *Proc. of ICASSP 2015*, pp. 5401–5405 (cit. on p. 88).

Chen, Xie, Xunying Liu, et al. (2016). "CUED-RNNLM – An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *Proc. of ICASSP 2016*. Shanghai, China, pp. 6000–6004 (cit. on p. 88).

Chen, Xie, Xunying Liu, et al. (2017). "Future word contexts in neural network language models". In: *Proc. of ASRU 2017*, pp. 97–103 (cit. on p. 76).

CommonCrawl (2014). `http://commoncrawl.org/` (cit. on p. 89).

del-Agua, M.A. et al. (Nov. 2014). "The translectures-UPV toolkit". In: *Proc. of Advances in Speech and Language Technologies for Iberian Languages 2014*, pp. 269–278 (cit. on p. 87).

Eldiario.es (2020). `https://www.eldiario.es/` (cit. on p. 89).

ElPeriodico.com (2020). `https://www.elperiodico.com/` (cit. on p. 89).

Giménez, Adrià et al. (2014). "Discriminative Bernoulli HMMs for isolated handwritten word recognition". In: *Pattern Recognition Letters* 35, pp. 157–168. published (cit. on p. 88).

Graves, A. and J. Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks* 18.5-6, pp. 602–610 (cit. on p. 75).

Han, Kyu J, Jing Huang, et al. (2019). "Multi-Stride Self-Attention for Speech Recognition." In: *Proc. of InterSpeech 2019*, pp. 2788–2792 (cit. on p. 100).

Han, Kyu J, Jing Pan, et al. (2021). "Multistream CNN for robust acoustic modeling". In: *Proc. of ICASSP 2021*. IEEE, pp. 6873–6877 (cit. on p. 100).

Hori, Takaaki, Yotaro Kubo, and Atsushi Nakamura (2014). "Real-time one-pass decoding with recurrent neural network language model for speech recognition". In: *Proc. of ICASSP 2014*. IEEE, pp. 6364–6368 (cit. on p. 77).

Huang, Z. et al. (2014). "Cache based recurrent neural network language model inference for first pass speech recognition". In: *Proc. of ICASSP 2014*, pp. 6354–6358 (cit. on p. 77).

Irie, Kazuki et al. (2019). "Language Modeling with Deep Transformers". In: *Proc. of InterSpeech 2019*, pp. 3905–3909 (cit. on pp. 76, 93).

Jorge, Javier et al. (2018). "MLLP-UPV and RWTH Aachen Spanish ASR systems for the IberSpeech-RTVE 2018 speech-to-text transcription challenge". In: *Proc of IberSpeech 2018*, pp. 257–261 (cit. on p. 100).

Jorge, Javier, Adrià Giménez, Javier Iranzo-Sánchez, Jorge Civera, et al. (2019). "Real-time One-pass Decoder for Speech Recognition Using LSTM Language Models". In: *Proc. of InterSpeech 2019*, pp. 3820–3824 (cit. on pp. 77, 86).

Jorge, Javier, Adrià Giménez, Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, et al. (2020). "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP 2020*, pp. 7814–7818 (cit. on pp. 77, 83).

Jozefowicz, R. et al. (2016). "Exploring the limits of language modeling". In: *arXiv preprint arXiv:1602.02410* (cit. on p. 76).

Kombrink, Stefan et al. (2011). "Recurrent Neural Network based language modeling in meeting recognition". In: *Proc. of Interspeech 2011*, pp. 2877–2880 (cit. on pp. 76, 84).

Langzhou Chen and K. K. Chin (2008). "Efficient language model look-ahead probabilities generation using lower order LM look-ahead information". In: *Proc. of ICASSP*, pp. 4925–4928 (cit. on p. 85).

Lee, K. et al. (2015). "Applying GPGPU to recurrent neural network language model based fast network search in the real-time LVCSR". In: *Proc. of InterSpeech 2015*, pp. 2102–2106 (cit. on p. 77).

Lee, Kyungmin et al. (2018). "Accelerating recurrent neural network language model based online speech recognition system". In: *Proc. of ICASSP 2018*, pp. 5904–5908 (cit. on p. 77).

Lison, Pierre and Jörg Tiedemann (May 2016). "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". In: *Proc. of LREC 2016*, pp. 923–929 (cit. on p. 89).

Lleida, Eduardo et al. (2018). *RTVE2018 database description*. `http://catedrartve.unizar.es/reto2018/RTVE2018DB.pdf` (cit. on p. 87).

Miao, Haoran et al. (2020). "Transformer-based online CTC/attention end-to-end speech recognition architecture". In: *Proc. of ICASSP 2020*. IEEE, pp. 6084–6088 (cit. on p. 76).

Mnih, Andriy and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models". In: *Proc. of ICML* (cit. on p. 88).

Mohamed, A. et al. (2015). "Deep bi-directional recurrent networks over spectral windows". In: *Proc. of ASRU 2015*, pp. 78–83 (cit. on p. 76).

Moritz, N. et al. (2020). "Streaming automatic speech recognition with the transformer Model". In: *Proc. of ICASSP 2020*, pp. 6074–6078 (cit. on pp. 76, 100).

News Crawl corpus (WMT workshop) (2015). `http://www.statmt.org/wmt15/translation-task.html` (cit. on p. 89).

Ney, Hermann and Stefan Ortmanns (2000). "Progress in dynamic programming search for LVCSR". In: *Proc. of IEEE 2000* 88.8, pp. 1224–1240 (cit. on p. 84).

Nguyen, Thai-Son et al. (2020). "High Performance Sequence-to-Sequence Model for Streaming Speech Recognition". In: *Proc. of InterSpeech 2020* (cit. on p. 76).

Nolden, D. et al. (2011). "Exploiting sparseness of backing-off language models for efficient look-ahead in LVCSR". In: *Proc. of ICASSP*, pp. 4684–4687 (cit. on p. 85).

Nolden, David (Apr. 2017). "Progress in Decoding for Large Vocabulary Continuous Speech Recognition". PhD thesis. Computer Science Department RWTH Aachen University Aachen (Germany): RWTH Aachen University (cit. on pp. 76, 81, 82).

Ogawa, A. et al. (2018). "Rescoring N-Best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model". In: *Proc. of ICASSP 2018*, pp. 6099–6103 (cit. on p. 76).

Ott, Myle, Sergey Edunov, Alexei Baevski, et al. (2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proc. of NAACL-HLT 2019*, pp. 48–53 (cit. on p. 89).

Panayotov, V. et al. (2015). "Librispeech: an ASR corpus based on public domain audio books". In: *Proc. of ICASSP 2015*, pp. 5206–5210 (cit. on pp. 77, 87).

Park, Daniel S. et al. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. of InterSpeech 2019*, pp. 2613–2617 (cit. on p. 88).

Rousseau, Anthony et al. (2014). "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks." In: *Proc. of LREC 2014*, pp. 3935–3939 (cit. on pp. 77, 87, 88).

Shi, Yongzhe et al. (2014). "Efficient One-Pass Decoding with NNLM for Speech Recognition". In: *IEEE Signal Processing Letters* 21.4, pp. 377–381 (cit. on pp. 77, 85).

Singh, Mittul et al. (2017). "Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition". In: *Proc. of InterSpeech 2017*, pp. 2720–2724 (cit. on p. 77).

Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech 2002*, pp. 901–904 (cit. on p. 88).

UFAL Medical Corpus (2017). `http://ufal.mff.cuni.cz/ufal_medical_corpus` (cit. on p. 89).

Wikipedia (2020). `https://www.wikipedia.org/` (cit. on p. 89).

Xue, S. and Z. Yan (2017). "Improving latency-controlled BLSTM acoustic models for online speech recognition". In: *Proc. of ICASSP*, pp. 5340–5344 (cit. on p. 76).

Young, S. J., J. J. Odell, and P. C. Woodland (1994). "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proc. of Workshop on Human Language Technology 1994*, pp. 307–312 (cit. on p. 88).

Yu, Dong and Li Deng (2014). *Automatic speech recognition: A deep learning approach*. Springer Publishing Company, Incorporated. ISBN: 1447157788 (cit. on p. 75).

Zeyer, A., R. Schlüter, and H. Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Proc. of InterSpeech 2016*, pp. 3424–3428 (cit. on pp. 76, 78).

Zeyer, Albert et al. (2017). "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition". In: *Proc. of ICASSP 2017*. IEEE, pp. 2462–2466 (cit. on pp. 75, 87, 88).

Zhang, Yu et al. (2016). "Highway long short-term memory RNNs for distant speech recognition". In: *Proc. of ICASSP 2016*, pp. 5755–5759 (cit. on pp. 76, 100).

Zhou, W. et al. (2020). "The RWTH ASR system for Ted-Lium Release 2: Improving hybrid HMM with specAugment". In: *Proc. of ICASSP*, pp. 7839–7843 (cit. on p. 100).

5 MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge

Jorge, Javier; Giménez, Adrià; Baquero-Arnal, Pau; Iranzo-Sánchez, Javier; Pérez-González-de-Martos, Alejandro; Garcés Díaz-Munío, Gonçal V; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Proc. of IberSPEECH 2020, pp. 118–122

Valladolid (Spain)

DOI 10.21437/IberSPEECH.2021-25

24-25 March 2021

MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge

Jorge, Javier; Giménez, Adrià; Baquero-Arnal, Pau; Iranzo-Sánchez, Javier; Pérez-González-de-Martos, Alejandro; Garcés Díaz-Munío, Gonçal V; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons

Abstract

This paper describes the automatic speech recognition (ASR) systems built by the MLLP-VRAIN research group of Universitat Politècnica de València for the Albayzin-RTVE 2020 Speech-to-Text Challenge.

The primary system (*p-streaming_1500ms_nlt*) was a hybrid BLSTM-HMM ASR system using streaming one-pass decoding with a context window of 1.5 seconds and a linear combination of an n-gram, a LSTM, and a Transformer language model (LM). The acoustic model was trained on nearly 4,000 hours of speech data from different sources, using the MLLP's transLectures-UPV toolkit (TLK) and TensorFlow; whilst LMs were trained using SRILM (n-gram), CUED-RNNLM (LSTM), and Fairseq (Transformer), with up to 102G tokens. This system achieved 11.6% and 16.0% WER on the *test-2018* and *test-2020* sets, respectively. As it is streaming-enabled, it could be put into production environments for automatic captioning of live media streams, with a theoretical delay of 1.5 seconds.

Along with the primary system, we submitted three contrastive systems. From these, we highlight the system *c2-streaming_600ms_t* that, following the same configuration of the primary one, but using a smaller context window of 0.6 seconds and a Transformer LM, scored 12.3% and 16.9% WER points respectively on the same test sets, with a measured empirical latency of 0.81±0.09 seconds (mean±stdev). This is, we obtained state-of-the-art latencies for high-quality automatic live captioning with a small WER degradation of 6% relative.

5.1 Introduction

This paper describes the participation of the *Machine Learning and Language Processing* (MLLP) research group from the *Valencian Research Institute for Artificial Intelligence* (VRAIN), hosted at the *Universitat Politècnica de València* (UPV), in the Albayzin-RTVE 2020 Speech-to-Text (S2T) Challenge.

Live audio and video streams such as TV broadcasts, conferences, lectures, as well as general-public video streaming services (e.g. Youtube) over the Internet have increased dramatically in recent years because of the advances in networking with high speed connections and proper bandwidth. Also, due to the COVID-19 pandemic, video meeting/conferencing platforms have experienced an exponential growth of usage, as public and private companies have leveraged teleworking for their employees to comply with the social distancing measures recommended by health authorities.

Automatic transcription and translation of such audio streams is a key feature in a globalized and interconnected world, in order to reach wider audiences or to ensure proper understanding between native and non-native speakers, depending on the use-case. Also, public governments are enforcing TV broadcasters by law to provide accessibility options to people with hearing disabilities, with a yearly increasing amount of contents to be captioned at a minimum (*RD 1494/2007, del 12 de noviembre* 2007; *Llei 1/2006, de 19 d'abril, GVA* 2006).

Some TV broadcasters and other live streaming services have assumed manual transcription from scratch of live audio or video streams, as an initial solution to comply with the current legislation, and/or to satisfy user expectations. However, it is a really hard task for professional linguists that, under very stressful conditions, are very prone to generate captioning errors. Besides, it is difficult to scale up such a service, as in these organizations, the amount of human resources devoted to this particular task is typically scarce.

Due to these reasons, the need and demand for high-quality real-time streaming Automatic Speech Recognition (ASR) has increased drastically in the last years. Automatic live audio stream subtitling enables professional linguists to correct live transcripts provided by these ASR systems, if they are not publishable as they come. This would dramatically expedite their productivity and significantly reduce the probability of producing transcription errors. However, the application of state-of-the-art ASR technology to video streaming is a highly complex and challenging task due to real-time and low-latency recognition constraints.

The MLLP-VRAIN, being aware of these demands from the society, have focused its research efforts in the past two years on streaming ASR. This work aims to disseminate our latest developments in this area, showing how our hybrid ASR technology can be successfully applied under streaming conditions, by providing high-quality transcriptions and state-of-the-art system latencies on real-life tasks such as the RTVE (*Radio Televisión Española*) database. Therefore, our partic-

ipation in the Albayzin-RTVE 2020 S2T Challenge consisted on the submission of a primary, performance-focused streaming ASR system, plus three contrastive systems: two latency-focused streaming ASR systems, and one conventional offline ASR system.

The rest of the paper is structured as follows. First, Section 5.2 briefly describes the Albayzin-RTVE 2020 S2T Challenge and the RTVE databases provided by the organizers. Next, Section 5.3 provides a detailed description of our participant ASR systems. Finally, Section 5.4 gives a summary of the work plus some concluding remarks.

5.2 Challenge description and databases

The Albayzin-RTVE 2020 Speech-To-Text Challenge consists of automatically transcribing different types of TV shows from the RTVE Spanish public TV station, and the assessment of ASR system performance in terms of Word Error Rate (WER) by comparing those automatic transcriptions with correct reference transcriptions (Lleida et al. 2020a).

The MLLP-VRAIN participated in the 2018 edition of the challenge (Jorge et al. 2018) in a joint collaboration with the *Human Language Technology and Pattern Recognition* (HLTPR) research group from the *RWTH Aachen University*. The evaluation was carried out on the RTVE2018 database (Lleida et al. 2018), that includes 575 hours of audio from 15 different TV shows broadcasted between 2015 and 2018. This database is allocated into four sets: *train, dev1, dev2* and *test* (*test-2018*). Our systems won in both the open-condition and closed-condition tracks (al. 2019), scoring 16.5% and 22.0% WER points respectively in the *test-2018* set.

For the 2020 edition of the challenge, the participation has been limited to a single open-condition track, and system evaluations have been carried out over the *test* (*test-2020*) set from the RTVE2020 database, which includes 78.4 hours from 15 different TV shows broadcasted between 2018 and 2019 (Lleida et al. 2020b).

5.3 MLLP-VRAIN Systems

In this section we describe the hybrid ASR systems developed by the MLLP-VRAIN that participated in the Albayzin-RTVE 2020 S2T Challenge.

Acoustic Modelling

Our acoustic models (AM) were trained using 205 filtered speech hours from the *train* set (187h) and our internal *dev1-train* set (18h), as in (Jorge et al. 2018), plus about 3.7K hours of other resources crawled from the Internet. Table 2.22 summarises all training datasets along with their total duration (in hours).

Table 2.22: Transcribed Spanish speech resources for AM training.

Resource	Duration (h)
Internal: entertainment	2932
Internal: educational	406
Internal: user-generated content	202
Internal: parliamentary data	158
Voxforge (*Voxforge* 2018)	21
RTVE2018: *train*	187
RTVE2018: *dev1-train*	18
TOTAL	3924

From this data, first, we extracted 16-dimensional MFCC features plus first and second derivatives (48-dimensional feature vectors) every 10ms to train a context-dependent feed-forward DNN-HMM with three left-to-right tied states using the transLectures-UPV toolkit (TLK) (del-Agua et al. 2014). The state-tying schema followed a phonetic decision tree approach (Young, Odell, and Woodland 1994) that produced 10K tied states. Then, feed-forward models were used to bootstrap a BLSTM-HMM AM, trained with 85-dimensional filterbank features, following the procedure described in (Zeyer et al. 2017). The BLSTM network was trained using both TLK and TensorFlow (Abadi, Agarwal, et al. 2015), and had 8 bidirectional hidden layers with 512 LSTM cells per layer and direction. As in (Zeyer et al. 2017), we performed chunking during training by considering a context to perform back-propagation through time to a window size of 50 frames. Additionally, SpecAugmentation was applied by means of time and frequency distortions (Park et al. 2019).

Language Modelling

Regarding language modelling, we trained count-based (n-gram) and neural-based (LSTM, Transformer) Language Models (LMs) to perform one-pass decoding with different linear combinations of them (Jorge, Giménez, Iranzo-Sánchez, Civera, et al. 2019), using the text data sources and corpora described in Table 2.23.

On the one hand, we trained 4-gram LMs using SRILM (Stolcke 2002) with all text resources plus the Google-counts v2 corpus (Lin et al. 2012), accounting for 102G running words. The vocabulary size was limited to 254K words, with an OOV ratio of 0.6% computed over our internal development set.

On the other hand, regarding neural LMs, we considered the LSTM and Transformer architectures. In both cases, LMs were trained using a 1-gigaword subset randomly extracted from all available text resources, except Google-counts. Their vocabulary was defined as the intersection between the n-gram vocabulary (254K words) and that derived from the aforementioned training subset. We did this to avoid having zero probabilities for words that are present in the system vocabulary but not in the training subset. This is taken into account when computing perplexities by renormalizing the unknown-word score accordingly.

Specific training details for each neural LM architecture are as follows. Firstly, LSTM LMs were trained using the CUED-RNNLM toolkit (Xie Chen, Liu, et al. 2016). Noise Contrastive Estimation (NCE) criterion (Mnih and Teh 2012) was used to speed up model training, and the normalization constant learned from training was used during decoding (X. Chen et al. 2015). Based on the lowest perplexity observed on our internal development set, we selected as final model that with a 256-unit embedding layer and two hidden LSTM layers of 2048 units. Secondly, Transformer LMs (TLMs) were trained using a customized version of the FairSeq toolkit (Ott, Edunov, Baevski, et al. 2019), selecting the following configuration that minimized perplexity in our internal development set: 24-layer network with 768 units per layer, 4096-unit FFN, 12 attention heads, and an embedding of 768 dimensions. These models were trained until convergence with batches limited to 512 tokens, 512 sentences, and 512 words per sentence. Parameters were updated every 32 batches. During inference, Variance Regularization (VR) was applied to speed up the computation of the TLM score (Baquero-Arnal et al. 2020).

Decoding strategy

Our hybrid ASR systems follow a real-time one-pass decoding by means of a History Conditioned Search (HCS) strategy, as described in (Jorge, Giménez, Iranzo-Sánchez, Civera, et al. 2019). This approach allows us to benefit from the direct usage of additional LMs during decoding while satisfying real-time constraints. This decoding strategy introduces two additional and relevant parameters to control the trade-off between Real Time Factor (RTF) and WER: LM history recombination (LMHR), and LM histogram prunning (LMHP). The static look-ahead table, needed by the decoder to use pre-computed look-ahead LM scores, was generated from a prunned version of the n-gram LM.

For streaming ASR, as the full sequence (context) is not available during decoding, BLSTM AMs are queried with a sliding, overlapping context window of limited size over the input sequence, averaging outputs of all windows for each frame to obtain the corresponding acoustic score (Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020). The size of the context window (in frames or seconds) is set in decoding, and defines the theoretical latency of the system. This limitation of the context prevents us to perform a Full Sequence Normaliza-

Table 2.23: 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 (Lison and Tiedemann 2016)	212635	1146861	1576
UFAL (*UFAL Medical Corpus* 2017)	92873	910728	2179
Wikipedia (*Wikipedia* 2020)	32686	586068	3373
UN (Callison-Burch et al. 2012)	11196	343594	381
News Crawl (*News Crawl corpus (WMT workshop)* 2015)	7532	198545	648
Internal: entertainment	4799	59235	307
eldiario.es (*Eldiario.es* 2020)	1665	47542	247
El Periódico (*ElPeriodico.com* 2020)	2677	46637	291
Common Crawl (*CommonCrawl* 2014)	1719	41792	486
Internal: parliamentary data	1361	35170	126
News Commentary (*News Crawl corpus (WMT workshop)* 2015)	207	5448	83
Internal: educational	87	1526	35
TOTAL	369434	3423146	5785
Google-counts v2 (Lin et al. 2012)	-	97447282	3693

tion (FSN), that is typically applied under the off-line setting. Instead, we applied the Weighted Moving Average (WMA) technique, that uses the content of the current context window to update normalization statistics on-the-fly, weighted by previous context from past windows with an α parameter (Jorge, Giménez, Silvestre-Cerdà, et al. 2022). Finally, as Transformer LMs have the inherent capacity of attending to potentially infinite word sequences, history is limited to a given maximum number of words, in order to meet the strict computational time constraints imposed by the streaming scenario (Baquero-Arnal et al. 2020). By applying all these modifications, our decoder acquires the capacity to deliver live transcriptions for incoming audio streams of potentially infinite length, with latencies lower-bounded by the context window size.

Experiments and results

To carry out our experiments, we used the development and test sets from the RTVE2018 database. More precisely, we devoted our internal *dev1-dev* set (Jorge et al. 2018) for development purposes, whilst *dev2* and *test-2018* were dedicated to test ASR performance. Finally, *test-2020* was the blind test used by the organisation to rank the participant systems. Table 2.24 provides basic statistics of these sets.

Table 2.24: Basic statistics of development and tests sets of RTVE databases, including our internal *dev1-dev* set: total duration (in hours), number of files, average duration of samples in seconds plus-minus standard deviation ($d_\mu \pm \sigma$), 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. Table 2.25 shows the PPLs of these interpolations, along with the optimum LM weights that minimized PPL in the *dev1-dev* set. The Transformer LM provides significant lower perplexities in all cases, and accordingly, takes very high weight values when combined with other LMs. Indeed, the TLM in isolation already delivers a strong perplexity baseline value of 63.3, while the maximum PPL improvement is of just 6% relative when all three LMs are combined.

Table 2.25: Perplexity (PPL) and interpolation weights, computed over the *dev1-dev* set, of all possible linear combinations of n-gram (ng), LSTM (ls) and Transformer (tf) LMs.

LM comb.	PPL	Weights(%)
ng	179.5	-
ls	98.4	-
tf	63.3	-
ng + ls	93.2	15 + 85
ng + tf	61.6	6 + 94
ls + tf	60.7	13 + 87
ng + ls + tf	59.5	5 + 10 + 85

Second, we tuned decoding parameters to provide a good WER-RTF tradeoff on *dev1-dev*, with the hard constraint of RTF<1 to ensure a real-time processing of the input. From these hyperparameters, we highlight, due to their relevance, LMHR=12, LMHP=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/no-speech segments as in (Jorge et al. 2018), followed by a real-time one-pass decoding with our BLSTM-HMM AM, using a FSN 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 for this competition, we explored streaming-related decoding parameters to optimize WER on *dev1-dev*, using the BLSTM-HMM AM and a linear combination of all three LMs. This resulted on using a context window size of 1.5 seconds and α=0.95 for the WMA normalization technique. This configuration defined our primary system, identified as *p-streaming_1500ms_nlt*, that showed WER rates of 11.6 and 16.0 in *test-2018* and *test-2020*, respectively. It is important to note that this system does not integrate any VAD module. This task is implicitly carried out by the decoder via the non-speech model of the BLSTM-HMM AM.

A small change on the configuration of the primary system, consisting on the removal of the LSTM LM from the linear interpolation, defined the contrastive system no. 1, identified as *c1-streaming_1500ms_nt*. The motivation behind this change is that the computation of LSTM LM scores is quite expensive in computational terms, and its contribution to PPL is negligible with respect to the n-gram LM + TLM combination (3% relative improvement). Hence, for the sake of system latency stability, we obtained nearly no degradation in terms of WER: 11.6 and 16.1 points in *test-2018* and *test-2020*, respectively.

Both streaming ASR systems, *p-streaming_1500ms_nlt* and *c1-streaming_1500ms_nt*, share the same theoretical latency of 1.5 seconds, as it is determined by the context window size. As stated in Section 5.3, this parameter can be adjusted in decoding time. This allows us to configure the decoder for lower latency responses or better transcription quality. Hence, our last commitment for this challenge was to find a proper system configuration that could provide state-of-the-art, stable latencies with minimal WER degradation. Figure 2.16 illustrates the evolution of WER on *dev1-dev* as a function of the context window size, limited to one second at maximum. As we focused on gauging AM performance, we used the n-gram LM in isolation for efficiency reasons. At the light of the results, we chose a window size of 0.6 seconds, as it brings a good balance between transcription quality and theoretical latency.

The last step to set up our latency-focused streaming system was to measure WER and empirical latencies as a function of different prunning parameters and LM combinations. In our experiments, latency is measured as the time elapsed between the instant at which an acoustic frame is generated, and the instant at it is fully processed by the decoder. We provide latency figures at the dataset level,

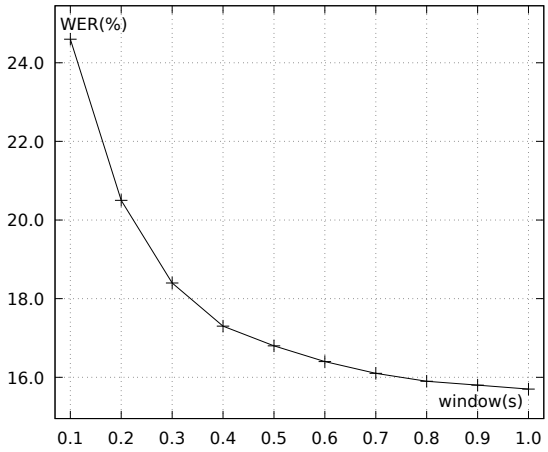

Figure 2.16: WER as a function of context window size (in seconds) for the streaming setup, computed over the *dev1-dev* set.

computed as the average of the latencies observed at the frame level on the whole dataset. Figure 2.17 shows WER vs mean empirical latency figures, computed over *dev1-dev*, with different prunning parameter values, and comparing the LM combinations that include the Transformer LM. These measurements were run on an Intel i7-3820 CPU @ 3.60GHz, with 64GB of RAM and a RTX 2080 Ti GPU card. On the one hand, we can see how combinations involving LSTM LMs are systematically shifted rightwards w.r.t. other combinations. This means that the LSTM LM has a clear negative impact on system latency, with little to no effect on system quality. This evidence corroborates our decision of removing the LSTM LM to define our contrastive system *c1-streaming_1500ms_nt*. On the other hand, TLM alone generally provides a good baseline that is slightly improved in terms of WER if we include the other LMs. However, this comes with the cost of increasing latency. Hence, we selected the Transformer LM in isolation for our final latency-focused streaming system. This system was our contrastive system no. 2, identified as *c2-streaming_600ms_t*. Its empirical latency on *dev1-dev* was 0.81±0.09 seconds (mean±stdev), and its performance was 12.3 and 16.9 WER points in *test-2018* and *test-2020*, respectively. This is, with just a very small relative WER degradation of 6% w.r.t. the primary system, we got state-of-the-art (mean=0.81s) and very stable (stdev=0.09s) empirical latencies. This system has a baseline consumption (when idle) of 9GB RAM and 3.5GB GPU memory (on a single GPU card), adding 256MB RAM and one CPU thread per each decoding (audio stream). For instance, the decoding of four simultaneous audio streams in a single machine would use four CPU threads, 10GB RAM and 3.5GB GPU memory.

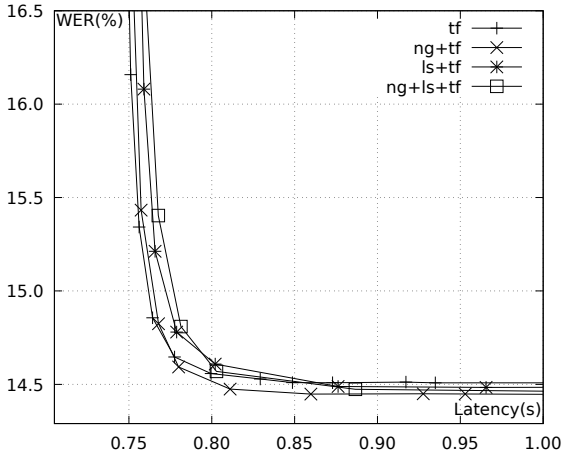

Figure 2.17: WER versus mean empirical latency (in seconds) on *dev1-dev*, measured with different prunning parameters, and considering only interpolation schemes that include TLM.

Table 2.26 summarises the results obtained with all the four participant ASR systems in the *dev2*, *test-2018* and *test-2020* sets, and adds the results obtained with our 2018 open-condition system for comparison. On the one hand, surprisingly, the off-line system is surpassed by the three streaming ones in *test-2020*, by up to 1.1 absolute WER points (6% relative). We believe that this is caused, first, by an improvable VAD module, based on Gaussian Mixture HMMs, that, in our experience, suffers from false negatives (speech segments labelled as non-speech). As the non-speech model was trained with music and noise audio segments, and given the inherent limitations of GMMs, it is likely to misclassify speech passages with loud background music and noise (often present in TV programmes) as non-speech. Second, the FSN technique might not be appropriate for some types of TV shows, as local acoustic condition changes become diluted in the full-sequence normalization, and acoustic scores computed for those frames may present some perturbations that can degrade system performance at that point. On the other hand, it is remarkable that our primary 2020 system significantly outperforms the 2018 winning system by 28% relative WER points on both *dev2* and *test-2018* (25% in the case of our latency-focused system *c2-streaming_600ms_t*), while adding the novel streaming capability at the same time.

All these streaming ASR systems can be easily put into production environments using our custom gRPC-based server-client infrastructure[3]. Indeed, ASR systems comparable to *c2-streaming_600ms_t* and *c1-streaming_1500ms_nt* are already

[3]https://mllp.upv.es/git-pub/jjorge/MLLP_Streaming_API

Table 2.26: 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.

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 (Jorge et al. 2018)	15.6	16.5	-

in production at our Transcription and Translation Plarform (TTP)[4] for streaming and off-line processing, respectively. Both can be freely tested using our public APIs, accessible via TTP.

5.4 Conclusions

In this paper we have described our four ASR systems that participated in the Albayzin-RTVE 2020 Speech-to-Text Challenge. The primary one, a streaming-enabled performance-focused hybrid ASR system (*p-streaming_1500ms_nlt*) provided a good score of 16.0 WER points in the *test-2020* set, and a remarkable 28% relative WER improvement over the 2018 winning ASR system on *test-2018*, with a theoretical latency of 1.5 seconds. Nearly the same performance was delivered by our first contrastive system (*c1-streaming_1500ms_nt*): 16.1 WER points on *test-2020*, at a significant lower computational cost. In pursuit of low, state-of-the-art system latencies, our second contrastive system (*c2-streaming_600ms_t*) provided a groundbreaking WER-latency balance, with a solid performance of 16.9 WER points on *test-2020* at an empirical latency of 0.81±0.09 seconds (mean±stdev). Finally, our contrastive off-line ASR system with VAD (*c3-offline*) provides the highest, yet still competitive, WER mark of 17.1 points, attributable to an improvable VAD module and to the limitations of FSN when dealing with local acoustic condition changes.

With a configurable system latency in decoding time, our ASR technology offers the flexibility to produce fast system responses for streaming applications, or to generate maximum quality transcriptions whenever hard time constraints do not apply. Also, results demonstrate that our streaming ASR technology is mature enough to be systematically put into production environments for high-quality automatic live captioning in TV stations, distance learning, conferencing platforms, or general-purpose video/audio streaming services, among others.

[4] https://ttp.mllp.upv.es/

References

Abadi, Martín, Ashish Agarwal, et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (cit. on p. 110).

al., Eduardo Lleida et (2019). "Albayzin 2018 Evaluation: The IberSpeech-RTVE Challenge on Speech Technologies for Spanish Broadcast Media". In: *Applied Sciences* 9.24, p. 5412 (cit. on p. 109).

Baquero-Arnal, Pau et al. (2020). "Improved Hybrid Streaming ASR with Transformer Language Models". In: *Proc. of InterSpeech 2020*, pp. 2127–2131 (cit. on pp. 111, 112).

Callison-Burch, Chris et al. (2012). "Findings of the 2012 Workshop on Statistical Machine Translation". In: *Proc. of WMT*, pp. 10–51 (cit. on p. 112).

Chen, X. et al. (2015). "Improving the training and evaluation efficiency of recurrent neural network language models". In: *Proc. of ICASSP 2015*, pp. 5401–5405 (cit. on p. 111).

Chen, Xie, Xunying Liu, et al. (2016). "CUED-RNNLM – An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *Proc. of ICASSP 2016*. Shanghai, China, pp. 6000–6004 (cit. on p. 111).

CommonCrawl (2014). `http://commoncrawl.org/` (cit. on p. 112).

del-Agua, M.A. et al. (Nov. 2014). "The translectures-UPV toolkit". In: *Proc. of Advances in Speech and Language Technologies for Iberian Languages 2014*, pp. 269–278 (cit. on p. 110).

Eldiario.es (2020). `https://www.eldiario.es/` (cit. on p. 112).

ElPeriodico.com (2020). `https://www.elperiodico.com/` (cit. on p. 112).

Jorge, Javier et al. (2018). "MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge". In: *Proc. of IberSPEECH 2018*, pp. 257–261 (cit. on pp. 109, 110, 113, 114, 117).

Jorge, Javier, Adrià Giménez, Javier Iranzo-Sánchez, Jorge Civera, et al. (2019). "Real-time One-pass Decoder for Speech Recognition Using LSTM Language Models". In: *Proc. of InterSpeech 2019*, pp. 3820–3824 (cit. on pp. 110, 111).

Jorge, Javier, Adrià Giménez, Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, et al. (2020). "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP 2020*, pp. 7814–7818 (cit. on p. 111).

Jorge, Javier, Adrià Giménez, Joan Albert Silvestre-Cerdà, et al. (2022). "Live Streaming Speech Recognition Using Deep Bidirectional LSTM Acoustic Models and Interpolated Language Models". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30, pp. 148–161 (cit. on p. 112).

Lin, Yuri et al. (2012). "Syntactic annotations for the Google Books Ngram Corpus". In: *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics (cit. on pp. 110, 112).

Lison, Pierre and Jörg Tiedemann (May 2016). "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". In: *Proc. of LREC 2016*, pp. 923–929 (cit. on p. 112).

Llei 1/2006, de 19 d'abril, GVA (2006) (cit. on p. 108).

Lleida, Eduardo et al. (2018). *RTVE2018 database description*. http://catedrartve.unizar.es/reto2018/RTVE2018DB.pdf (cit. on p. 109).
– (2020a). *IberSPEECH-RTVE 2020 Speech to Text Transcription Challenge*. http://catedrartve.unizar.es/reto2020/EvalPlan-S2T-2020-v1.pdf (cit. on p. 109).
– (2020b). *RTVE2020 database description*. http://catedrartve.unizar.es/reto2020/RTVE2020DB.pdf (cit. on p. 109).
Mnih, Andriy and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models". In: *Proc. of ICML* (cit. on p. 111).
News Crawl corpus (WMT workshop) (2015). http://www.statmt.org/wmt15/translation-task.html (cit. on p. 112).
Ott, Myle, Sergey Edunov, Alexei Baevski, et al. (2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proc. of NAACL-HLT 2019*, pp. 48–53 (cit. on p. 111).
Park, Daniel S. et al. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. of InterSpeech 2019*, pp. 2613–2617 (cit. on p. 110).
RD 1494/2007, del 12 de noviembre (2007) (cit. on p. 108).
Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech 2002*, pp. 901–904 (cit. on p. 110).
UFAL Medical Corpus (2017). http://ufal.mff.cuni.cz/ufal_medical_corpus (cit. on p. 112).
Voxforge (2018). http://www.voxforge.org (cit. on p. 110).
Wikipedia (2020). https://www.wikipedia.org/ (cit. on p. 112).
Young, S. J., J. J. Odell, and P. C. Woodland (1994). "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proc. of Workshop on Human Language Technology 1994*, pp. 307–312 (cit. on p. 110).
Zeyer, Albert et al. (2017). "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition". In: *Proc. of ICASSP 2017*. IEEE, pp. 2462–2466 (cit. on p. 110).

Chapter 3

General discussion of the results

This chapter will gather the main results to cover the goals established for this thesis and their outreach and impact over the last years during the development of several research projects. The goals for this work were introduced in the first chapter and summarized here for convenience:

- Provide a one-pass ASR system leveraging the neural LM with low RTF.
- Adapt the one-pass neural based system to perform streaming recognition.
- Evaluate the streaming ASR system in production environments.

To address the first goal, an initial version of our one-pass decoder integrating the LSTM LM was submitted to the international competition IberSpeech 2018 in the Speech-to-text track, as it is described in Paper 1. This competition was an excellent opportunity to test the features and pipelines included in our system: the data filtering, the model training (both AM and LM, with the new LSTM LM), and the decoding, including our one-pass algorithm. In the past, our internal toolkit TLK has been used to participate in other international challenges with good results (Miguel Ángel Del-Agua et al. 2015; Miguel Ángel Del-Agua et al. 2016) and we wanted to measure its performance in comparison with the last versions of the commonly used Kaldi ASR toolkit.

Considering this challenge an opportunity to benchmark our decoder, our system was submitted to the closed-condition track, where the data was limited to the RTVE2018 set provided by the organization, as described in (Lleida et al. 2018). The results for this competition and all the details from the participants were reported in (Lleida et al. 2019). This closed-condition track proposed an additional challenge related to the data provided by the organization. The RTVE2018 dataset comprises only 460 hours, with very noisy audio and transcriptions. Thus, aligning and filtering the data, making the most out of it, became the critical factor in obtaining good results in this track. For this reason, while for the open-condition there were seven teams, where participants are not limited in the amount of data to use, only three of them participated in the closed-condition track.

Table 3.1 includes the results in WER for the blind test 2018 obtained by the participants in this closed track plus the open track's winner for comparison. It is important to remark that the SIGMA team used additional human supervision to increase the amount of data correctly transcribed. Due to this, the team was disqualified from the competition, but their results are included here to reflect all three submissions.

As results show, our one-pass decoder using the LSTM LM directly obtained the best WER when considering fully automatic pipelines, as the competition required. Even considering the team that used up to 350 hours of supervised data (vs. our 205 hours of automatic aligned and filtered data), our system showed very competitive results. In fact, the difference was around 10% of WER reduc-

Table 3.1: WER of the participants of the closed-condition track from IberSpeech 2018 on *test-2018*.

Team	WER
MLLP-RWTH	21.96
VICOMTECH-PRHLT (Arzelus et al. 2018)	22.22
SIGMA (Perero-Codosero et al. 2018)	19.57
MLLP-RWTH (OPEN WINNER)	16.45

tion with an increase of the manually transcribed data of around 37%. Regarding the technique and tools, VICOMTECH-PRHLT and SIGMA followed the traditional WFST with the Kaldi toolkit, performing a second pass based on a lattice rescoring approach using count-based models. Finally, the RTF of the MLLP system was also analyzed as the speed was one of the main motivations to migrate from two-pass to one-pass decoding strategy. As reported in Paper 1, the primary system (WER-oriented) provided an excellent RTF of 1.5, but by increasing the pruning can be improved easily to get 0.8. This speed improvement comes with a slight increase of WER only of 1.5% relative.

After this public evaluation competing with other NLP teams, obtaining such good and competitive results working below real-time encouraged us to continue developing our ASR system. In Paper 2, this development and in-depth study continued. In this paper, a comprehensive experimental analysis was carried out to compare the aforementioned one-pass decoder with our two-pass version. In this second paper, the one-pass system is described thoroughly, proposing the final version that allowed us to perform real-time one-pass decoding using the LSTM LM and defining the architecture that is currently used in our decoder so far. Paper 2 defines the decoder architecture that changed from a traditional WFST to an HCS that leverages several advanced LM-related techniques. On the other hand, this new architecture enabled the integration of the LSTM LM, achieved thanks to the improvements in the decoding process, including static precomputation of the look-ahead scores, together with the efficient computation of the LM scores with the neural model using the GPU and the self-normalized scores. Finally, advanced pruning techniques helped the system work below real-time, paving the way to our final purpose: providing a competitive streaming system.

The results and conclusions from Paper 1 and Paper 2 were, indeed, that a system based on only one-pass of recognition that benefits from the neural LM can provide competitive results. These results are very similar or even better than a two-pass strategy, with lower RTF, making the whole system a good fit for deployment in a streaming setup. This conclusion leads us to the second goal from this thesis, related to changing the current offline one-pass decoder to work on streaming.

In addition to these academic results, participating in several research projects offered the opportunity to evaluate the system and its behavior in real-life scenarios. In the context of X5Gon project described in the introduction, several ASR systems were developed for Open Educational Resources (OER) transcription in 4 languages (En, Es, Sl, De). Moreover, throughout the X5Gon project, the proposed architecture evolved, making significant progress in offline and streaming ASR. A brief description of the developments will be included in this section, but several reports are publicly available to the reader to extend this information (Jorge et al. 2020; Pérez et al. 2020). In particular, three systems were trained and systematically improved and evaluated on these sets: the English system (evaluated on VideoLectures.NET and poliMedia), the Spanish system (evaluated on poliMedia) and the Slovenian system (evaluated on VideoLectures.Net).

During M30 (February 2020) there was carried out a comparative study between Google Cloud Speech-to-text API and our proposed approach in terms of WER with these evaluation sets. Table 3.2 shows the results obtained for these experiments. In general, our ASR systems provide a relative error reduction of ∼54% with respect to the quality offered by the commercial provider Google Cloud Speech-To-Text web service. Moreover, this figure is even higher in the in-domain tasks for Spanish (54% for pM) and Slovenian (56% for VL). It is worth mentioning that our system can be biased to educational content because of the datasets used for training, but this could also harm the performance in sets such as RTVE. However, our system provided the best performance also for this task.

Table 3.2: WER scores provided by X5Gon ASR systems and Google Cloud Speech-To-Text, on different in-domain and out-domain tasks in English, Spanish, and Slovenian.

ASR	VL En	pM Es	RTVE Es	VL Sl	TED Sl
Google Cloud API	28.6	19.9	49.3	50.0	38.1
X5gon systems	18.8	9.1	13.0	22.0	18.3
Δ%	-34.3	-54.3	-73.6	-56.0	-52.0

The second goal of this thesis is addressed in Paper 3, where a production-ready streaming ASR system is introduced following the base architecture from the offline decoder posed in Paper 2. The main achievement developed during this work was the adaptation of the acoustic input, first modifying the whole-sequence normalization to a limited one. Second, this limited context also involves the AM process, and the input for this model should be appropriately adapted. These are the main contributions of Paper 3, providing the initial adaptation of our offline system to the streaming processing. These improvements were validated with the commonly used benchmarks LibriSpeech and TED-LIUM release 3, in terms of WER, latency and the trade-off between these two measures. The main conclusion drawn from this work was that the proposed system could be adapted to the streaming setup with minimal degradation in WER when compared with the offline system. This degradation comes mainly from the limited context pro-

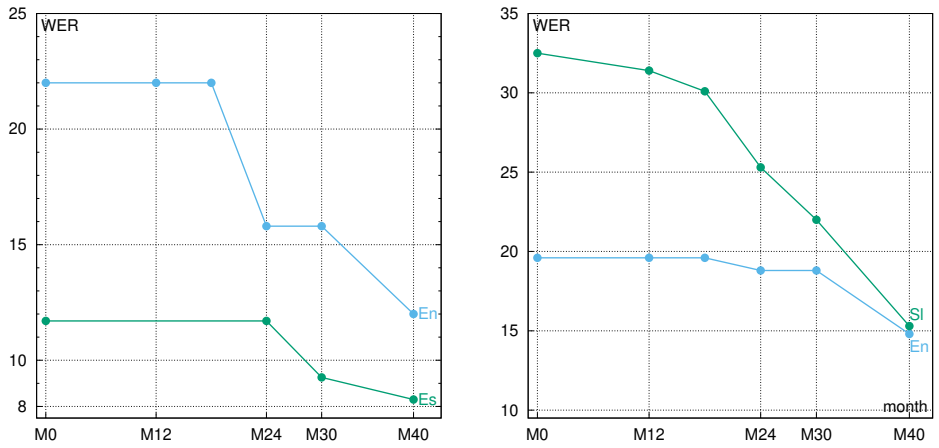

Figure 3.1: WER scores over project month for poliMedia Spanish/English (left) and VideoLectures.Net Slovenian/English (right) transcriptions.

vided to perform the acoustic processing (i.e., normalize the input and process it through the windows-based BLSTM). The proposed streaming ASR system offered competitive WER on these two benchmarks with around ∼1-second latency.

Revising the impact on the projects that coexisted with this thesis, to illustrate the evolution of the system, results and evolution from the X5Gon project are shown in Figure 3.1, reflecting WER in the y-axis along the 40 months of the project in the x-axis. The work described in this thesis contributed to developing; first the systems used during months 24-30, as reported in (Jorge et al. 2020) (Papers 1 and 2), and second from month 30 to the end of the project (Paper 3), as described in (Pérez et al. 2020). As the Figure 3.1 reflects, with the improvements presented in Papers 1 and 2 and the incorporation of new data, the WER improved significantly during the first period for Spanish and Slovenian (M24-M30), even where WER values were already reduced (pM-Es from M24 to M30) and approaching values below 20% for Slovenian, a crucial language in the project. On the other hand, the work done in Paper 3 helped lower these figures, reducing the error rates for English and, even more, for the other two languages, ending up with WER numbers close to 15% or below.

Regarding the streaming systems, Table 3.3 shows comparative results between the offline systems and the streaming low-latency systems prepared for these tasks of the project. It is important to remark that these systems, with a very low latency of 0.8 seconds, only degraded the WER in ranges between 3 to 12% in the worst case. These figures enabled the direct use of these streaming systems to transcribe OER live content in the X5Gon platform, as described in (Pérez et al. 2020).

Table 3.3: WER scores provided by offline and streaming-adapted M40 ASR systems on VideoLectures.NET (in English and Slovene) and poliMedia (in Spanish and English).

	VideoLectures.Net		poliMedia	
ASR system	En	Sl	En	Es
M40 Offline	14.8	15.3	12.0	8.3
M40 Streaming	15.4	15.8	13.4	8.7
Δ%	4.1	3.3	11.7	4.8

Regarding the R&D collaboration and technology transfer agreements related to the development of this thesis, the impressive results with these streaming ASR developments fueled an important landmark with the agreement between UPV and the Corporació Valenciana de Mitjans de Comunicació, with its visible head À Punt, the local TV channel from Valencian Community. This agreement was focused on providing the computer-aided closed-captioning and subtitling of audiovisual content in real-time based on the outcomes of this thesis, among other works done in the MLLP research group. At the beginning of this agreement, several comparative experiments using our general-purpose systems on internal datasets were carried out to illustrate the quality of the Catalan, Spanish and English systems, languages demanded in the agreement mentioned above. Table 3.4 shows these results in terms of WER, providing additionally analogous figures for general-purpose systems commercially available from Google Cloud Speech-To-Text, in its two versions, standard and enhanced (if available) and their relative value with respect to those built at the MLLP with the technological developments proposed up to the work done in Paper 3. The general-purpose MLLP system for Catalan and Spanish provided impressive WER figures with respect to the commercial API from Google, while keeping a similar performance to the enhanced system in English, the Google's more advanced language.

Table 3.4: WER scores provided by MLLP and Google ASR systems on different content (es=Spanish, ca=Catalan, and en=English) and relative improvement.

ASR system	À Punt (ca)	RTVE (es)	PoliMedia (en)
MLLP	20.0	13.1	15.8
Google - standard	45.0	49.0	36.1
Google - enhanced	n/a	n/a	13.3
Δ%	125	274	-15.8

The last goal is addressed in the last part of this thesis, aimed to integrate and evaluate the streaming ASR system in production environments with challenging real-world tasks. Regarding this goal, Papers 4 and 5 mainly cover the work done to achieve it. Paper 4 summarizes all the development carried out during previous papers, as well as proposing new techniques and methods to improve the quality and the speed of the system, as well as using a neural LM based on the Transformer architecture, as follows:

- The windows-based BLSTM AM is widely covered to illustrate the proper adaptation of the model to the streaming setup.
- Another feature normalization technique is proposed to improve the quality and latency of the system, as previous methods required an initial delay, increasing the global latency.
- A new pruning technique for the AM is posed, called Acoustic Model Look-ahead (AMLA), following a similar approach to the look-ahead from the LM.

All these improvements were evaluated on the reference benchmarks LibriSpeech and TED-LIUM and the challenging set RTVE2018. In addition, another task was proposed based on using the TED-LIUM dev/test sets, but instead of using them at the segment level, working at the video level, making the task better suited for evaluating streaming systems. This stream-based version of TED-LIUM allows us to assess the system performance under real conditions, that is, a continuous audio stream. Thus, the speech recognizer provides the output in real-time with low latency with respect to the input stream.

The final remarks from this Paper 4 were that the proposed streaming hybrid ASR system produced a very competitive WER with adjustable latency, with several options to tune the trade-off between these two measures. As it was evaluated on real-world and complex tasks such as RTVE, with many different TV shows offering excellent performance, it can be concluded that the system is ready to be used in production under this scenario. This conclusion is directly aligned with the goals posed for this thesis.

The last paper included in this thesis, Paper 5, closes the loop that started with the IberSpeech2018 challenge, with the participation in the 2021 edition of this international contest. Hence, our last systems and the evolution of technology is contrasted with our initial one-pass architecture. In this regard, the final system proposed in Paper 4 was tuned to this task to obtain several setups: the streaming setup both with the best WER with reasonable latency and the best WER/latency trade-off, and the previous offline pipeline with pre-segmented utterances.

This IberSpeech2020 gave the system another opportunity to be tested against an even more challenging test set (compared with the test from 2018), with more diverse and demanding TV shows. This time, there was only an open-condition track where seven teams participated with their submissions, but only five submitted a system paper with the details of their submission. Table 3.5 shows the best results per team along with our submissions, indicating the ASR toolkit, the training data, and WER on the blind test set (test2020).

In addition to these results, there are also included results from our recent publication (Baquero-Arnal et al. 2022). In this work there were carried out a comparison against the others submissions in a closed-condition fashion. This results

of this comparison highlighted the quality of our toolkit and decoder isolating the training data differences. For this comparison, only the data from the RTVE2018 edition was used. Additionally, there are included results with this closed system plus the open LM with all the data available. This comparison illustrates how the system can be improved easily with this kind of text data, as it is easily obtained from open resources, unlike properly transcribed audio data.

Table 3.5: WER, toolkit, decoder type and data in hours of the participant systems on *test-2020*, including the closed-conditioned version from our MLLP-VRAIN submission (str=streaming-ready).

Participant	Toolkit	Decoder	Data (h)	WER
MLLP-VRAIN	TLK	One-pass (str)	3.924	**16.04**
BIOMETRICVOX (Font and Grau 2021)	Kaldi	Multi-pass	1.030	30.26
BCN2BRNO (Kocour et al. 2021)			780	23.33
VICOMTECH (Alvarez et al. 2021)			743	19.28
SIGMA (Perero-Codosero et al. 2021)			615	27.68
MLLP-VRAIN (closed)			205	23.10
+ open LM (Baquero-Arnal et al. 2022)				19.90

The results showed that our streaming one-pass decoder could achieve a very competitive WER without any additional step or module (i.e., VAD or lattice-rescoring). It is important to remark that this decoder works at low latency, adjustable with minor WER degradation, ranging from 0.8 to 1.5 seconds, as reported in Papers 3,4, and 5. The relative difference between the results with our whole dataset (3.9K hour) vs. the internal closed-condition task (205 hours) is worth noting. Surprisingly, reducing the acoustic data by almost 95% involved a WER increase of 7 absolute points in the closed-condition alone and only 3.9 absolute when including the open LM. This clearly illustrates the direct impact of including the neural LM on the streaming one-pass decoder, providing similar WER figures compared to having much more acoustic data (a factor of 19 times more data). These figures reflect the accuracy and speed achieved through the improvements related to the goals of this thesis, reinforcing the overall quality of the whole system since the 2018 edition.

Despite the fact that all the work was made in collaboration with other members of the MLLP group, the theoretical framework, developments, experimentation and evaluations presented along this thesis have been carried out primarily or as co-author by the author of this work. Additionally, there were other scientific contributions that were not included in this thesis where the author contributed as main author or co-author, listed below:

- Passive-Aggressive online learning with nonlinear embeddings, **Jorge, Javier**; Paredes, Roberto;, Pattern Recognition, Elsevier, 2018, Journal Q1
- Empirical Evaluation of Variational Autoencoders for Data Augmentation, **Jorge, Javier**; Vieco, Jesús; Paredes, Roberto;Sánchez, Joan Andreu; Benedí, José Miguel; VISIGRAPP 2018, Conference Core B
- Europarl-ST: A Multilingual Corpus for Speech Translation of Parliamentary Debates; Iranzo-Sánchez, Javier; Silvestre-Cerdà, Joan Albert; **Jorge, Javier**; Roselló, Nahuel; Giménez, Adrià; Sanchis, Albert; Civera, Jorge; Juan, Alfons; ICASSP, 2020, Conference Class A
- Improved Hybrid Streaming ASR with Transformer Language Models; Baquero-Arnal, Pau ; **Jorge, Javier**; Giménez, Adrià ; Silvestre-Cerdà, Joan Albert ; Iranzo-Sánchez, Javier ; Sanchis, Albert ; Civera, Jorge ; Juan, Alfons; InterSpeech 2020, Conference Core A
- Streaming cascade-based speech translation leveraged by a direct segmentation model; Iranzo-Sánchez, Javier; **Jorge, Javier**; Baquero-Arnal, Pau; Silvestre-Cerdà, Joan Albert ; Giménez, Adrià; Civera, Jorge; Sanchis, Albert; Juan, Alfons; Neural Networks, Elsevier, 2020, Journal Q1
- Europarl-ASR: A Large Corpus of Parliamentary Debates for Streaming ASR Benchmarking and Speech Data Filtering/Verbatimization; Garcés Díaz-Munío, Gonçal V; Silvestre-Cerdà, Joan Albert ; **Jorge, Javier**; Giménez, Adrià; Iranzo-Sánchez, Javier; Baquero-Arnal, Pau; Roselló, Nahuel; Pérez-González-de-Martos, Alejandro; Civera, Jorge; Sanchis, Albert; Juan, Alfons; InterSpeech 2021, Conference Core A
- Towards simultaneous machine interpretation; Pérez-González-de-Martos, Alejandro; Iranzo-Sánchez, Javier; Giménez Pastor, Adrià ; **Jorge, Javier**; Silvestre-Cerdà, Joan-Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons; InterSpeech 2021, Conference Core A
- MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge: Extension; Baquero-Arnal, Pau; **Jorge, Javier**; Giménez, Adrià; Iranzo-Sánchez, Javier; Pérez-González-de-Martos, Alejandro; Garcés Díaz-Munío, Gonçal V; Silvestre-Cerdà, Joan Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons; Applied Sciences, MDPI; 2022; Journal Q2.

References

Alvarez, Aitor et al. (2021). "The Vicomtech Speech Transcription Systems for the Albayzın-RTVE 2020 Speech to Text Transcription Challenge". In: *Proc. IberSPEECH 2021*, pp. 104–107 (cit. on p. 128).

Arzelus, Haritz et al. (2018). "The Vicomtech-PRHLT Speech Transcription Systems for the IberSPEECH-RTVE 2018 Speech to Text Transcription Challenge." In: *Proc. of IberSPEECH 2018*, pp. 267–271 (cit. on p. 123).

Baquero-Arnal, Pau et al. (2022). "MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-to-Text Challenge: Extension". In: *Applied Sciences* 12.2 (cit. on pp. 127, 128).

Del-Agua, Miguel Ángel et al. (Dec. 2015). "The MLLP ASR systems for IWSLT 2015". In: *Proc. of IWSLT 2015*. Da Nang, Vietnam, pp. 39–44 (cit. on p. 122).

Del-Agua, Miguel Ángel et al. (2016). "The MLLP system for the 4th CHiME challenge". In: *Proc. of the International Workshop on Speech Processing in Everyday Environments 2016*. CHiME, pp. 57–59 (cit. on p. 122).

Font, Roberto and Teresa Grau (2021). "The Biometric Vox System for the Albayzin-RTVE 2020 Speech-to-Text Challenge". In: *Proc. IberSPEECH 2021*, pp. 99–103 (cit. on p. 128).

Jorge, Javier et al. (Feb. 2020). *X5gon deliverable 3.5: Final support for Cross-lingual OER*. Tech. rep. `www.x5gon.org/science/deliverables`. Universitat Politècnica de València (cit. on pp. 124, 125).

Kocour, Martin et al. (2021). "BCN2BRNO: ASR System Fusion for Albayzin 2020 Speech to Text Challenge". In: *Proc. of IberSPEECH 2021* (cit. on p. 128).

Lleida, Eduardo et al. (2018). *RTVE2018 database description*. `http://catedrartve.unizar.es/reto2018/RTVE2018DB.pdf` (cit. on p. 122).

– (2019). "Albayzin 2018 evaluation: the iberspeech-RTVE challenge on speech technologies for spanish broadcast media". In: *Applied Sciences* 9.24, p. 5412 (cit. on p. 122).

Perero-Codosero, Juan M et al. (2018). "Exploring Open-Source Deep Learning ASR for Speech-to-Text TV program transcription." In: *Proc. of IberSPEECH 2018*, pp. 262–266 (cit. on p. 123).

– (2021). "Sigma-UPM ASR Systems for the IberSpeech-RTVE 2020 Speech-to-Text Transcription Challenge". In: *Proc. IberSPEECH 2021*, pp. 108–112 (cit. on p. 128).

Pérez, Alex et al. (Dec. 2020). *X5gon deliverable 5.3: Final report on piloting*. Tech. rep. `www.x5gon.org/science/deliverables`. Universitat Politècnica de València (cit. on pp. 124, 125).

Chapter 4

Conclusions and future work

To conclude this thesis, this chapter provides some final remarks and open research lines to continue working as future contributions. First, the work of this thesis resulted in a fully-fledged streaming neural-based ASR system that has been defined, developed, evaluated, and deployed for almost three years, starting from an offline multi-pass system. These developments have been included as features of the internal ASR toolkit TLK, which is used intensively in production environments to transcribe recorded and live content. Examples of this content range from lectures, talks, or seminars to challenging TV content such as news, TV series, or morning shows. Moreover, the ASR systems developed during this thesis participated in the last two editions of the international ASR challenge IberSpeech, winning the competition in both editions.

In summary, the main contributions of this thesis are:

- The adaptation of the multi-pass decoder to the one-pass version leveraging neural-based language models, such as LSTM LM, obtaining a fast and accurate system.
- The adaptation of the acoustic modeling required to use the one-pass decoder with the neural LM in the streaming setup, providing high-quality transcription with low-latency outputs.
- The evaluation of the WER, RTF, and latency of the resulting system under real-world conditions, deploying these tools in production environments where it is used intensively.

These contributions lead to a successful hybrid streaming ASR system, demonstrating state-of-the-art results in publicly available datasets such as LibriSpeech, TED-LIUM and challenging real-world tasks such as RTVE2018 and RTVE2020. The scientific and technical success of the proposed tools fueled several research projects, R&D, and technology transfer agreements that are paving the way to continue working on this line of research with other advanced NLP-related tasks, such as machine translation or speech synthesis.

The contributions presented can be further extended in several directions regarding future work. In fact, new neural models providing better results appear constantly, based on new architectures (Gulati et al. 2020; Zeineldeen et al. 2022) or improvements such as self-supervised training (Baevski and Mohamed 2020; Sadhu et al. 2021). The improvements developed for our internal toolkit TLK enable the use of new models quickly, but they may require additional training/evaluation pipelines to get the most out of them. For example, there are different proposals to efficiently exploit the previous/future context in the streaming setup for the latest neural architecture, the Transformer model, for both the LM and AM (Hori et al. 2020; Shenoy et al. 2021; Ma et al. 2021; C. Wu et al. 2020). Exploring the best methods for both models is still an open research question. On the other hand, posing new ASR challenges focused on streaming processing will be very interesting for the ASR community, such as streaming-oriented datasets

with well-defined and widely used latency metrics. From our group we proposed tasks following this idea, but there is still more work to do (Iranzo-Sánchez et al. 2020; Díaz-Munío et al. 2021). This kind of datasets will frame the research and boost the interest from academia to work on real-world streaming conditions, and not only on academic tasks.

References

Baevski, Alexei and Abdelrahman Mohamed (2020). "Effectiveness of Self-Supervised Pre-Training for ASR". In: *Proc. of ICASSP 2020*, pp. 7694–7698 (cit. on p. 132).

Díaz-Munío, Gonçal V. Garcés et al. (Aug. 2021). "Europarl-ASR: A Large Corpus of Parliamentary Debates for Streaming ASR Benchmarking and Speech Data Filtering/Verbatimization". In: *Proc. of InterSpeech 2021* (cit. on p. 133).

Gulati, Anmol et al. (2020). "Conformer: Convolution-augmented Transformer for Speech Recognition". In: (cit. on p. 132).

Hori, Takaaki et al. (2020). "Transformer-Based Long-Context End-to-End Speech Recognition". In: *Proc. of InterSpeech 2020* (cit. on p. 132).

Iranzo-Sánchez, Javier et al. (2020). "Europarl-ST: A Multilingual Corpus For Speech Translation Of Parliamentary Debates". In: *Proc. of ICASSP 2020*, pp. 8229–8233 (cit. on p. 133).

Ma, Xutai et al. (2021). "Streaming Simultaneous Speech Translation with Augmented Memory Transformer". In: *Proc. of ICASSP 2021*, pp. 7523–7527 (cit. on p. 132).

Sadhu, Samik et al. (Aug. 2021). "wav2vec-C: A Self-Supervised Model for Speech Representation Learning". In: *Proc. of InterSpeech 2021*, pp. 711–715 (cit. on p. 132).

Shenoy, Ashish et al. (Aug. 2021). "Adapting Long Context NLM for ASR Rescoring in Conversational Agents". In: *Proc. of InterSpeech 2021*, pp. 3246–3250 (cit. on p. 132).

Wu, Chunyang et al. (Oct. 2020). "Streaming Transformer-Based Acoustic Models Using Self-Attention with Augmented Memory". In: *Proc. of InterSpeech 2020*, pp. 2132–2136 (cit. on p. 132).

Zeineldeen, Mohammad et al. (May 2022). "Conformer-based Hybrid ASR System for Switchboard Dataset". In: *Proc. of ICASSP*. To Appear. Singapore (cit. on p. 132).