UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Streaming Neural Speech Translation

June 2023

Author: Javier Iranzo Sánchez

Supervisors: Dr. Alfons Juan Císcar
Dr. Jorge Civera Saiz

Agradecimientos

Escribo estas líneas en mi lengua materna, porque si bien el inglés es la lengua más adecuada para transmitir el conocimiento científico, resulta más apropiado utilizar el castellano para transmitir de manera clara lo que siento. Resulta totalmente imposible dar las gracias en únicamente dos páginas a todas las personas que durante estos 4 años han sido una parte importante de este periplo, así que inevitablemente habrá alguna exclusión inmerecida, debido a restricciones organizativas y la dificultad de resumir tanto tiempo en unas pocas líneas.

Quiero agradecer la labor de supervisión de mis directores Alfons y Jorge, que llevan guiándome desde mucho antes de comenzar esta tesis. Su atención a lo largo de estos años ha sido un factor determinante en el éxito de esta, especialmente las largas sesiones de revisión y reescritura conjunta a las que hemos sometido a los trabajos publicados, sin las cuales su claridad y rigor matemático serían bastante inferiores a las que se han obtenido finalmente. De igual manera, quiero agradecer a los compañeros del equipo MLLP, Adrià Giménez, Adrià Martínez, Albert, Álex, Gonçal, Pau, Javi, Joan Albert, Miguel, Nahuel y Santi, no únicamente por sus contribuciones profesionales al buen funcionamiento del grupo, sino también por todas las aventuras, discusiones político-históricas de todo tipo y demás ocurrencias vividas, sin olvidar las partidas de Towerfall y sesiones de juegos de mesa, que han hecho de esta una etapa inolvidable. Quiero agradecer especialmente a Adrià Giménez, por sus conocimientos técnicos y aportaciones intelectuales a lo largo de muchas discusiones y sesiones de pizarra de ideas locas, muchas de las cuales no han

visto la luz del día, pero que han sido la semilla que ha hecho crecer muchos de los trabajos que he publicado a lo largo de estos años.

Durante todo este tiempo he contado con el apoyo y el cariño de mi familia y amigos. Mis padres Pascual y María José, y mis hermanos Ana y Jorge han sido una fuente constante de afecto y ánimos. Esta tesis está dedicada a mis abuelos, que ya no están este nosotros, que desde pequeño fueron cultivando, animando y haciendo crecer mi curiosidad y mi interés por el estudio. Durante todo este viaje he tenido la increíble suerte de haber estado acompañado en todo momento por mi alma gemela, Aysha, que no ha dejado de creer en mí en ningún momento y me ha animado constantemente a llevar hasta el final este proyecto. No sólo me has hecho sentir la persona con más suerte del mundo, sino que también, a nivel personal y organizativo, has conseguido que funcionaran las cosas cuando las fechas de entregas me exigían sacrificios personales. El mérito de haber conseguido esta tesis es también en parte tuyo, porque sin ti no habría sido lo mismo.

Por último, quiero también enviar un recuerdo a mis mejores amigos, Rafa, Vicente y Clara, así como a Ramon, Jaime y el resto de compañeros del máster y la facultad, todas las personas a las que desgraciadamente he tenido que dejar de ver tanto como me hubiera gustado. No tengo la menor duda de que, una vez libre de la Espada de Damocles, nuestros caminos volverán a encontrarse.

Abstract

Thanks to significant advances in Deep Learning, Speech Translation (ST) has become a mature field that enables the use of ST technology in production-ready solutions. Due to the ever-increasing hours of audio-visual content produced each year, as well as higher awareness of the importance of media accessibility, ST is poised to become a key element for the production of entertainment and educational media.

Although significant advances have been made in ST, most research has focused on the offline scenario, where the entire input audio is available. In contrast, online ST remains an under-researched topic. A special case of online ST, streaming ST, translates an unbounded input stream in a real-time fashion under strict latency constraints. This is a much more realistic problem that needs to be solved in order to apply ST to a variety of real-life tasks.

The focus of this thesis is on researching and developing key techniques necessary for a successful streaming ST solution. First, in order to enable ST system development and evaluation, a new multilingual ST dataset is collected, which significantly expands the amount of hours available for ST. Then, a streaming-ready segmenter component is developed to segment the intermediate transcriptions of our proposed cascade solution, which consists in an Automatic Speech Recognition (ASR) system that transcribes the audio, followed by a Machine Translation (MT) system that translates the intermediate transcriptions into the desired language. Research has shown that segmentation quality plays a significant role in downstream MT performance, so the development of an effective streaming segmenter is a critical step in the streaming ST pro-

cess. This segmenter is then integrated and the components of the cascade are jointly optimized to achieve an appropriate quality-latency trade-off.

Streaming ST has much more strict latency constraints than standard online ST, as the desired latency level must be maintained during the whole translation process. Therefore, it is crucial to be able to accurately measure this latency, but the standard online ST metrics are not well suited for this task. As a consequence, new evaluation methods are proposed for streaming ST evaluation, which ensure realistic, yet interpretable results.

Lastly, a novel method is presented for improving translation quality through the use of contextual information. Whereas standard online ST systems translate audios in isolation, there is a wealth of contextual information available for improving streaming ST systems. Our approach introduces the concept of streaming history by storing the most recent information of the translation process, which is then used by the model in order to improve translation quality.

Resumen

Gracias a avances significativos en aprendizaje profundo, la traducción del habla (ST) se ha convertido en un campo consolidado, lo que permite la utilización de la tecnología ST en soluciones para entornos de producción. Como consecuencia del aumento constante del número de horas de contenido audiovisual generado cada año, así como una mayor sensibilización sobre la importancia de la accesibilidad, la ST está preparada para convertirse en un elemento clave para la producción de contenidos audiovisuales, tanto de ocio como educativos.

A pesar de que se ha progresado significativamente en ST, la mayor parte de la investigación se ha centrado en el escenario en diferido (*offline*), en el cual todo el audio de entrada está disponible. En cambio, la ST en directo (*online*) es una temática en la que falta mucho por investigar. En concreto, existe un caso de traducción en directo, la traducción continua (*streaming*), que traduce un flujo continuo de palabras en tiempo real y bajo unas estrictas condiciones de latencia. Este es un problema mucho más realista, que es necesario resolver para que sea posible aplicar la ST a una variedad de tareas de la vida real.

Esta tesis está centrada en investigar y desarrollar las técnicas claves que son necesarias para una solución de ST continua. En primer lugar, de cara a permitir el desarrollo y la evaluación de sistemas de ST, se ha recopilado un nuevo conjunto de datos para ST multilingüe, que expande significativamente el número de horas disponibles para ST. A continuación se ha desarrollado un segmentador preparado para la condición continua, que se utiliza para segmentar las transcripciones intermedias de nuestra solución por etapas, que consiste

en un sistema de reconocimiento automático del habla (ASR), seguido de un sistema de traducción automática (MT) encargado de traducir las transcripciones intermedias al idioma de destino elegido. Diversas investigaciones han concluido que la calidad de la segmentación es un factor muy influyente es la calidad del sistema MT, por lo que el desarrollo de un segmentador efectivo es un paso fundamental en el proceso de ST continua. Este segmentador se ha integrado en la solución por etapas, y estas se optimizan de manera conjunta para alcanzar el equilibrio óptimo entre calidad y latencia.

La ST continua tiene unas restricciones de latencia mucho más estrictas que la ST en directo, ya que el nivel deseado de latencia tiene que mantenerse durante todo el proceso de traducción. Por tanto, es crucial ser capaz de medir de manera precisa esta latencia, pero las métricas estándar de ST en directo no se adaptan bien a esta tarea. Como consecuencia de esto, se proponen nuevos métodos para la evaluación de ST continua, que garantizan unos resultados precisos a la vez que interpretables.

Por último, se presenta un nuevo método para mejorar la calidad de la traducción continua mediante el uso de información contextual. Mientras que los sistemas tradicionales de ST en directo traducen audios de manera aislada, existe abundante información contextual que está disponible para mejorar los sistemas de ST continua. Nuestra propuesta introduce el concepto de historia continua, que consiste en el almacenamiento de la información más reciente del proceso de traducción, que se utiliza más adelante por el modelo para mejorar la calidad de la traducción.

Resum

Gràcies a avanços significatius en aprenentatge profund, la traducció de la parla (ST) s'ha convertit en un camp consolidat, la qual cosa permet la utilització de la tecnologia ST en solucions per a entorns de producció. A conseqüència de l'augment constant del nombre d'hores de contingut audiovisual generat cada any, així com una major sensibilització sobre la importància de l'accessibilitat, la ST està preparada per a convertir-se en un element clau per a la producció de continguts audiovisuals, tant d'oci com educatius.

A pesar que s'ha progressat significativament en ST, la major part de la recerca s'ha centrat en l'escenari en diferit, en el qual tot l'àudio d'entrada està disponible. En canvi, la ST en directe és una temàtica en la qual falta molt per investigar. En concret, existeix un cas de traducció en directe, la traducció contínua, que tradueix un flux continu de paraules en temps real i sota unes estrictes condicions de latència. Aquest és un problema molt més realista, que és necessari resoldre perquè sigui possible aplicar la ST a una varietat de tasques de la vida real.

Aquesta tesi està centrada en investigar i desenvolupar les tècniques claus que són necessàries per a una solució de ST contínua. En primer lloc, de cara a permetre el desenvolupament i l'avaluació de sistemes de ST, s'ha recopilat un nou conjunt de dades per a ST multilingüe, que expandeix significativament la quantitat de dades disponibles per a ST. A continuació s'ha desenvolupat un segmentador preparat per a la condició contínua, que s'utilitza per a segmentar les transcripcions intermèdies de la nostra solució per etapes, que consisteix en un sistema de reconeixement automàtic de la parla (ASR), seguit d'un

sistema de traducció automàtica (MT) encarregat de traduir les transcripcions intermèdies a l'idioma de destí triat. Diveros treballs de recerca han conclòs que la qualitat de la segmentació és un factor molt important en la qualitat del sistema MT, per la qual cosa el desenvolupament d'un segmentador efectiu és un pas fonamental en el procés de ST contínua. Aquest segmentador s'ha integrat en la solució per etapes, i aquestes s'optimitzen de manera conjunta per a aconseguir l'equilibri òptim entre qualitat i latència.

La ST contínua té unes restriccions de latència molt més estrictes que la ST en directe, ja que el nivell desitjat de latència ha de mantindre's durant tot el procés de traducció. Per tant, és crucial ser capaç de mesurar de manera precisa aquesta latència, però les mètriques estàndard de ST en directe no s'adapten bé a aquesta tasca. A conseqüència d'això, es proposen nous mètodes per a l'avaluació de ST contínua, que garanteixen uns resultats precisos alhora que interpretables.

Finalment, es presenta un nou mètode per a millorar la qualitat de la traducció contínua mitjançant l'ús d'informació contextual. Mentre que els sistemes tradicionals de ST en directe tradueixen àudios de manera aïllada, existeix abundant informació contextual que està disponible per a millorar els sistemes de ST contínua. La nostra proposta introdueix el concepte d'història contínua, que consisteix en l'emmagatzematge de la informació més recent del procés de traducció, que s'utilitza més endavant pel model per a millorar la qualitat de la traducció.

Contents

List of Figures

List of Tables

List of acronyms

ACT	Adaptive Computation Time
AL	Average Lagging
AM	Acoustic Model
AP	Average Precision
ASR	Automatic Speech Recognition
BLEU	Bilingual Evaluation Understudy
BLSTM	Bidirectional Long Short-Term Memory
BPE	Byte-Pair Encoding
BPTT	Back-Propagation Through Time
CART	Classification and regression tree
CC	Creative Commons
CER	Character Error Rate
CPU	Central Processing Unit
DAL	Differentiable Average Lagging
DNN	Deep Neural network
DS	Direct Segmentation

FFN Feed Forward Network

GD Gradient Descent

GMM Gaussian Mixture Model

GPU Graphical Processing Unit

GRU Gated Recurrent Unit

HMM Hidden Markov Model

IL Infinite Lookback

IWSLT International Conference on Spoken Language Translation

LM Language Model

LMHP Language Model Histogram Pruning

LMHR Language Model Histogram Recombination

LSTM Long Short-Term memory

MEP Member of the European Parliament

MFCC Mel Frequency Cepstral Coefficients

ML Machine Learning

MLP MultiLayer Perceptron

MMA Monotonic Multi-Head Attention

MMAH Hard Monotonic Multi-Head Attention

MT Machine Translation

NCE Noise-Contrastive Estimation

NLP Natural Language Processing

NMT Neural Machine Translation

NN Neural Network

OOV Out Of Vocabulary

POS Part-Of-Speech

PPL	Perplexity
RELU	Rectified Linear Unit
RNN	Recurrent Neural Network
RNNLM	Recurrent Neural Network Language Model
RTF	Real-Time Factor
SD	Speaker Diarization
SGD	Stochastic Gradient Descent
SHAS	Supervised Hybrid Audio Segmentation
SLT	Spoken Language Translation
SMT	Statistical Machine Translation
ST	Speech Translation
STR	Streaming
TER	Translation Edit Rate
TL	Translation Lag
TLK	transLectures UPV toolkit
TLM	Transformer Language Model
TTS	Text To Speech
VAD	Voice Activity Detection
VR	Variance Regularization
WER	Word Error Rate
WFST	Weighted Finite-State Transducer
WMT	Conference on Machine Translation

Chapter 1

Introduction

1 Motivation

In an increasingly globalized world, language barriers are still a limiting factor for communication between humans. For example, in 2012, only 54% of the European population was able to have a conversation in a language different than their mother tongue. In terms of the world's most commonly spoken language, only 38% of European citizens were able to hold a conversation in English (European Commission 2012). Owing to this lack of shared language, there is a significant communication gap waiting to be solved.

Machine Translation (MT), a subfield of Artificial Intelligence, seeks to build automatic systems that can translate between two languages. The data-driven approach to MT started to gather widespread adoption with the introduction of the word-based IBM models (Brown, Cocke, et al. 1990; Brown, Pietra, et al. 1993). In the early 2000s, a significant leap forward in quality was obtained moving from word-based to phrase-based translation models (Koehn, Och, and Marcu 2003). This culminated in the creation of Google Translate in 2006, at a time where the field had matured enough for the first widespread commercial offering to appear. In parallel with other fields that use Machine Learning (ML), the arrival of deep learning radically changed the MT field which saw radical improvements in quality in a few years (Bahdanau, Cho, and Bengio 2015; Vaswani et al. 2017). Likewise, deep learning has also radically changed the field of Automatic Speech Recognition (ASR) (Dahl et al. 2011; Chan et al. 2016; Irie et al. 2019; Park et al. 2019; Jorge, Giménez, et al. 2020). Thanks to these advances in both ASR and MT, current Speech Translation (ST) systems, which aim to automatically generate the text translation of an audio waveform, have just now begun to obtain acceptable results for widespread adoption. For this thesis, the ST task is approached using a cascaded system: A streaming ASR system transcribes the audio, which is then split into chunks using a streaming segmenter component, and the chunks are given as input to a streaming MT system which produces the final translation. Although end-to-end ST systems exist, at the time of writing this thesis, cascaded ST is still the dominant choice (Anastasopoulos et al. 2022). As a result of all the previous issues, the development of a successful streaming ST system poses some significant challenges.

Nevertheless, the majority of the work in ST is concerned only with the *offline* task, that is, the task in which the entire input audio is available, and no real-time constraints exist. In contrast, in the *online* task, the input audio is incrementally received as time passes, and the system must produce a translation of a partial input within a certain latency threshold, in a real-time

fashion. Online ST is inherently a harder problem, because the partial input compromises the quality of the translation, and due to the real-time constraint, the computational efficiency of the system cannot be ignored.

At the same time, the demand for online ST systems is increasing, with the number of hours of audio-visual content produced increasing year after year. Additionally, there is a trend by governments to mandate that a higher percentage of these materials must be accessible, including in other languages, which further fuels the need for ST systems.

When combined, all of these factors ensure that streaming ST is a challenging and exciting research topic, with the potential of a significant impact in breaking down barriers in human communication.

2 Scientific goals

The main topic of this thesis is streaming ST, which we consider to be a special task of online ST. We have already discussed how online ST has two main characteristics: translation of partial input, and real-time constraints. This thesis focuses on a harder problem, streaming ST, which is a particular case of online ST. Specifically, we define streaming ST as an online ST task where the input is an unbounded audio stream. Many online ST works are evaluated on academic datasets consisting in short isolated audio clips. Therefore, the conclusions that can be extracted from this type of research is limited, and can only be directly applied to tasks involving the translation of short isolated audio clips, such as voice commands. In contrast, streaming ST is a more realistic setting for the study of ST, which closely mimics the conditions of many real-time tasks, such as the translation of a live conversation, a lecture or a news broadcast.

This setting brings with it some additional complications. As the input is an unbounded audio stream, it must be processed, segmented and translated on-the-fly, so that real-time translation can occur. On the other hand, the temporal information contained in the input stream can be exploited in order to improve the quality of the translation. In this way, streaming ST can also be understood as an specific case of contextual MT. Contextual MT, also known as document-level MT, tries to avoid the shortcomings of translating sentences in isolation, by augmenting the model with contextual information about the surrounding source sentences or the already generated translations. Additionally, greater care must be put into the computational efficiency and

behaviour of the system, as delays in some part of the stream translation can be propagated later, leading to poor results.

In order to develop a streaming ST system, one must first have a suitable dataset for system training and evaluation. This dataset, which consists in raw audio paired with its transcription and translation, should be big enough to be used by modern state-of-the-art systems. In order to be useful for a streaming task, such a dataset must also preserve the original ordering and temporal information. Additionally, the dataset should ideally cover multiple languages and translation directions, in order to test that developments can also be applied to other translation tasks, unlike the English-centric approach of mainstream research. Thus, our first challenge lies in how to construct this dataset.

Secondly, the output of the streaming ASR system, which is an unbounded stream of words, can be segmented into sentence-like segments or chunks, because MT systems are trained using complete sentences, and the quality of the translation has been shown to be correlated with the quality of the segmentation (Rangarajan Sridhar et al. 2013; Gaido et al. 2021; Tsiamas et al. 2022; Anastasopoulos et al. 2022). Given that we are working in the streaming task, this segmentation must be carried out using only the partial input available at each timestep. Likewise, standard ASR systems have been trained using normalized text, in which all words all lowercased, punctuation signs removed, and numbers are replaced by their phonetic transliteration. In contrast, standard MT systems expect inputs with proper casing and punctuation. Thus, a second significant challenge lies in how to best process and segment ASR output in order to maximize the performance of the downstream MT system.

Up to this point, no specific evaluation framework has been introduced for streaming MT, so we are limited to evaluating streaming ST in the same way that we do for online systems. However, as previously mentioned, in classical online ST, the evaluation is carried out by translating small ($\simeq$10s) isolated chunks of audio, independent from each other. This is not a realistic task for streaming ST, in which we want to take into account how the behaviour and delays of the system interact during the whole translation process. Current online ST metrics are not accurate when adapted to the streaming case (Schneider and Alexander Waibel 2020), so our third challenge consists in developing evaluation procedures and metrics that can reliably evaluate the performance of streaming ST systems.

Last, but not least, in a streaming ST task there exists temporal information and an ordering between the words and chunks received by the MT model.

This contextual information is not present on the baseline online task, and could be used by the model in order to improve translation quality. Works in related areas of contextual MT have shown significant improvements when using contextual information (Tiedemann and Scherrer 2017; Müller et al. 2018; Voita, Sennrich, and Titov 2019b; Voita, Sennrich, and Titov 2019a), but these techniques have not been adapted to the streaming task in which only a partial context is available. Thus, there exists a significant challenge in how contextual MT techniques can be applied to the streaming ST case.

In order to solve these challenges, the following scientific questions will be addressed in this thesis:

- **How can we obtain a ST dataset that can be used for realistic, multilingual development and evaluation of streaming ST systems?**
- **How can the output of the ASR system be processed and segmented in order to maximize the performance of a cascaded MT system?**
- **How can the performance of streaming ST systems be evaluated with a procedure that takes into account the sequential nature of the problem and is at the same time interpretable?**
- **How can a streaming ST system best take advantage of contextual information in order to improve translation quality?**

3 Preliminaries

This section introduces the preliminaries that are required for understanding the rest of the work carried out during this thesis. Parts of this section are derived from the author's B.S. (Iranzo-Sánchez 2018) and M.S. (Iranzo-Sánchez 2019) theses.

3.1 Machine Learning

The field of Machine Learning (ML) studies the feasibility of creating automatic systems that can learn. (Mitchell 1997) defines it as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." ML techniques have been applied to a very

diverse range of tasks T, in order to build systems that can carry out these tasks for us.

Many of the most popular ML tasks fall under the umbrella of supervised learning: The ML model must learn a function $f(\boldsymbol{x})$ that maps some input $\boldsymbol{x} \in \mathcal{X}$ into some output $\boldsymbol{y} \in \mathcal{Y}$. Under this task, the model will have access to a set of experiences E which consists in many pairs of $(\boldsymbol{x}, \boldsymbol{y})$, which it will use in order to learn how to perform the task.

An ML model is composed of a series of mathematical operations, and the parameters $\boldsymbol{\theta}$ that characterize them. Thus, the model defines the function $f(\boldsymbol{x}; \boldsymbol{\theta})$. In order to maximize the model's performance, as measured by P, during the training stage the model will use the samples in order to optimize $\boldsymbol{\theta}$. More formally, let

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}). \tag{1.1}$$

For each sample pair $(\boldsymbol{x}, \boldsymbol{y})$, the performance of the model can be measured using a loss function $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$. By convention, the goal is to minimize the loss function. For example, for regression problems it is typical to use the Mean-Squared-Error (MSE) loss:

$$L_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = (\hat{\boldsymbol{y}} - \boldsymbol{y})^2. \tag{1.2}$$

Based on this, the goal is to minimize a cost function $J(\boldsymbol{\theta})$, which is defined as the expected value of the loss function over the underlying data distribution:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim p_{\text{data}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y}). \tag{1.3}$$

As we do not have access to the true data distribution, we must instead optimize the cost function over the training data distribution $\hat{p}_{\text{data}}$.

It is also common to introduce other terms (sometimes called auxiliary losses) into the cost function if one wishes to take into account some other property of the parameters. If one wished to control the magnitude of the weights, an L2 auxiliary loss could for example be introduced. This is also known as regularization:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim \hat{p}_{\text{data}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda L_{\text{L2}}(\boldsymbol{\theta}) \tag{1.4}$$

$$L_{\mathrm{L2}}(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2^2. \tag{1.5}$$

In order to optimize $\boldsymbol{\theta}$, the gradient descent (GD) algorithm is used to iteratively subtract the gradient of the parameters with respect to the cost function:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{1.6}$$

The *learning rate* or *step size* α controls how big is the update we make on each step. Computing the exact gradient over the entire training data is computationally expensive, so the gradient is usually computed over a small subset of the training data which is randomly sampled in each step, called a minibatch. This modification is known as stochastic gradient descent (SGD). Conceptually, SGD can be understood as a modification of GD consisting in updating the parameters with a noisy estimation of the gradient:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \epsilon). \tag{1.7}$$

In practice, the exact gradient is not necessary and SGD reaches a good result in the parameter space, with the additional noise acting as a regularizer which can help during the learning process. Furthermore, SGD can converge faster than GD due to the fact that the parameters are updated more often (Bottou and Bousquet 2007). In order to optimize the model, one must solve the problem of computing the partial derivatives $\frac{\partial J}{\partial \theta_i}$, something which is not straightforward at first. Nowadays, the state-of-the-art approach to many ML tasks is to use a neural network model. These models are composed of multiple layers, each one applying an operation over the output of the previous layer. Computing the gradient for these models involves many layers and parameters, which introduces computational concerns. This problem can be solved efficiently by using the backpropagation algorithm, which is based on the chain rule.

Using the chain rule, if $y = f_1(x, \theta)$ and $J = f_2(y)$, then the derivative can be computed as

$$\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial \theta}. \tag{1.8}$$

It is common to use a structure known as a computational graph in order to represent an ML model. A computational graph is a directed graph composed

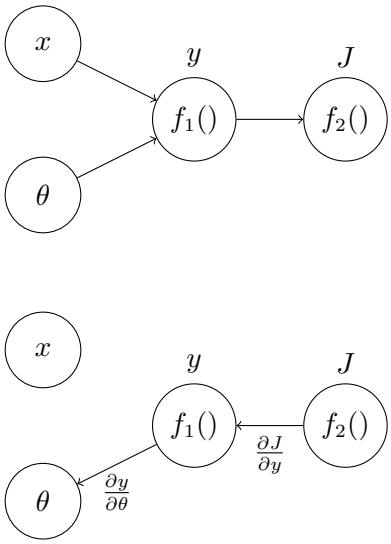

Figure 1.1: Example of gradient computation using the chain rule. Both the forward pass (top) and the backward pass (bottom) are shown.

of variables, each represented by a node. Each node computes an operation over the input variables, represented by edges, and produces an output. This representation explicitly records the parameters and functions that define an ML model, as well as the dependencies between those operations. Figure 1.1 shows a computational graph illustrating the computation of the partial derivative.

The chain rule can be used to generalize the process of computing $\frac{\partial J}{\partial a}$ for any arbitrary node a in the computational graph, by following this iterative process:

1. Perform all computations defined by the graph, and store the results of each node. This is known as the forward pass.
2. Starting with the output node, compute for each node a and each of its incoming nodes b, $\frac{\partial a}{\partial b}$. This value is associated to the edge linking a and b. This is known as the backward pass.
3. For each node for which we require $\frac{\partial J}{\partial a}$, this can be computed as the product of the partial derivatives along the path from a to J.

If there exist multiple paths from a to J, $\frac{\partial J}{\partial a}$ is the sum of the values of each path, which can be computed by taking into account all outgoing nodes. A

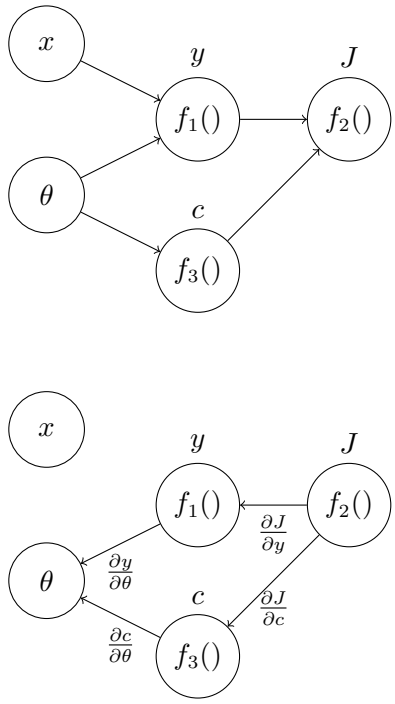

Figure 1.2: Gradient computation using the multivariate chain rule (single input variable). Both the forward pass (top) and the backward pass (bottom) are shown.

graphical example of this is shown in Figure 1.2. For this specific case, the gradient for θ is computed as

$$\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial \theta} + \frac{\partial J}{\partial c}\frac{\partial c}{\partial \theta}. \tag{1.9}$$

3.2 Machine Translation

As stated in Equation 1.1, a ML system learns a function from $\mathcal{X}$ to $\mathcal{Y}$. In MT, $\boldsymbol{x} \in \mathcal{X}$ represents the source sentence in the original language, and $\boldsymbol{y} \in \mathcal{Y}$ represents the translation of the input sentence. As previously stated, state-of-the-art MT systems are based on neural networks. Although their specifics vary from case to case, most current MT systems fall under the encoder-decoder paradigm. Under this paradigm, an MT system is formed by 3 distinct compo-

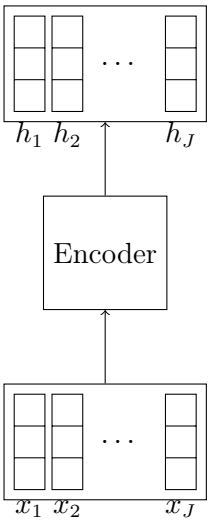

Figure 1.3: Basic schematic of the encoder of an encoder-decoder system.

nents: an encoder component, a decoder component and an encoder-decoder attention component.

The encoder component takes the vector representation of the input words, processes them through a series of layer, and produces, for each input word x_i, an encoded representation h_i, which contains a richer representation of the word and its context. Figure 1.3 shows the basic schema of an encoder component.

The decoder component uses two sources of information to produce the translation, the encodings produced by the encoder, and optionally information about the current state of the translation process. Standard encoder-decoder models are auto-regressive, that is, the translation is produced one word at a time, taking into account the previously produced words. This is why the decoder must also have access to the translation state, in order to account for which words have already been translated.

The decoder receives the representations produced by the encoder by means of an encoder-decoder attention mechanism. An attention mechanism is a function which maps a set of multiple vectors into a single one. This is what enables the model to work with variable length input sentences. An attention function is described by (Vaswani et al. 2017) as a function whose arguments are a query, $\mathbf{q}$, and a set of key-value pairs, grouped into matrices $\mathbf{K}$ and $\mathbf{V}$,

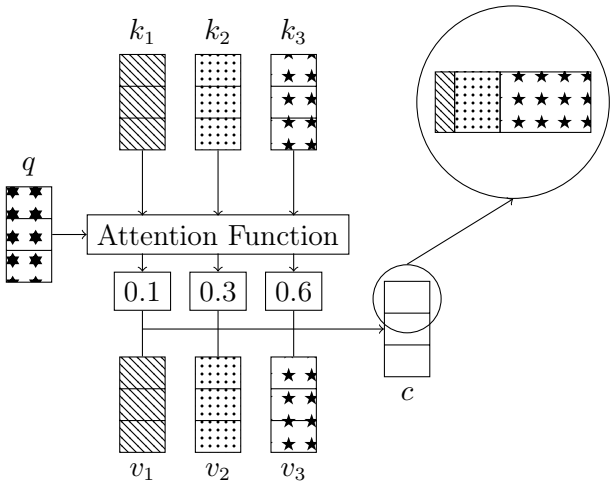

Figure 1.4: Example of how the attention mechanism works. In this example, the six-pointed stars of the query are conceptually closer to the five-pointed stars (k_3), followed by the dots(k_2). For simplicity, in this case we assume that the key and associated value share the same representation.

and whose output is a weighted sum of the values. The weights for each value are computed by a compatibility function between its key and the query.

Attention can thus be interpreted as, given a query, choosing which keys are more relevant as a result of some operation with the query, and producing an output vector, which is the weighted sum of the values, assigning more weight to each value depending on the relevance of each one. The hope is that, if the attention mechanism is working as intended, more weight will be assigned to the source words that are the most relevant for producing the current word to be translated. Figure 1.4 shows a graphical example of how attention works.

A basic encoder-decoder attention mechanism produces one context vector c_i at each time step:

$$c_i = \sum_j \alpha(j|i)\mathbf{V}_j, \tag{1.10}$$

where

$$\alpha(j|i) = \text{softmax}(\text{attention}(\mathbf{q}, \mathbf{K}))_j. \tag{1.11}$$

In MT, we compute the context vector as the weighted sum of the different encoder representations at each time step. The generic attention mechanism described in Equations (1.10) and (1.11) can be specified to produce a context vector c_i from a series of encoder representations $h_1, \ldots, h_J$ as key-value pairs and using the decoder state, $\mathbf{s_{i-1}}$ as the query. Therefore, $\mathbf{q} = \mathbf{s_{i-1}}$, $\mathbf{K_j} = \mathbf{h_j}$, and $\mathbf{V_j} = \mathbf{h_j}$, leaving us with:

$$\mathbf{c}_i = \sum_j \alpha(j|i)\mathbf{h}_j, \tag{1.12}$$

where

$$\alpha(j|i) = \text{softmax}(\text{attention}(\mathbf{s}_{i-1}, \mathbf{K}))_j. \tag{1.13}$$

Dot-Product Attention is typically used as the compatibility function:

$$\text{attention}(\mathbf{s}_{i-1}, \mathbf{K}) = \mathbf{s}_{i-1}^T\mathbf{K}. \tag{1.14}$$

These $\alpha(j|i)$ weights can be interpreted as acting as an alignment. At each time step i, $\alpha(j|i)$ can be understood as the probability that the target word at position i is aligned with the input word at position j. Figure 1.5 shows the basic schema of the decoder.

3.3 Transformer architecture

The Transformer (Vaswani et al. 2017) architecture replaces recurrent layers by a new type of layer, a Self-Attention layer, as well as a series of architectural changes in both the encoder and decoder components. This model achieves significant improvements in both speed and quality of translations, and is currently considered the state of the art.

This section provides a general overview of the model and the proposed improvements. Due to the model's complexity and the wide variety of introduced changes, readers should refer to the original article to learn about details of the implementation.

Both the encoder and the decoder are composed of a series of layer blocks stacked on top of each other. Each of these blocks is made up of a series of sub-layers. A sub-layer implements a function, such as a neural network

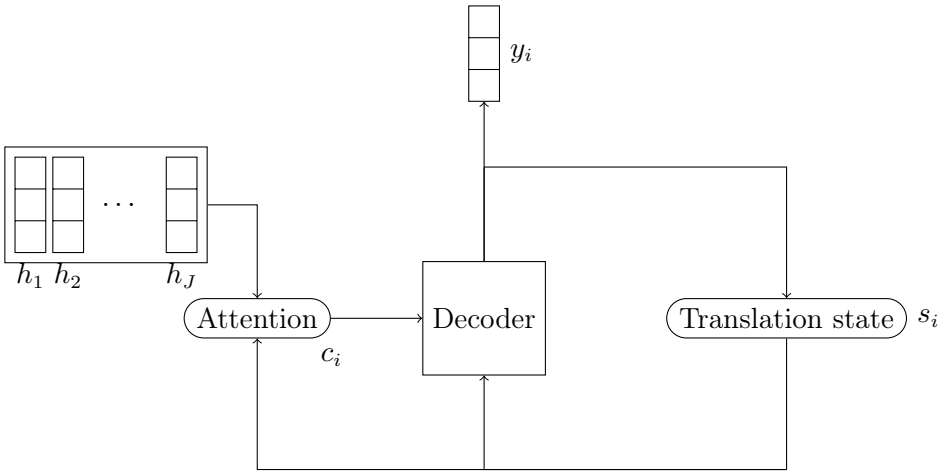

Figure 1.5: Basic schematic of the autoregressive decoder of an encoder-decoder system.

hidden layer, jointly with Residual Connections (He et al. 2016) and Layer Normalization (Ba, Kiros, and Hinton 2016). In what follows, each of these techniques is described, starting with the main contribution of the Transformer model, the introduction of a new way of computing attention.

The attention layers of the Transformer architecture perform multiple Scaled Dot-Product Attention by using Multi-Head Attention. We have previously introduced Dot-Product Attention in Equation (1.14). Scaled attention introduces a scaling term that depends on the dimensions of the key, d_k. The Transformer computes scaled attention as

$$\text{attention}(\mathbf{q}, \mathbf{K}) = \frac{\mathbf{q}^T \mathbf{K}}{\sqrt{d_k}}. \tag{1.15}$$

If we wish to compute attention with multiple queries, the queries can be packed in a matrix Q, and the computation may be expressed solely in terms of matrix multiplication. This means that the entire attention process is computed as

$$\text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{1.16}$$

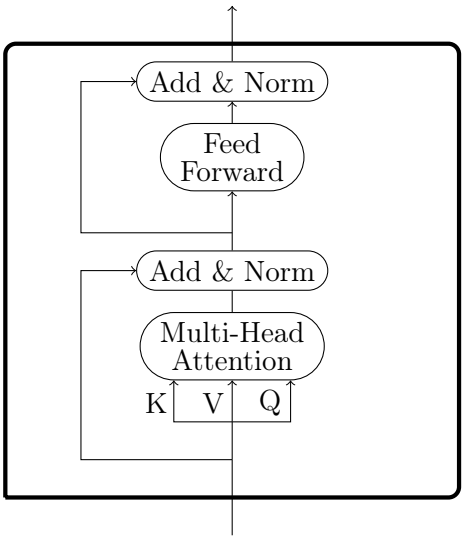

Figure 1.6: Transformer encoder block.

Up to now, the attention mechanism has provided us with an answer for each query. Multi-Head attention extends the previous mechanism in order to produce an answer that is the combination of multiple key-query comparisons. Multi-head attention consists in performing several attention operations in parallel and combining the results to obtain the final context vector. Each individual attention operation or head is carried out by applying a linear projection to the query, keys and values, computing attention between them, and then projecting back into a common space:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^o \tag{1.17}$$

$$\text{head}_i = \text{attention}(\mathbf{QW_i^Q}, \mathbf{KW_i^K}, \mathbf{VW_i^V}). \tag{1.18}$$

The projections are applied by means of matrices $\mathbf{W}^o$,$\mathbf{W_i^Q}$,$\mathbf{W_i^K}$ and $\mathbf{W_i^V}$. These projection matrices are parameters learned during training.

There are two types of sub-layers in the Transformer architecture, Feed Forward layers and the previously described Multi-Head Attention layers. Feed Forward layers are standard neural network layers consisting on weight matrix multiplication followed by an activation function. Figure 1.6 shows a Transformer encoder block.

The standard Transformer block consists in a Multi-Head attention sub-layer followed by a Feed Forward sub-layer. The entire input sentence is fed to the encoder, and the input sentence representation is computed all at the same time. The self attention layers in the encoder apply Multi-Head attention to the output of the previous layer, using that output as both query and key-value pairs. Therefore, at each layer, the encoder produces a representation for each word, that can incorporate information about any other word in the sentence thanks to the self-attention mechanism. The entire representation can therefore be produced in a single pass.

Figure 1.7 shows a Transformer decoder block. In the same way as all other neural MT models, the decoder produces one word at a time, conditioned by the previously emitted words. The decoder is fed the previously emitted words and its Multi-Head attentions sub-layers perform their computations over the output of the previous decoder layer. Compared with the encoder blocks, the decoder blocks include an additional Multi-Head cross attention sub-layer that attends to the output of the encoder stack, allowing the decoder to access the input sentence representations. In these cross attention sub-layers, the output of the previous decoder layer acts as query, while the encoder output acts as key-value pair.

A graphical overview of the Transformer architecture is shown in Figure 1.8. Note that, in the case of the Transformer architecture, self-attention layers, by themselves, offer no way of knowing word order in a sentence, since they treat all inputs the same way. In order to provide the missing word-order information, the Transformer architectures uses positional embeddings that encode this information using sine and cosine functions.

It is usual to choose one of the following configurations, Transformer Base or Big, when building MT systems:

- **Transformer Base**: 6 encoder/decoder blocks, embedding dimension 512, hidden layer size of 2048 and 8 attention heads.
- **Transformer Big**: 6 encoder/decoder blocks, embedding dimension 1024, hidden layer size of 4096 and 16 attention heads.

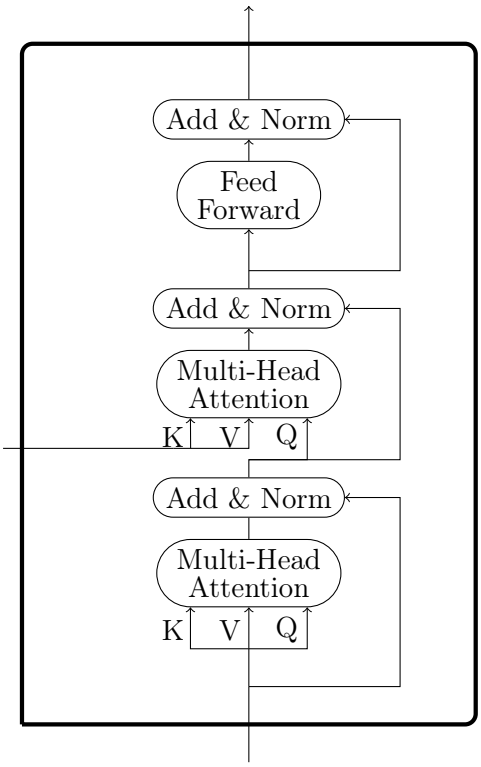

Figure 1.7: Transformer decoder block.

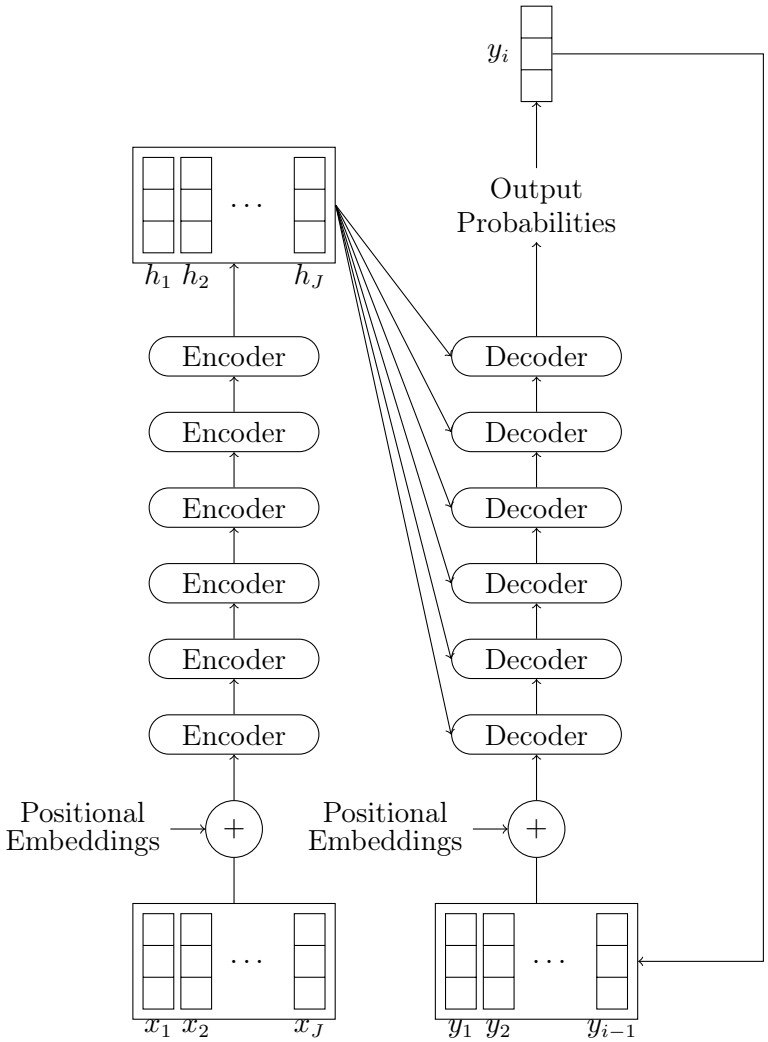

Figure 1.8: Figure of the Transformer architecture.

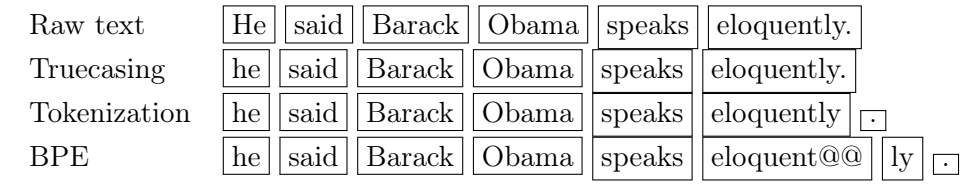

Figure 1.9: Pre-processing pipeline example. Each step is applied over the output of the previous one. Notice how rare word "eloquently" has been split into subword units, avoiding a potential Out-of-vocabulary problem.

3.4 Data processing and benchmarking

Neural networks work with real numbers, so before any processing can occur, it is necessary to transform the textual representation of the input words into a vector representation. Vectorization is carried out by assigning each unique word to a unique vector. The set of words to be considered is called the vocabulary, and words outside it are by convention all mapped to an special "out-of-vocabulary" or "unknown" vector. These vectors, also known as word embeddings, are considered parameters and are typically jointly optimized with the rest of the model parameters. As mentioned in Section 3.3, Transformer self-attention layers need positional embeddings in order to access word-order information. The positional embedding is added together with the word embedding to produce the input to the network.

There is a strong interest in avoiding redundant entries in the vocabulary, so as to not waste parameters and computation on redundant features. This is achieved by a step known as pre-processing, with the goal of collapsing similar words into a single one. One typical step in pre-processing is truecasing, to avoid having separate uppercase and lowercase representations of the same word, as well as tokenization, which breaks up words into individual components (for example, separating words from punctuation signs). The resulting vocabulary should be as small as possible without compromising modelling capabilities. However, the use of a fixed-sized vocabulary leaves the model unable to process out-of-vocabulary words. This is why words are typically broken down into sub-units, called sub-words. This technique enables open vocabulary translation if we use a sub-word vocabulary, because unknown words can be broken into known sub-units. Through this thesis, Byte Pair Encoding (BPE) subwords are used (Sennrich, Haddow, and Birch 2016). Figure 1.9 shows an example of the whole pre-processing procedure.

Data availability is one of the most critical factors for building successful ML systems, and ST is no exception. In order to build a cascaded ST system, data for both the ASR and MT systems must be collected. For the ASR system, this will consist in audio segments paired with their corresponding transcriptions. For MT translation, parallel corpora needs to be collected, consisting in sentences and their translations into the target language. Additionally, it is also useful to collect vast quantities of monolingual data in order to train the language model component of the ASR system or to generate synthetic backtranslations for the MT system. Although obtaining these resources is not easy, the use of modern data crawling and data filtering techniques such as (Jorge, Martínez-Villaronga, et al. 2018) and (Bañón et al. 2020) means that one can obtain sufficient resources for training state-of-the-art ASR and MT architectures.

However, data scarcity remains a significant concern for building and evaluating ST systems, because ST data consists of triples of (audio, transcription, translation), which are significantly harder to obtain than ASR or MT data. This is even more critical for end-to-end ST systems, which require this ST data for training. The lack of ST training data can be partially alleviated, for example, by generating synthetic data using MT and Text-to-Speech (TTS) systems if not enough ST data is available. However, the lack of sufficient ST benchmarks remains a problem for ST evaluation, specially for language pairs that do not include English. Table 1.1 shows an overview of ST datasets at the beginning of 2019, the year this thesis was started.

As it can be observed, a very limited amount of ST data was available, and the majority of it consists in English audio data or English target translations. ST data for non-English language pairs is scarce, so it can be used only in low-resource scenarios and for evaluation. The lack of suitable ST evaluation sets is a limiting factor for ST research, as many of the insights obtained with high-resource English-centric settings might not apply for realistic scenarios involving other language pairs. Apart from this, another issue present in 2019 was the use of the Fisher and Callhome corpus (Post et al. 2013) as the standard benchmark for ST, which is not freely available to researchers.

The need for a reliable, multilingual ST that is freely available to the ST community motivated the creation of the Europarl-ST corpus, the first major contribution of this thesis, which is described in Paper 1.

Table 1.1: Overview of ST resources at the start of 2019. Reproduced with permission from (Di Gangi et al. 2019).

Corpus	Languages	Hours
(Niehues et al. 2018)	En→De	273
(Kocabiyikoglu, Besacier, and Kraif 2018)	En→Fr	236
(Tohyama et al. 2005)	En↔Jp	182
(Paulik and Alex Waibel 2009)	En→Es	111
	Es→En	105
(Post et al. 2013)	En→Es	38
(Stüker et al. 2012)	De→En	37
(Shimizu et al. 2014)	En↔Jp	22
(Federmann and Lewis 2017)	En → Zh	22
(Bendazzoli and Sandrelli 2007)	En↔It/Es	18
	It↔Es	18
(Bérard et al. 2016)	Fr→En	17
(Federmann and Lewis 2016)	En↔Fr/De	8
(Woldeyohannis, Besacier, and Meshesha 2017)	Am→En	7
(Godard et al. 2018)	Mboshi→Fr	4

3.5 Evaluation of results

Streaming ST systems are typically evaluated across two axes: translation quality and latency. Translation quality is typically understood as the *goodness* of the translation, whereas latency is the elapsed time between a word being spoken and its translation being generated by the system. These two objectives are at odds with each other, because translation quality is heavily influenced by the amount of context available for translating each word, but waiting for more context implies an additional waiting time before the translation is available. Thus, there exists a latency-quality trade-off that must be adjusted depending on the requirements of the specific solution being developed.

The question of how to best assess the translation quality of MT systems remains open. Manual evaluation, that is, evaluation made by humans about the quality of the translated text, could very well be the best evaluation measure, but it has the disadvantage of needing a human to carry out the task. Carrying out manual evaluations every time we define a new system configuration and translate a large amount of test sentences quickly becomes unfeasible.

This has given rise to search for *automatic evaluation* metrics that are ideally correlated with human judgment. Automatic evaluation is carried out by

comparing the output of a system with a *reference translation* produced by a human. The most basic evaluation metric is the *precision*, the ratio between correct output words (shared words between the system output and the reference translation) and the number of words present on the output sentence. The problem with this metric is that it does not penalize short sentences and therefore can be easily fooled [1]. A better evaluation measure that takes the idea of precision into account together with hypothesis length is the *Bilingual Evaluation Understudy* (BLEU) (Papineni et al. 2002) score. The BLEU score computes a modified precision, p_n, at different n-gram levels. Unlike regular precision, the clipped-precision used by the BLEU metric requires that an n-gram appears the same number of times both in the reference translation and in the candidate translation. If a certain n-gram appears more times in the candidate than in the reference, it will only be counted as correct as many times as it appears in the reference. BLEU is a corpus-level measure which is computed by checking and clipping the number of matching n-grams for each pair of hypothesis and reference sentences. The number of matches is added over all sentences of the test set and the precision is then computed:

$$\text{AveragePrecision}(N) = \frac{1}{N}\sum_{n=1}^{N} \log p_n. \tag{1.19}$$

This score also includes a Brevity Penalty term that is detrimental if the length of the candidate translation (c) is smaller than the length of the reference (r):

$$\text{BrevityPenalty} = \begin{cases} 1 & : \text{if } c > r \\ \exp(1 - \frac{r}{c}) & : \text{if } c \leq r \end{cases} \tag{1.20}$$

The usual definition of BLEU is computed up to n-grams of order 4, such that

$$\text{BLEU}(4) = \text{BrevityPenalty} * \text{AveragePrecision}(4). \tag{1.21}$$

The final result is a value ranging from 0 to 1, higher values are better. The value is usually multiplied by 100 to obtain better readability. BLEU belongs to the family of *string* metrics, that is, metrics that estimate the translation quality by comparing the words of the hypothesis and the reference translation and computing statistics such as precision and recall. BLEU is the most

[1] A system that emitted the translation "the" for any given input sentence would achieve an unusually high precision, since "the" is the most common English word. (Koehn 2010)

popular string-based metric, but there exist others such as TER (Snover et al. 2006) and chrF (Popović 2015).

More recently, a new type of metrics called *neural* metrics has started to gather widespread attention. Instead of count-based statistics, neural metrics typically work by using a neural network that takes the hypothesis and the reference translation as input, and tries to estimate the quality of the translation. These models are usually trained on a supervised manner using Direct Assessment (DA) data, which contains triples of source sentence, hypothesis translation and reference translation, as well as a human scored assigned to the hypothesis translation. The latest research concludes that neural metrics are more correlated with human judgements than string-based metrics (Freitag et al. 2022). Neural metrics have also been shown to be susceptible to certain types of errors (Amrhein and Sennrich 2022) and bias (Gowda et al. 2021), so they need to be used with special care.

Likewise, the second evaluation criteria for streaming ST, latency, is not trivial to measure. In human interpretation studies, Ear-Voice Span (EVS), also known as time lag (Barik 1973), is the main measure used to understand interpreter latency. EVS is computed by measuring the elapsed time between when a word is uttered by the speaker and when the interpreter utters its translation. This measure is computed manually, by tagging word pairs with similar semantic meaning in both the original speaker and the interpreters interventions. The annotation process for EVS is a time-consuming step, and there are disagreements over the exact specifics of how EVS should be computed (Collard 2019).

Online ST uses automatic measures whose goal is the same as EVS, to measure the latency of the translation. Average Proportion (AP) (Cho and Esipova 2016), Average Lagging (AL) (Ma et al. 2019) and Differentiable Average Lagging (DAL) (Cherry and Foster 2019) are the most commonly used computationally agnostic measures. For AP/AL/DAL, latency is measured by assuming a fixed, monotonic alignment between source and target words. These measures are computed independently for each sentence pair, and the average over all sentences is reported. However, translating sentences in isolation and then measuring latency is not representative of a system's behaviour when faced with the real task of streaming ST. Moreover, as previously mentioned, AP/AL/DAL cannot be directly applied to the streaming scenario (Schneider and Alexander Waibel 2020). The lack of a realistic framework for evaluating streaming ST latency motivated the work carried out in Paper 4.

4 List of publications

4.1 Paper 1

Title	Europarl-ST: A Multilingual corpus for Speech Translation of Parliamentary Debates
Authors	Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, Javier Jorge, Nahuel Roselló Adrià Giménez, Albert Sanchis, Jorge Civera, Alfons Juan
Year	2020
Type	International Conference - GGS Class 2
DOI	10.1109/ICASSP40776.2020.9054626
Name	Proc. of ICASSP 2020
Pages	8229-8233

Current ST research is often hampered by the lack of specific data resources for this task, as currently available ST datasets are restricted to a limited set of language pairs. This work presents Europarl-ST, a novel multilingual ST corpus containing paired audio-text samples from and into 6 European languages (English, German, French, Spanish, Italian, Portuguese), for a total of 30 different translation directions. This corpus has been compiled using the debates held in the European Parliament in the period between 2008 and 2012. The corpus creation process is described in detail, which has been carefully aligned and filtered in order to provide a reliable benchmark for streaming ST systems.

The paper presents a series of automatic speech recognition, machine translation and spoken language translation experiments that highlight the potential of this new resource, carried out using the English, German, French and Spanish sets, for a total of 12 ST directions. The results show the usefulness of this resource for both domain adaptation and evaluation, as well as highlighting some of the challenges to be solved on the road to streaming ST.

4.2 Paper 2

Title	Direct Segmentation Models for Streaming Speech Translation
Authors	Javier Iranzo-Sánchez, Adrià Giménez, Joan Albert Silvestre-Cerdà, Pau Baquero-Arnal, Jorge Civera, Alfons Juan
Year	2020
Type	International Conference - GGS Class 1
DOI	10.18653/v1/2020.emnlp-main.206
Name	Proc. of EMNLP 2020
Pages	2599-2611

This paper studies how to optimize the processing and segmentation of the ASR system output so that the downstream MT performance is maximized. Specifically, this publication is focused on studying the segmentation problem for the streaming scenario. We introduce a novel neural segmenter architecture, Direct Segmentation (DS), which considers the segmentation process as a classification problem. Using a sliding window approach, for every position of the ASR stream, the segmenter decides whether or not to produce a chunk by using a fixed local history and a small look-ahead window. The performance of this approach is evaluated on the previously introduced Europarl-ST corpus, by training an offline MT system and testing its performance when combined with different segmenters, for the English $\leftrightarrow$ {German, French, Spanish} directions. Experiments are also performed showing that adding audio features to the segmenter improves performance.

The proposed architecture is computationally efficient while outperforming other segmentation approaches, and is able to work straight-out-of-the box in the streaming scenario. Additionally, the work studies how the MT training data should be processed so that it better matches the ASR transcriptions, avoiding the need for an intermediate inverse text normalization step.

4.3 Paper 3

Title	Streaming cascade-based speech translation leveraged by a direct segmentation model
Authors	Javier Iranzo-Sánchez, Javier Jorge, Pau Baquero-Arnal Joan Albert Silvestre-Cerdà, Adrià Giménez, Jorge Civera Albert Sanchis, Alfons Juan
Year	2021
Type	Journal - JCR Q1 Artificial Intelligence (18/227)
DOI	10.1016/j.neunet.2021.05.013
Name	Neural Networks, 142
Pages	303–315

This paper extends the previous one by moving from a simulated streaming scenario into a real one. Previously, we worked on a simulated scenario which used the fixed transcriptions of an ASR system and an offline MT system to asses the feasibility of the proposed DS system. This work uses streaming ASR and MT systems whose hyperparameters are jointly optimized with the DS segmenter in order to maximize the latency-quality trade-off of the streaming process, and the streaming scenario is tested by using as input the raw, unsegmented interventions of the Europarl-ST corpus.

The experiments are carried out with a Spanish ASR system, and Spanish-English and Spanish-French MT systems, which highlights how Europarl-ST enables non-English centric ST. Two online MT approaches are tested, MMAH and wait-k translation, and our experiments show how, for these settings, wait-k is the preferred approach in both quality and latency.

4.4 Paper 4

Title	Stream-level Latency Evaluation for Simultaneous Machine Translation
Authors	Javier Iranzo-Sánchez, Jorge Civera, Alfons Juan
Year	2021
Type	International Conference
DOI	10.18653/v1/2021.findings-emnlp.58
Name	Findings of EMNLP 2021
Pages	664-670

This paper introduces a novel evaluation procedure for streaming MT. Standard online MT metrics only work with short audio segments, evaluated in

isolation, and do not take into account the sequential nature of the streaming scenario. Our proposed streaming evaluation method fixes these issues, and as a bonus, it can be applied to the standard metrics used for online MT with a small modification. Our proposal keeps track of a global latency score across the entire translation process, and uses a re-alignment step that matches translated words with the correct reference segment.

A significant advantage of our proposal is that the evaluation procedure is not system/segmentation dependent and can be used to compare different systems, as well as maintaining the original interpretability of the metrics. Comparative experiments show that, unlike competing approaches, our proposal correctly ranks systems based on their latency, as well as keeping the previously mentioned properties.

4.5 Paper 5

Title	From Simultaneous to Streaming Machine Translation by Leveraging Streaming History
Authors	Javier Iranzo-Sánchez, Jorge Civera, Alfons Juan
Year	2022
Type	International Conference - GGS Class 1
DOI	10.18653/v1/2022.acl-long.480
Name	Proc. of ACL 2022
Pages	6972-6985

This paper presents a general methodology for building context-aware state-of-the-art streaming MT systems, by incorporating the previously developed DS streaming segmenter and using our proposed streaming metrics for evaluation. This publication takes advantage of the insights developed in the previous publications in order to build a strong streaming baseline MT system, and improves it with a novel context-aware training methodology which obtains significant improvements. Further improvements are also obtained with a proposed Partial Bidirectional Encoder (PBE) that has access to a larger portion of the input prefix.

Our approach is similar to the concatenative approach used in context-aware MT, and uses a sliding window which contains the previous streaming history that has been produced during the translation process. History-augmented training samples are constructed from document-level corpora, and at inference time, the real streaming history is used. Extensive experiments are carried on IWSLT English-German data in order to study the behaviour of the model and

optimize the latency-quality trade-off. Using our proposed streaming latency metrics, our system is compared with the ACT streaming approach (Schneider and Alexander Waibel 2020) and the submissions to the IWSLT 2020 simultaneous track (Ansari et al. 2020), achieving a similar level of quality for a fraction of the latency.

References

Amrhein, Chantal and Rico Sennrich (2022). "Identifying Weaknesses in Machine Translation Metrics Through Minimum Bayes Risk Decoding: A Case Study for COMET". In: *Proc. of AACL-IJCNLP*. ACL, pp. 1125–1141 (cit. on p. 22).

Anastasopoulos, Antonios et al. (2022). "Findings of the IWSLT 2022 Evaluation Campaign". In: *Proc. of IWSLT*. ACL, pp. 98–157 (cit. on pp. 2, 4).

Ansari, Ebrahim et al. (2020). "FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN". In: *Proc. of IWSLT*. Online: ACL, pp. 1–34 (cit. on p. 27).

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (cit. on p. 13).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*. Ed. by Yoshua Bengio and Yann LeCun (cit. on p. 2).

Bañón, Marta et al. (2020). "ParaCrawl: Web-Scale Acquisition of Parallel Corpora". In: *Proc. of ACL*. Association for Computational Linguistics, pp. 4555–4567 (cit. on p. 19).

Barik, H. (1973). "Simultaneous Interpretation: Temporal and Quantitative Data". In: *Language and Speech* 16, pp. 237–270 (cit. on p. 22).

Bendazzoli, Claudio and Annalisa Sandrelli (2007). "An approach to corpus-based interpreting studies: developing EPIC (European Parliament Interpreting Corpus)". In: *Challenges of Multidimensional Translation, Proceedings of the Marie Curie Euroconferences MuTra: Challenges of Multidimensional Translation* (cit. on p. 20).

Bérard, Alexandre et al. (2016). "Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation". In: *Proceedings of the NIPS Workshop on end-to-end learning for speech and audio processing*. Barcelona, Spain (cit. on p. 20).

Bottou, Léon and Olivier Bousquet (2007). "The Tradeoffs of Large Scale Learning". In: *Proc. of NIPS*, pp. 161–168 (cit. on p. 7).

Brown, Peter F., John Cocke, et al. (1990). "A Statistical Approach to Machine Translation". In: *Comput. Linguistics* 16.2, pp. 79–85 (cit. on p. 2).

Brown, Peter F., Stephen Della Pietra, et al. (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Comput. Linguistics* 19.2, pp. 263–311 (cit. on p. 2).

Chan, William et al. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. of ICASSP*. IEEE, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621 (cit. on p. 2).

Cherry, Colin and George Foster (2019). "Thinking Slow about Latency Evaluation for Simultaneous Machine Translation". In: *arXiv:1906.00048* (cit. on p. 22).

Cho, Kyunghyun and Masha Esipova (2016). "Can neural machine translation do simultaneous translation?" In: *arXiv preprint arXiv:1606.02012* (cit. on p. 22).

Collard, Camille (2019). "A corpus-based study of simultaneous interpreting with special reference to sex". PhD thesis. Ghent University (cit. on p. 22).

Dahl, George E. et al. (2011). "Large vocabulary continuous speech recognition with context-dependent DBN-HMMS". In: *Proc. of ICASSP*. IEEE, pp. 4688–4691 (cit. on p. 2).

Di Gangi, Mattia et al. (2019). "MuST-C: a Multilingual Speech Translation Corpus". In: *NAACL-HLT 2019* (cit. on p. 20).

European Commission, Special Eurobarometer (2012). "Europeans and their languages". In: *Special Eurobarometer* 386.June (cit. on p. 2).

Federmann, Christian and William D. Lewis (2016). "Microsoft Speech Language Translation (MSLT) Corpus: The IWSLT 2016 release for English, French and German". In: *Proc. of IWSLT*. International Workshop on Spoken Language Translation (cit. on p. 20).

– (2017). "The Microsoft Speech Language Translation (MSLT) Corpus for Chinese and Japanese: Conversational Test data for Machine Translation and Speech Recognition". In: *Proc. of MTSUMMIT*, pp. 72–85 (cit. on p. 20).

Freitag, Markus et al. (2022). "Results of WMT22 Metrics Shared Task: Stop Using BLEU – Neural Metrics Are Better and More Robust". In: *Proc. of WMT*. ACL, pp. 46–68 (cit. on p. 22).

Gaido, Marco et al. (2021). "Beyond Voice Activity Detection: Hybrid Audio Segmentation for Direct Speech Translation". In: *Proc. of ICNLSP*. ACL, pp. 55–62 (cit. on p. 4).

Godard, Pierre et al. (2018). "A Very Low Resource Language Speech Corpus for Computational Language Documentation Experiments". In: *Proc. of LREC*. ELRA (cit. on p. 20).

Gowda, Thamme et al. (2021). "Macro-Average: Rare Types Are Important Too". In: *Proc. of NAACL*. ACL, pp. 1138–1157 (cit. on p. 22).

He, Kaiming et al. (2016). "Deep Residual Learning for Image Recognition". In: *Proc. of CVPR*. IEEE Computer Society, pp. 770–778 (cit. on p. 13).

Iranzo-Sánchez, Javier (2018). "A comparative study of Neural Machine Translation frameworks for the automatic translation of open data resources". B.S. Thesis. Universitat Politècnica de València (cit. on p. 5).

– (2019). "Online Multilingual Neural Machine Translation". M.S. Thesis. Universitat Politècnica de València (cit. on p. 5).

Irie, Kazuki et al. (2019). "Language Modeling with Deep Transformers". In: *Proc. Interspeech 2019*. ISCA, pp. 3905–3909. DOI: 10.21437/Interspeech.2019-2225 (cit. on p. 2).

Jorge, Javier, Adrià Giménez, et al. (Jan. 1, 2020). "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP*. IEEE (cit. on p. 2).

Jorge, Javier, Adrià Martínez-Villaronga, et al. (2018). "MLLP-UPV and RWTH Aachen Spanish ASR Systems for the IberSpeech-RTVE 2018 Speech-to-Text Transcription Challenge". In: *Proc. IberSPEECH 2018*, pp. 257–261 (cit. on p. 19).

Kocabiyikoglu, Ali Can, Laurent Besacier, and Olivier Kraif (2018). "Augmenting Librispeech with French Translations: A Multimodal Corpus for Direct Speech Translation Evaluation". In: *Proc. of LREC* (cit. on p. 20).

Koehn, Philipp (2010). *Statistical Machine Translation*. 1st. New York, NY, USA: Cambridge University Press (cit. on p. 21).

Koehn, Philipp, Franz Josef Och, and Daniel Marcu (2003). "Statistical Phrase-Based Translation". In: *Proc. of HLT-NAACL*. Ed. by Marti A. Hearst and Mari Ostendorf. ACL (cit. on p. 2).

Ma, Mingbo et al. (2019). "STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework". In: *Proc. of ACL*. ACL, pp. 3025–3036. DOI: 10.18653/v1/P19-1289 (cit. on p. 22).

Mitchell, Tom M. (1997). *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill (cit. on p. 5).

Müller, Mathias et al. (2018). "A Large-Scale Test Set for the Evaluation of Context-Aware Pronoun Translation in Neural Machine Translation". In: *Proc. of WMT*. Brussels, Belgium: Association for Computational Linguistics (cit. on p. 5).

Niehues, Jan et al. (2018). "The IWSLT 2018 Evaluation Campaign". In: *Proc. of IWSLT*, pp. 2–6 (cit. on p. 20).

Papineni, Kishore et al. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *ACL 2002* (cit. on p. 21).

Park, Daniel S. et al. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. Interspeech*, pp. 2613–2617 (cit. on p. 2).

Paulik, Matthias and Alex Waibel (2009). "Automatic translation from parallel speech: Simultaneous interpretation as MT training data". In: *Proc. of ASRU*. IEEE, pp. 496–501 (cit. on p. 20).

Popović, Maja (2015). "chrF: character n-gram F-score for automatic MT evaluation". In: *Proc. of WMT*. ACL, pp. 392–395. DOI: 10.18653/v1/W15-3049 (cit. on p. 22).

Post, Matt et al. (2013). "Improved speech-to-text translation with the Fisher and Callhome Spanish-English speech translation corpus". In: *Proc. of IWSLT* (cit. on pp. 19, 20).

Rangarajan Sridhar, Vivek Kumar et al. (2013). "Segmentation Strategies for Streaming Speech Translation". In: *Proc. of NAACL*. ACL, pp. 230–238 (cit. on p. 4).

Schneider, Felix and Alexander Waibel (2020). "Towards Stream Translation: Adaptive Computation Time for Simultaneous Machine Translation". In: *Proc. of IWSLT*. ACL, pp. 228–236 (cit. on pp. 4, 22, 27).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *ACL 2016* (cit. on p. 18).

Shimizu, Hiroaki et al. (2014). "Collection of a Simultaneous Translation Corpus for Comparative Analysis". In: *Proc. of LREC*, pp. 670–673 (cit. on p. 20).

Snover, Matthew et al. (2006). "A Study of Translation Edit Rate with Targeted Human Annotation". In: *Proc. of AMTA*. AMTA, pp. 223–231 (cit. on p. 22).

Stüker, Sebastian et al. (2012). "The KIT Lecture Corpus for Speech Translation". In: *Proc. of LREC*, pp. 3409–3414 (cit. on p. 20).

Tiedemann, Jörg and Yves Scherrer (2017). "Neural Machine Translation with Extended Context". In: *Proc. of DiscoMT@EMNLP*. ACL, pp. 82–92 (cit. on p. 5).

Tohyama, Hitomi et al. (2005). "Construction and utilization of bilingual speech corpus for simultaneous machine interpretation research". In: *Proc. of INTERSPEECH*. ISCA, pp. 1585–1588 (cit. on p. 20).

Tsiamas, Ioannis et al. (2022). "SHAS: Approaching optimal Segmentation for End-to-End Speech Translation". In: *Proc. Interspeech 2022*, pp. 106–110 (cit. on p. 4).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS*. Ed. by Isabelle Guyon et al., pp. 5998–6008 (cit. on pp. 2, 10, 12).

Voita, Elena, Rico Sennrich, and Ivan Titov (2019a). "Context-Aware Monolingual Repair for Neural Machine Translation". In: *Proc. of EMNLP-IJCNLP*. ACL, pp. 877–886 (cit. on p. 5).

– (2019b). "When a Good Translation is Wrong in Context: Context-Aware Machine Translation Improves on Deixis, Ellipsis, and Lexical Cohesion". In: *Proc. of ACL*. ACL, pp. 1198–1212 (cit. on p. 5).

Woldeyohannis, Michael Melese, Laurent Besacier, and Million Meshesha (2017). "A Corpus for Amharic-English Speech Translation: The Case of Tourism Domain". In: *Proc. of ICT4DA*. Ed. by Fisseha Mekuria et al. Vol. 244. Springer, pp. 129–139 (cit. on p. 20).

Chapter 2

Selected Papers

1 Europarl-ST: A Multilingual corpus for Speech Translation of Parliamentary Debates

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

ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8229-8233

Barcelona (Spain)

10.1109/ICASSP40776.2020.9054626

4-8 May 2020

Europarl-ST: A Multilingual corpus for Speech Translation of Parliamentary Debates

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

Abstract

Current research into spoken language translation (SLT), or speech-to-text translation, is often hampered by the lack of specific data resources for this task, as currently available SLT datasets are restricted to a limited set of language pairs. In this paper we present *Europarl-ST*, a novel multilingual SLT corpus containing paired audio-text samples for SLT from and into 6 European languages, for a total of 30 different translation directions. This corpus has been compiled using the debates held in the European Parliament in the period between 2008 and 2012. This paper describes the corpus creation process and presents a series of automatic speech recognition, machine translation and spoken language translation experiments that highlight the potential of this new resource. The corpus is released under a Creative Commons license and is freely accessible and downloadable.

Keywords: speech translation, spoken language translation, automatic speech recognition, machine translation, multilingual corpus

1.1 Introduction

The significant developments in the automatic speech recognition (ASR) and machine translation (MT) fields in the last five years, which have been mainly driven by advances in deep learning models and greater data availability, have picked up interest in spoken language translation (SLT) as the natural convergence of the two previous fields.

However, SLT is far from solved. Two approaches are currently used: cascade (Sperber, Niehues, and Waibel 2017; Cho, Niehues, and Waibel 2017; E. Matusov et al. 2018) and end-to-end models (Weiss et al. 2017; Sperber, Neubig, et al. 2019; Salesky, Sperber, and Black 2019), without one being clearly adopted by the community. The latest IWSLT 2018 evaluation campaign showed that the cascade approach outperforms end-to-end models (Niehues et al. 2018), but recent developments in the area are shrinking that gap (Jia et al. 2019). The performance of SLT, and especially end-to-end SLT models, is limited by the lack of SLT corpora when compared with the more resource-rich ASR and MT fields. Furthermore, most of the existing SLT corpora are limited to only English speech data paired with translations into other languages, such as the recently released MuST-C corpus (Di Gangi et al. 2019). This fact limits the SLT research than could be carried out in language pairs other than English. Moreover, recent studies report their main results using either the paid Fisher/Callhome corpora (Sperber, Niehues, and Waibel 2017; Weiss et al. 2017; Sperber, Neubig, et al. 2019; Salesky, Sperber, and Black 2019; Post et al. 2013), or private propietary datasets (Jia et al. 2019), which limits reproducibility for the research community.

In order to alleviate these problems, we have created the *Europarl-ST* corpus out of European Parliament (EP) debates and their official transcriptions and translations. To our knowledge, Europarl-ST is the first fully self-contained, publicly available corpus with both, multiple (speech) source and target languages, which will also enable further research into multilingual SLT (cf. (Boito et al. 2020)). The Europarl-ST corpus is released under a Creative Commons Attribution-NonCommercial 4.0 International license (CC BY-NC 4.0), and can be freely accessed and downloaded at `www.mllp.upv.es/europarl-st`.

1.2 Data collection and processing

The corpus has been created using the publicly available videos from European Parliament debates[1]. In order to ease the access to the different attributes of each debate the LinkedEP database is used (Van Aggelen et al. 2017). The basic unit of this corpus is a *speech*, an intervention made by a single speaker at the Parliament.

The EP debates suffer from missing videos, inaccurate timestamps and, as of 2011, many translations into languages other than English are missing. Indeed, after 2012, the translation of EP debates is not available. Additional data is discarded when constructing the Europarl-ST corpus, since in order to build

[1] http://www.europarl.europa.eu/plenary/en/debates-video.html

Table 2.1: Number of speech hours after each step of the data filtering pipeline, and CER of the filtered data sets.

	Initial	Step 1	Step 2	CER
De	207	149	44	10.7
En	346	252	120	12.9
Es	80	59	34	9.1
Fr	183	132	47	10.7

a corpus of audio-transcription-translation triples, it is necessary to properly define forced audio-text and text-text sentence alignments, and intra-sentence word-alignment.

For this initial release of the corpus, experiments are reported from and into English (En), German (De), French (Fr) and Spanish (Es), since these languages accumulate a larger number of speech hours. Additional languages, such as Italian and Portuguese, will also be included in the initial release, but experimental results are not reported due to time constraints.

Audio-to-text alignment and data filtering

One of the challenges processing this corpus is that timestamps provided for the EP speeches can be wildly inaccurate, and as a side-effect, they often contain fragments from both the preceding and following speeches. In order to ameliorate this, first we carried out a Speaker Diarization (SD) step for each speech using the *LIUM SpkDiarization* (Rouvier et al. 2013) toolkit. Second, for each speech, the longest sequence of audio segments belonging to the same speaker was clipped, making the assumption that it does correspond to the actual intervention of the speaker of this speech. Finally, a forced alignment of the clipped audio segments was carried out against their corresponding transcriptions to obtain correct word timestamps. Forced alignments were carried out using the TLK toolkit's decoder (Agua et al. 2014) and the FF-DNN acoustic models (AM) described in Section 1.3, restricting the search graph of the decoder to the provided transcription. As a result of the procedure describe above (Step 1), around 28% of the original audio data was discarded (see Table 2.1 for language-based statistics).

Next, in order to produce a reliable corpus than could be used to both train and evaluate models, a second data filtering step was carried out based on character error rate (CER) computed at the speech level. First, we apply ASR over all speeches, using the ASR system described in Section 1.3. Second, we

measure how much the recognition outputs differ from the provided reference transcriptions by computing CER values. Our aim is to eliminate speeches that exhibit significant amounts of non-verbatim transcriptions, as well as non-transcribed speech or unuttered transcripts that could be present either due to mistakes of the SD process or to annotation errors in the original data. In comparison with the well-known word error rate (WER) metric, the CER is more convenient for our purposes, as it better gauges the phonetic similarity between the recognised speech and the candidate reference transcripts, and alleviates the effect of ASR out-of-vocabulary words.

Finally, language-dependent CER thresholds were defined, 15% for French, German and Spanish and 20% for English, in order to exclude those speeches whose CER exceeded these thresholds. Thresholds were defined based on previous experience filtering crawled speech data. As a result of this filtering step (Step 2), around 40-70% of the audio data selected in the previous step was discarded (see Table 2.1 for detailed statistics). CER figures computed on the selected speeches after Step 2 are also provided in Table 2.1. These figures are an approximation to a quality assurance measure to ensure that only speeches with little or no noise are included into the corpus. At the end of this process, around 60-80% of the original data was filtered out.

Source-to-target text alignment

Each selected speech, both transcription and translation, is divided into sentences, using the *sentence-split.pl* script from the Moses toolkit (Koehn et al. 2007), that are aligned using Gargantua (Braune and Fraser 2010). Sentences longer than 20 seconds were split into shorter ones in order to accommodate the data for training purposes. Shorter sentences were generated by computing word-alignments using Fast-align (Dyer, Chahuneau, and Smith 2013) and pairing them to guarantee intra-sentence alignments. The statistics of the remaining data after text-aligning and excluding speeches with no translation into the respective target language are shown in Table 2.2. As observed in Table 2.2, this corpus is provided with segmentations, both at the speech and sentence level. The sentence-level segmentation is expected to be devoted to training purposes, while evaluations at the speech level are reported in Section 1.3.

A speaker-independent train/dev/test partition was defined, devoting approximately 3 hours of audio to each of the dev and test sets, and the rest was left as training data. The dev/test speakers are the same for language directions with the same source language. However, the number of speeches may differ because

Table 2.2: Statistics of the preprocessed Europarl-ST corpus.

Src	Trg	Speeches	Sent.	Hours	Src w.	Trg w.
	En	1521	18.1K	42	345K	409K
De	Es	863	10.2K	24	196K	242K
	Fr	839	9.6K	24	191K	265K
	De	3233	35.5K	89	811K	793K
En	Es	3184	34.4K	87	796K	865K
	Fr	3174	34.5K	87	794K	974K
	De	694	7.0K	20	193K	186K
Es	En	1131	11.2K	32	305K	307K
	Fr	684	6.9K	20	190K	225K
	De	832	9.6K	25	263K	227K
Fr	En	1306	15.1K	38	394K	371K
	Es	817	9.4K	25	260K	246K

for some speeches there are translations missing. The training data might be used to fine-tune and adapt out-of-domain models to this specific domain, or even to train basic in-domain ASR, MT and SLT models from scratch.

1.3 Experiments and results

This section introduces the setup used for the experiments performed with the Europarl-ST corpus. In addition to ASR and MT experiments, SLT experiments following a cascade approach, in which the output of an ASR system is used as input for an MT system, are reported. First, the performance of models trained on general domain data when applied to the Europarl-ST corpus are evaluated, and second, the usefulness of the Europarl-ST training data for adapting models to the EP specific domain is also assessed. More precisely, results of ASR, MT and SLT experiments are reported using the 4 selected languages (English, German, Spanish and French), for a total of 12 translation directions in the case of translation experiments. Results are reported in terms of WER for ASR experiments, and BLEU (Papineni et al. 2002) for MT and SLT experiments.

In order to properly compute BLEU, both the system hypothesis and the reference translation must have the same number of lines. However, in a SLT experiment, the number of lines will depend on the segmentation applied to the output of the ASR system in the cascade case, and the SLT system in the end-to-end case. Therefore, it is standard to re-segment the system hypothesis in order to get the same number of lines as in the reference. This re-segmentation

is performed with the *mwerSegmenter* (Evgeny Matusov et al. 2005), and then evaluated by computing case-sensitive BLEU (including punctuation signs) with SacreBLEU (Post 2018). All evaluations are carried out at the speech level, so re-segmentation is applied to both, MT and SLT experiments, in order to evaluate them under the same conditions.

ASR

General-purpose ASR systems for German (De), English (En), Spanish (Es) and French (Fr) were used to generate automatic transcripts for audio speeches in the development and test sets of each language pair. These automatic transcripts are the input text for subsequent MT systems within the SLT cascade approach.

These ASR systems are based on the hybrid deep neural network hidden Markov model (DNN-HMM) approach. Acoustic models, are generated using the TLK toolkit (Agua et al. 2014) to train feed-forward (FF) DNN-HMM models of three left-to-right tied triphone states, using 48 (De, Es, Fr) or 80-dimensional (En) Mel frequency cepstral coefficients (MFCCs) as input features. State tying was done by applying language-dependent classification and regression trees (CART), which resulted in 10K (Es, Fr) or 18K (De, En) tied triphone states. With the exception of the French ASR system which only features FF-DNNs, these models were used to bootstrap bidirectional long-short term memory (BLSTM) DNN models, the latter model trained using Tensorflow (*TensorFlow* 2018). For German, Spanish and French, we also trained fCMLLR AMs, so that these systems follow a two-step recognition process.

On the other hand, regarding the language models (LM), we used a linear combination of several n-gram LMs trained with SRILM (Stolcke 2002), combined with a recurrent NN (RNN) LM trained using the RNNLM toolkit (*The RNNLM Toolkit* 2012) (De, Es, Fr), or an LSTM LM trained with the CUED-RNNLM toolkit (Chen et al. 2016) (En). The vocabulary of these systems was restricted to 200K words. Table 2.3 shows overall statistics of the amount of training data that were used to train the acoustic models, in terms of speech hours, and the language models, in terms of sentences and words. The number of English words includes 294G words from Google Books counts.

Table 2.4 shows, for each SLT test set, WER figures computed from the ASR part only. Rows represent source (ASR) languages, whilst columns represent target (MT) languages. It is important to remind that the set of source speeches, though mostly overlapping, are different because the correspoding

Table 2.3: Statistics of AM and LM training data.

	Hours (K)	Sentences (M)	Words (G)
De	0.9	71	0.8
En	5.6	532	300
Es	0.8	24	0.7
Fr	0.7	110	1.8

Table 2.4: ASR results in terms of WER on the test sets.

	De	En	Es	Fr
De	–	19.8	19.8	19.9
En	17.2	–	17.2	17.1
Es	14.6	15.0	–	14.6
Fr	27.3	24.3	27.2	–

target text translation may not exist. Results show that most WER figures are below 20%, except in those pairs having French as input language. This is explained because the French ASR system does not feature BLSTM acoustic models, and it is the language with least acoustic resources.

MT

A Neural Machine Translation (NMT) system was built for each translation direction mainly using publicly available corpora from OPUS (Tiedemann 2012) and excluding the Europarl corpus to avoid data overlapping. The training data used in each language pair is shown in Table 2.23. This includes the list of corpora and the total number of sentences.

The corpora were preprocessed by applying 40K BPE (Sennrich, Haddow, and Birch 2016) operations, learnt jointly over the source and target data. The models follow the Transformer NMT architecture (Vaswani et al. 2017) and are trained using the Transformer BASE configuration using 4GPU machines and an initial learning rate of 5e−4, decayed using the inverse square root scheme. Once the training converges, a fine-tuning step was carried out using the training data generated in Section 1.2. To do so, we fix the learning rate to 5e−5, and we use a standard SGD optimizer instead of Adam. We measure performance on the dev set and stop training once the perplexity stops decreasing. Table 2.6 shows BLEU scores of the out-of-domain MT systems compared with those obtained by fine-tuning with the Europarl-ST training data shown between parenthesis. These MT systems are evaluated

Table 2.5: Training data used for the MT systems

Pair	Corpora	# sents(M)
De↔En	DGT,eubookshop TildeMODEl, Wikipedia	21.0
De↔Es	DGT, eubookshop, JRC-Acquis, TildeModel	14.3
De↔Fr	eubookshop, JRC-Acquis, TildeModel	14.3
En↔Es	commoncrawl, eubookshop, EU-TT2, UN, Wikipedia	21.1
En↔Fr	commoncrawl, giga, undoc, news-commentary	38.2
Es↔Fr	DGT, eubookshop, JRC-Acquis, UNPC	37.2

Table 2.6: BLEU scores of out-of-domain MT systems with reference transcriptions as input and fine-tuning BLEU scores between parenthesis.

	De	En	Es	Fr
De	–	32.6 (**36.3**)	26.8 (**29.3**)	23.2 (**27.1**)
En	33.6 (**37.6**)	–	46.3 (**48.2**)	34.7 (**39.2**)
Es	20.9 (**24.8**)	39.2 (**41.8**)	–	29.3 (**33.1**)
Fr	23.3 (**26.3**)	38.7 (**42.3**)	34.8 (**36.3**)	–

on automatic outputs generated from reference transcriptions as a standalone MT task.

The results vary depending on the amount of resources used for each system as well as the intrinsic difficulty of each translation direction. As observed, the fine-tuned systems trained on the Europarl-ST corpus provide very significant improvements over the out-of-domain systems, ranging from +1.9 up to +4.0 BLEU, which confirms the quality and usefulness of the training data.

SLT

This section presents the results of the SLT experiments following the cascade approach, in which we use the output of the ASR system as input for the MT system. The output of the ASR system is segmented based on detected silences. For this task, we will combine the ASR and MT models described in Sections 1.3 and 1.3. We use the fine-tuned MT systems as they outperform

Table 2.7: BLEU scores of cascade-based SLT experiments with fine-tuned models assessed on the test sets.

	De	En	Es	Fr
De	–	21.3	17.5	15.7
En	22.4	–	28.0	23.4
Es	15.6	26.5	–	22.0
Fr	15.3	25.4	23.2	–

the out-of-domain systems in all cases. The results of the SLT experiments are shown in Table 2.7.

Table 2.7 shows that BLEU scores in the SLT experiments are lower than those in the MT experiments. This is to be expected, as the MT system has to cope not only with error propagation from incorrect transcriptions, but also with a sub-optimal segmentation of the input which might not correspond with whole sentences. This could be improved with a specific segmentation and punctuation module (Cho, Niehues, and Waibel 2017). As expected, although the overall BLEU scores are lower, the ranking of the performance across translation directions is preserved, with MT systems that obtained the highest scores in the MT experiments, also obtaining the highest scores in the SLT experiments, and vice versa. Although SLT results are constrained by the complexity of this task, these results serve as a good starting baseline for future developments.

1.4 Conclusions

We have presented a novel SLT corpus built from European Parliament proceedings. The experiments presented have shown how our proposed filtering pipeline is able to extract good quality data that is useful both for evaluating the performance of out-of-domain systems in this task, as well as for system adaptation to the specific domain of parliamentary debates. We believe that the release of this multi-source and multi-target corpus will enable further research into multilingual SLT.

In terms of future work, the presented filtering pipeline can be extended to cover additional languages in the future. Additionally, we will study new filtering techniques to increase the amount of hours available per each language pair.

Finally, we also plan on gauging the performance of end-to-end models for this task, and compare it with cascade systems that use MT models adapted to the translation of ASR output. This adaptation can be carried out by training MT systems on real ASR output as source input (Peitz et al. 2012) or on simulated ASR output by applying noising techniques to the source side (Sperber, Niehues, and Waibel 2017).

References

Agua, Miguel A. del et al. (2014). "The Translectures-UPV Toolkit". In: *IberSpeech 2014* (cit. on pp. 39, 42).

Boito, Marcely Zanon et al. (2020). "MaSS: A Large and Clean Multilingual Corpus of Sentence-aligned Spoken Utterances Extracted from the Bible". In: *LREC 2020* (cit. on p. 38).

Braune, Fabienne and Alexander M. Fraser (2010). "Improved Unsupervised Sentence Alignment for Symmetrical and Asymmetrical Parallel Corpora". In: *COLING 2010* (cit. on p. 40).

Chen, Xi et al. (2016). "CUED-RNNLM — An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *ICASSP 2016* (cit. on p. 42).

Cho, Eunah, Jan Niehues, and Alex Waibel (2017). "NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation". In: (cit. on pp. 38, 45).

Di Gangi, Mattia et al. (2019). "MuST-C: a Multilingual Speech Translation Corpus". In: *NAACL-HLT 2019* (cit. on p. 38).

Dyer, Chris, Victor Chahuneau, and Noah A. Smith (2013). "A Simple, Fast, and Effective Reparameterization of IBM Model 2". In: *NAACL-HLT 2013* (cit. on p. 40).

Jia, Ye et al. (2019). "Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model". In: *Interspeech 2019*. DOI: 10.21437/Interspeech.2019-1951 (cit. on p. 38).

Koehn, Philipp et al. (2007). "Moses: Open Source Toolkit for Statistical Machine Translation". In: *ACL 2007* (cit. on p. 40).

Matusov, E. et al. (2018). "Neural Speech Translation at AppTek". In: *IWSLT 2018* (cit. on p. 38).

Matusov, Evgeny et al. (2005). "Evaluating machine translation output with automatic sentence segmentation". In: *IWSLT 2005* (cit. on p. 42).

Niehues, Jan et al. (2018). "The IWSLT 2018 Evaluation Campaign". In: *IWSLT 2018* (cit. on p. 38).

Papineni, Kishore et al. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *ACL 2002* (cit. on p. 41).

Peitz, Stephan et al. (2012). "Spoken language translation using automatically transcribed text in training". In: *IWSLT 2012* (cit. on p. 46).

Post, Matt (2018). "A Call for Clarity in Reporting BLEU Scores". In: *WMT18* (cit. on p. 42).

Post, Matt et al. (2013). "Improved speech-to-text translation with the Fisher and Callhome Spanish–English speech translation corpus". In: *IWSLT 2013* (cit. on p. 38).

Rouvier, Mickael et al. (2013). "An open-source state-of-the-art toolbox for broadcast news diarization". In: *INTERSPEECH 2013* (cit. on p. 39).

Salesky, Elizabeth, Matthias Sperber, and Alan W Black (2019). "Exploring Phoneme-Level Speech Representations for End-to-End Speech Translation". In: *ACL 2019* (cit. on p. 38).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *ACL 2016* (cit. on p. 43).

Sperber, Matthias, Graham Neubig, et al. (2019). "Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation". In: *Transactions of the Association for Computational Linguistics* 7, pp. 313–325. DOI: 10.1162/tacl_a_00270 (cit. on p. 38).

Sperber, Matthias, Jan Niehues, and Alex Waibel (2017). "Toward robust neural machine translation for noisy input sequences". In: *IWSLT 2017* (cit. on pp. 38, 46).

Stolcke, A. (Sept. 2002). "SRILM – an extensible language modeling toolkit". In: *ICSLP*. Denver, CO, USA, pp. 901–904 (cit. on p. 42).

TensorFlow (2018). https://www.tensorflow.org/ (cit. on p. 42).

The RNNLM Toolkit (2012). http://www.fit.vutbr.cz/~imikolov/rnnlm/ (cit. on p. 42).

Tiedemann, Jörg (2012). "Parallel Data, Tools and Interfaces in OPUS". In: *LREC 2012* (cit. on p. 43).

Van Aggelen, Astrid et al. (2017). "The debates of the European Parliament as linked open data". In: *Semantic Web* 8.2, pp. 271–281 (cit. on p. 38).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS*. Ed. by Isabelle Guyon et al., pp. 5998–6008 (cit. on p. 43).

Weiss, Ron J. et al. (2017). "Sequence-to-Sequence Models Can Directly Translate Foreign Speech". In: *Interspeech 2017* (cit. on p. 38).

2 Direct Segmentation Models for Streaming Speech Translation

Javier Iranzo-Sánchez, Adrià Giménez,
Joan Albert Silvestre-Cerdà, Pau Baquero-Arnal,
Jorge Civera, Alfons Juan

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 2599-2611

Online

10.18653/v1/2020.emnlp-main.206

16-18 November 2020

Direct Segmentation Models for Streaming Speech Translation

Javier Iranzo-Sánchez, Adrià Giménez,
Joan Albert Silvestre-Cerdà, Pau Baquero-Arnal,
Jorge Civera, Alfons Juan

Abstract

The cascade approach to Speech Translation (ST) is based on a pipeline that concatenates an Automatic Speech Recognition (ASR) system followed by a Machine Translation (MT) system. These systems are usually connected by a segmenter that splits the ASR output into, hopefully, semantically self-contained chunks to be fed into the MT system. This is specially challenging in the case of streaming ST, where latency requirements must also be taken into account. This work proposes novel segmentation models for streaming ST that incorporate not only textual, but also acoustic information to decide when the ASR output is split into a chunk. An extensive and thorough experimental setup is carried out on the Europarl-ST dataset to prove the contribution of acoustic information to the performance of the segmentation model in terms of BLEU score in a streaming ST scenario. Finally, comparative results with previous work also show the superiority of the segmentation models proposed in this work.

2.1 Introduction

ST is a field that is very closely aligned with ASR and MT, as it is their natural evolution to combine the advances in both areas. Thus, the goal is to obtain the translation of an utterance that has been spoken in a different language, without necessarily requiring the intermediate transcription. At the same time, it is desirable to have high quality translations without compromising the speed of the system. Although research into ST started in the nineties (Waibel et al. 1991), the field did not really take off until significant breakthroughs were achieved in ASR (Chan et al. 2016; Irie et al. 2019;

Park et al. 2019; Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020) and MT (Bahdanau, K. Cho, and Bengio 2015; Sennrich, Haddow, and Birch 2016b; Sennrich, Haddow, and Birch 2016a; Vaswani et al. 2017), mainly due to the introduction of deep neural networks (NN). Thanks to this, the field has recently attracted significant amounts of attention from both the research and industry communities, as the field is now mature enough that it has tangible and well-performing applications (Ma et al. 2019; Jia et al. 2019).

Currently, there are two main approaches to ST: cascade and end-to-end models. The goal of the end-to-end approach is to train a single system that is able to carry out the the entire translation process (Weiss et al. 2017; Berard et al. 2018; Gangi et al. 2019). This has only recently been possible thanks to advances in neural modeling. Due to a lack of ST data, different techniques such as pre-training and data augmentation (Bahar, Bieschke, and Ney 2019; Pino et al. 2019) have been used in order to alleviate this lack of data. It is important to remark that the currently proposed end-to-end models work in an offline manner and must process the entire input sequence. Therefore they cannot be used for a streaming scenario.

In the cascade approach, an ASR system transcribes the input speech signal, and this is fed to a downstream MT system that carries out the translation. The provided input to the MT step can be the 1-best hypothesis, but also n-best lists (Ng et al. 2016) or even lattices (Matusov and Ney 2011; Sperber, Neubig, et al. 2019). Additional techniques can also be used to improve the performance of the pipeline by better adapting the MT system to the expected input, such as training with transcribed text (Peitz et al. 2012) or chunking (Sperber, Jan Niehues, and Waibel 2017). The cascade approach can be used to take advantage of independent developments in ASR and MT, and it is significantly easier to train due to greater data availability. Thus, it is very relevant to study improvements for the ST cascade pipeline.

This work focuses on the effects of segmentation in streaming ST, as cascade systems still outperform end-to-end systems in standard setups (J. Niehues et al. 2019; Pino et al. 2019). Following a cascade approach, a streaming ST setup can be achieved with individual streaming ASR and MT components. Advances in neural streaming ASR (Zeyer, Schlüter, and Ney 2016; Jorge, Giménez, Iranzo-Sánchez, Civera, et al. 2019; Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020) allow the training of streaming models whose performance is very similar to offline ones. Recent advances in simultaneous MT show promise (Arivazhagan et al. 2019; Ma et al. 2019; Zheng et al. 2019), but current models have additional modelling and training complexity, and are not ready for translation of long streams of input text. For the scenario to be

considered (translation of parliamentary speeches, with an average duration of 100s), it is required for the ST systems to have a minimum throughput, but simultaneous translation is not required, so the translation of chunks[2] is still acceptable. In this case we prioritize quality over simultaneous translation, with a streaming ASR system followed by a standard offline MT system. This way, the resulting ST cascade system can provide transcribed words in real-time, that are eventually split into chunks to be translated by the offline MT system.

Following this approach, it is necessary to incorporate a segmentation component in the middle in order to split the output of the ASR system into (hopefully semantically self-contained) chunks that can be successfully processed by the MT model, while maintaining a balance between latency and quality. In (E. Cho, Jan Niehues, and Waibel 2012; E. Cho, Jan Niehues, Kilgour, et al. 2015; E. Cho, Jan Niehues, and Waibel 2017), the authors approach this problem by training a monolingual MT system that predicts punctuation marks, and then the ASR output is segmented into chunks based on this punctuation. Another approach is to segment the ASR output by using a language model (LM) that estimates the probability of a new chunk to start (Stolcke and Shriberg 1996; Wang, Finch, et al. 2016; Wang, Utiyama, and Sumita 2019). Binary classifiers with POS and reordering features have also been proposed to maximise MT quality (Oda et al. 2014; Siahbani et al. 2018). It is also possible to segment the ASR output using handcrafted heuristics such as those based on a fixed number of words per chunk (Cettolo and Federico 2006) or acoustic information (Fügen, Waibel, and Kolss 2007). These heuristic approaches present the disadvantage of being very domain and speaker specific. Alternatively, segmentation can also take advantage of information extracted from the MT decoding process itself, such as coverage (Kolss, Vogel, and Waibel 2008).

This work introduces a statistical framework for the problem of segmentation in ST, which incorporates both textual and acoustic information. Jointly with this, we propose a set of novel models that follow this framework, and a series of extensive experiments are carried out, which show how these new models outperform previously proposed segmentation models. In addition, we study the effect of the preprocessing scheme applied to the input of the MT system, the performance degradation explained by transcription and/or segmentation errors, and the latency due to the components of the ST system.

This paper is organized as follows. The next section describes the statistical framework of the segmenter in the streaming ST scenario. Section 2.3 fol-

[2] A chunk must be understood as a sequence of words.

lows, detailing how our proposed models are instantiated in this framework. Then, Section 2.4 describes the Europarl-ST dataset that is used in the experiments and the three main components of our streaming ST system based on a cascade approach: ASR and MT systems, and the segmentation models. Next, Section 2.5 reports a detailed evaluation in terms of BLEU score on the Europarl-ST dataset and comparative results with previous work, and latency figures. Finally, Section 4 draws the main conclusions of this work and devises future research lines.

2.2 Statistical framework

We define the streaming ST segmentation as a problem in which a continuous sequence of words provided as the output of an ASR system is segmented into chunks. These chunks will then be translated by a downstream MT system. The goal of the segmentation is to maximize the resulting translation accuracy while keeping latency under the response-time requirements of our streaming scenario.

Formally, the segmentation problem is the task of dividing a sequence of input words x_1^J into non-overlapping chunks. We will represent this with a sequence of split/non-split decisions, y_1^J, with $y_j = 1$ if the associated word x_j is the word that ends a chunk; and $y_j = 0$, otherwise. Optionally, additional input features can be used. In this work, we use word-based acoustic features (a_1^J) aligned with the sequence of words output by the ASR system.

Ideally, we would choose the segmentation $\hat{y}_1^J$ such that,

$$\begin{aligned}\hat{y}_1^J &= \arg\max_{y_1^J} p(y_1^J \mid x_1^J, a_1^J) \\ &= \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_1^{j-1}, x_1^J, a_1^J). \end{aligned} \tag{2.1}$$

However, in a streaming setup, we need to bound the sequence to w words into the future (hereafter, *future window*) to meet latency requirements

$$\hat{y}_1^J = \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_1^{j-1}, x_1^{j+w}, a_1^{j+w}). \tag{2.2}$$

Indeed, for computational reasons, the sequence is also bounded to h words into the past (hereafter, *history size*)

$$\hat{y}_1^J = \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_{j-h}^{j-1},\, x_{j-h}^{j+w},\, a_{j-h}^{j+w}). \tag{2.3}$$

Previous works in the literature can be stated as a particular case of the statistical framework defined above under certain assumptions.

LM based segmentation (Stolcke and Shriberg 1996; Wang, Finch, et al. 2016; Wang, Utiyama, and Sumita 2019). In this approach, an n-gram LM is used to compute the probability

$$P(y_j) = p(x_{j-n+1}^{j-1}, y_{j-n+1}^{j-1}, x_j, y_j, x_{j+1}^{j+n-1}) \tag{2.4}$$

where y_j is zero or one depending on the non-split or split decision to be taken, respectively. Split and non-split probabilities are combined into a function to decide whether a new chunk is defined after x_j

$$\hat{y}_j = \arg\max_{y_j} f(P(y_j)). \tag{2.5}$$

Monolingual MT segmentation (E. Cho, Jan Niehues, and Waibel 2012; E. Cho, Jan Niehues, Kilgour, et al. 2015; E. Cho, Jan Niehues, and Waibel 2017). Following this setup, a monolingual MT model translates from the original, (unpunctuated) words x_1^J into a new sequence z_1^J that contains segmentation information (via puntuaction marks). Each z_j can be understood as a pair (x_j, y_j), so the segmentation can be defined as an MT problem

$$\hat{z}_1^J = \arg\max_{z_1^J} p(z_1^J \mid x_1^J), \tag{2.6}$$

that basically reverts to

$$\hat{y}_1^J = \arg\max_{y_1^J} p(y_1^J \mid x_1^J) \tag{2.7}$$

since x_1^J is given.

In contrast with previous approaches, which treats segmentation as a by-product of a more general task, we propose a model that directly represents the probability of the split/non-split decision.

2.3 Direct Segmentation Model

Now that we have introduced the theoretical framework, we are going to describe our proposed segmentation model. This approach has the advantage of allowing a future dependency and consider not only textual, but also acoustic features. This provides the model with additional evidence for taking a better split/non-split decision.

First, the *Text* model computes text state vectors s_j^{j+w} that consider each word in x_{j-h}^{j+w} using an embedding function $e()$ and one or more recurrent layers, represented by the function $f_1()$. In order to incorporate information about previous decisions y_{j-h}^{j-1}, we create a new sequence $\tilde{x}_{j-h}^{j+w}$ by inserting an end-of-chunk token into the text input sequence every time a split decision has been taken. This sequence is bounded in length by h.

$$\tilde{x}_{j-h}^{j+w} = f_c(x_{j-h}^{j+w}, y_{j-h}^{j-1}). \tag{2.8}$$

Then, the state vectors are defined as follows

$$s_j^{j+w} = f_1(e(\tilde{x}_{j-h}^{j+w}))). \tag{2.9}$$

Next, the split probability is computed by concatenating the state vectors of the current word and those in the future window, and passing them through a series of feedforward layers $f_2()$

$$p(y_j \mid y_{j-h}^{j-1}, x_{j-h}^{j+w}) = f_2([s_j^{j+w}]). \tag{2.10}$$

If we include acoustic information, acoustic state vectors are computed using function $f_3()$

$$c_j^{j+w} = f_3(a_{j-h}^{j+w}) \tag{2.11}$$

and are concatenated with the text state vectors in order to compute the split/non-split probability

$$p(y_j|\, y_{j-h}^{j-1}, x_{j-h}^{j+w}, a_{j-h}^{j+w}) = f_2([s_j^{j+w}; c_j^{j+w}]). \tag{2.12}$$

In the case of audio information, we assess two variants, depending whether the acoustic sequence is passed through a RNN (*Audio w/ RNN*) or not (*Audio w/o RNN*). These word-based acoustic feature vectors are obtained as follows. The Audio w/o RNN (also referred to as *copy*) option extracts three acoustic features associated to each word: duration of the current word, duration of the previous silence (if any), and duration of the next silence (if any). These three features were selected due to their effectiveness to improve system performance, as well as being word-based features which therefore can be directly integrated into the proposed model. At training time, these features are obtained by carrying out a forced alignment between the audio and the reference transcription, while at testing time are directly provided by the ASR system.

The Audio w/ RNN option adds an independent RNN as f_3, to process the sequence a_{j-h}^{j+w} of three-dimensional acoustic feature vectors just described, and the acoustic state vectors are concatenated at word level with the text state vectors. Whenever acoustic features are used, first the Text model is pretrained and frozen, and then the feedforward network is updated with the acoustic data.

$$f_3(a_{j-h}^{j+w}) = \begin{cases} RNN(a_{j-h}^{j+w}) & \text{Audio w/ RNN} \\ a_j^{j+w} & \text{Audio w/o RNN} \end{cases} \tag{2.13}$$

Figure 2.3 provides an overview of the proposed model architecture behind the streaming ST segmenter. The part of the model inside the dashed-line boundary represents the Text model (see Eqs. 2.8 and 2.9), while the complete model that additionally considers acoustic information is represented outside the boundary for the Audio w/ RNN and Audio w/o RNN cases (see Eqs. 2.11 and 2.13). State vectors are concatenated right before the feed-forward network (FFN). In this way, Eq. 2.10 computes the split probability for the Text-only model, while Eq. 2.12 does the same for the Audio models.

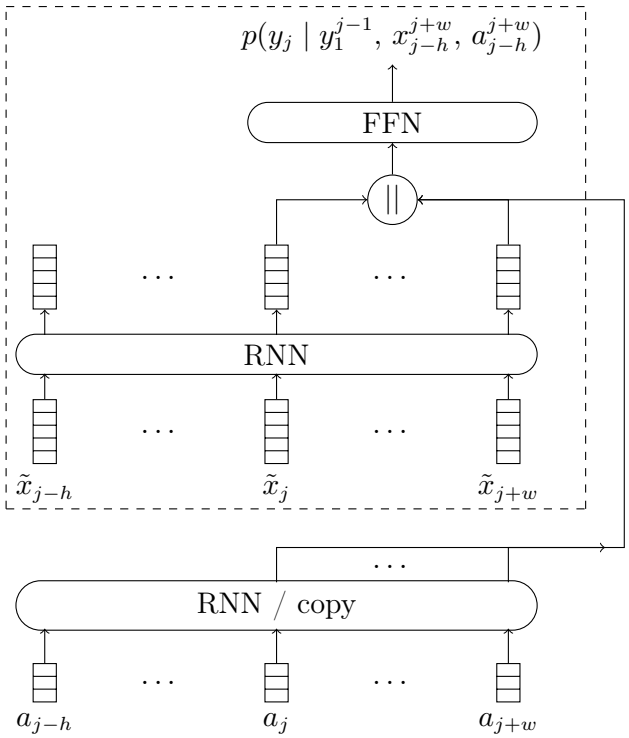

Figure 2.1: Overview of the model architecture for the streaming ST segmenter. The dashed-line boundary separates the Text model including word embeddings, RNN and state vectors, from the two possible Audio models, RNN and copy, outside the boundary.

Table 2.8: Basic statistics of the Europarl-ST corpus for the training, development and test partitions for the six language pairs involved in the evaluation.

ST Direction	Training			Development		
	Videos	Kwords		Videos	Kwords	
		Source	Target		Source	Target
En-De	2937	753	730	134	29	28
En-Es	2926	738	800	131	29	31
En-Fr	2918	738	901	132	29	34
De-En	1082	245	289	218	50	58
Es-En	727	203	200	202	53	53
Fr-En	1053	328	395	148	39	36

ST Direction	Test		
	Videos	Kwords	
		Source	Target
En-De	126	28	27
En-Es	127	28	31
En-Fr	124	72	33
De-En	226	52	59
Es-En	206	50	50
Fr-En	166	48	45

2.4 Experimental setup

To study the effects of our streaming ST segmenter in terms of BLEU score (Papineni et al. 2002), state-of-the-art ASR and MT systems were trained to perform ST from German (De), Spanish (Es) and French (Fr) into English (En), and vice versa. ASR and MT systems were treated as black boxes in order to focus our efforts on evaluating the proposed streaming ST segmentation models on the recently released and publicly available Europarl-ST corpus (Iranzo-Sánchez et al. 2020). Basic statistics of the six language pairs of the Europarl-ST corpus involved in the evaluation are shown in Table 2.19.

ASR systems

In our cascade ST setting, input speech signal is segmented into speech/non-speech regions using a Gaussian Mixture Model - Hidden Markov Model based voice activity detection (VAD) system (Silvestre-Cerdà et al. 2012), which will be referred to as the baseline segmentation system. Detected speech chunks are delivered to our general-purpose hybrid ASR systems for German (De), English (En), Spanish (Es) and French (Fr).

On the one hand, acoustic models (AM) were generated using the TLK toolkit (Agua et al. 2014) to train Feed-Forward deep neural Network - Hidden Markov Models (FFN-HMM). These models were used to bootstrap bidirectional long-short term memory (BLSTM) NN models (Zeyer, Doetsch, et al. 2017), trained using Tensorflow (Abadi et al. 2015), except for the French ASR system which only features FFNs. These AMs were trained with 0.9K (De), 5.6K (En), 3.9K (Es), and 0.7K (Fr) hours of speech data from multiple sources and domains.

On the other hand, language models (LM) consist of a linear interpolation of several n-gram LMs trained with SRILM (Stolcke 2002), combined with a recurrent NN (RNN) LM trained using the RNNLM toolkit (Mikolov 2011) (De, Fr), or an LSTM LM trained with the CUED-RNNLM toolkit (Chen et al. 2016) (Es, En). The vocabulary of LMs was restricted to 200K words. As training monolingual text data, we disposed of 0.8G (De), 300G (En), 0.7G (Es) and 1.8G (Fr) tokens.

Regarding ASR performance, these systems show 19.8 (De), 17.2 (En), 10.9 (Es) and 24.3 (Fr) Word Error Rate% (WER%) figures in their corresponding test sets of the Europarl-ST corpus.

MT systems

Neural MT systems were trained for each of the translation directions to be studied using the fairseq toolkit (Ott et al. 2019). The initial models are general out-of-domain systems trained with millions (M) of sentences: 21.0M for De↔En, 21.1M for En↔Es and 38.2M for En↔Fr. These models followed the sentence-level Transformer (Vaswani et al. 2017) BASE configuration, and were finetuned using the Europarl-ST training data.

Two MT systems were trained for each translation direction depending on the preprocessing scheme applied to the source sentences in the training set. The first scheme uses a conventional MT preprocessing (tokenization, truecasing, etc.), while the second scheme applies a special ST preprocessing to the source

Table 2.9: BLEU scores of the cascade ST on the Europarl-ST test sets depending on the preprocessing scheme.

Source prep. scheme	En-De	En-Es	En-Fr	Es-En	Fr-En	De-En
Conventional MT	22.4	28.0	23.4	26.5	25.4	21.3
Special ST	26.5	35.5	29.3	33.8	29.9	25.8

side of the training set, by lowercasing, transliterating and removing punctuation marks from all sentences (Matusov, Wilken, et al. 2018). This latter preprocessing scheme guarantees that the same conditions for the MT input are found at training and inference time. Since conventional MT preprocessing was applied to the target side, our hope is that the model is also able to learn to recover casing and punctuation information from the source to the target side. Both preprocessing schemes were evaluated by translating ASR hypotheses provided in chunks given by the baseline VAD segmenter. Results are shown in Table 2.9. As the segmentation is different from that of the reference, the evaluation is carried out by re-segmenting the translations so that they match the segmentation of the reference (Matusov, Leusch, et al. 2005).

As shown in Table 2.9, BLEU score improvements of the ST scheme over the MT scheme range from 4.1 (En-De) to 7.5 (En-Es), due to the fact that the ST source processing scheme fixes the mismatch between training and inference time. At the same time, MT systems are able to recover punctuation information that was not available in the ASR output. Thus, the special ST preprocessing scheme was applied in the rest of experiments.

Segmentation models

Depending on the segmentation model, text and optionally audio belonging to Europarl-ST were used as training data. As a preprocessing step, an end-of-chunk token was inserted in the text training data after each punctuation mark, such as full point, question/exclamation marks, etc., delimiting a chunk. In addition, the ST preprocessing scheme was applied to the annotated reference transcriptions in order to obtain training data that mimics ASR output. In the case of Audio models, as mentioned before, audio and reference transcriptions were forced-aligned using the AMs described in Section 2.4 in order to compute word and silence durations as acoustic features.

Due to the class imbalance present in the segmentation problem, (95% of samples belong to the non-split class), training batches were prepared by weighted random sampling so that on average, one third of the samples belongs to the

split class. Otherwise, the model degenerates to always classifying into the non-split class.

The Text model consists of 256-unit word-embedding layer, followed by a forward GRU-based RNN of 256 units. Second, for the Audio w/ RNN model, acoustic features are processed by a forward GRU-based RNN of 8 units. State vectors from Text, and optionally Audio w/ RNN, are fed into a two-layer FFN of 128 units and RELU activation. A dropout of 0.3 is applied after the RNN and FFN layers. Architecture decisions were taken on the basis of the BLEU results obtained on the dev set.

Given the sequential nature of the split/non-split decision process as a streaming ASR output is processed, greedy and beam search decoding algorithms were implemented and compared, but negligible differences were observed between them.

2.5 Evaluation

In order to perform an evaluation that simulates real conditions, the ASR hypothesis of an entire speech (intervention made by a MEP, with an average duration of 100 seconds) is fed to the segmentation model whose generated chunks are translated by the MT system. The chunks are translated independently from each other. The quality of the MT output, in terms of BLEU score, provides a clear indication of the performance of the streaming ST segmenter and allows us to compare different segmenters.

Figure 2.2 shows BLEU scores as a function of the length of the future window for the English-German (En-De) and Spanish-English (Es-En) dev sets. On the left-hand side, the three segmenters (Text, Audio w/ RNN and Audio w/o RNN) are compared averaging their BLEU scores over history sizes 5, 10 and 15 for the sake of clarity. On the right-hand side, the effect of history sizes is analysed for the Audio w/o RNN segmenter. In both cases, reference transcriptions were used as input to the segmenter.

As observed, the length of the future window is a very significant parameter to decide whether to split or not, which validates our decision to use a model that considers not only past history, but also a future window. In the case of En-De, adding a future window significantly improves the results, up to 5.8 and 3.5 BLEU points on average, in the Text and Audio models, respectively. Similarly for Es-En, but at a lower magnitude, a gain of up to 3.7 and 3.4 BLEU points on average in the Text and Audio models, respectively, is obtained at larger future windows.

When comparing the segmenters (Figure 2.2 on the left), the Text segmenter provides a performance that is clearly lower than the Audio-based segmenters for the English-German pair, and similar or lower for the Spanish-English pair. Audio-based segmenters offers nearly the same BLEU scores for English-German and Spanish-English. However, the Audio w/o RNN being a simpler model reaches slightly better BLEU scores using future window of length 4. This window length presents an appropriate trade-off between system latency and accuracy in our streaming scenario. Focusing on the Audio w/o RNN segmenter (Figure 2.2 on the right), longer history sizes such as 10 and 15 clearly provide better BLEU scores than the shorter history size ($h = 5$). A history size of 10 reaches the best BLEU scores for English-German, and similar performance is achieved between 10 and 15 in Spanish-English for future window of length 4. Based on these results, a history size of 10 and a future window of length 4 were selected for the rest of the experiments.

Table 2.10 presents BLEU scores of conventional cascade ST systems, in which the ASR output is segmented using the three proposed models and passed down to the MT system, from English into German, Spanish and French, and vice versa. As an upper-bound reference, results on an oracle segmenter are provided, which we have approximated by splitting the text into chunks using punctuation marks. The oracle segmenter shows which are the best BLEU scores that can be achieved with our current ASR and MT systems.

Table 2.10: BLEU scores on the test sets provided by the conventional cascade ST system with ASR output.

Segmenter	En-De	En-Es	En-Fr	Es-En	Fr-En	De-En
Baseline (VAD)	26.5	35.5	29.3	33.8	29.9	25.8
Text	27.6	37.0	29.4	34.7	31.6	28.1
Audio w/o RNN	28.4	37.2	30.0	34.4	32.1	28.3
Audio w/ RNN	28.4	37.3	30.1	33.9	32.1	28.2
Oracle	31.6	41.3	33.6	38.1	35.3	31.3

BLEU scores show how, except for Spanish-English, the models with acoustic features are able to outperform those that are only text-based. The largest improvement is in the English-German case, with a 0.8 BLEU-point improvement of Audio models over the Text model. When comparing the Audio models, there does not seem to be an improvement of using RNN to process the acoustic features with respect to directly feeding the acoustic features to the FFN. In the case of the Spanish-English system, as segmentation hyperparameters were optimised and shared across language directions, the segments generated from the Spanish-English Audio models turned out to be around 60% longer

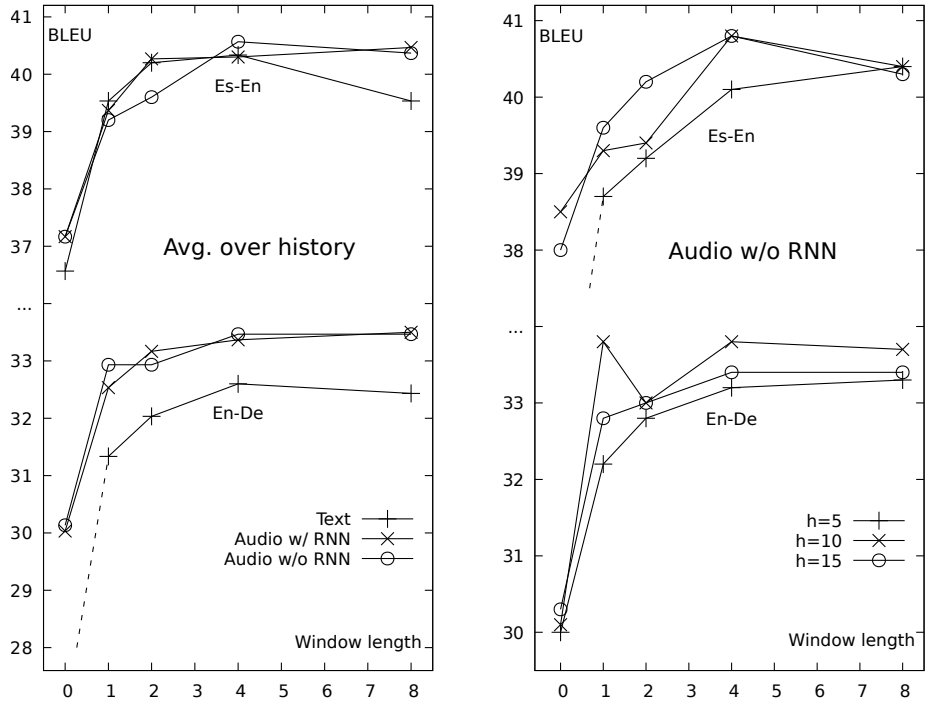

Figure 2.2: BLEU scores in the English-German (En-De) and Spanish-English (Es-En) dev sets as a function of future window length, averaged over history sizes for the three segmenters on the left-hand side, and on history sizes 5, 10 and 15 for the Audio w/o RNN segmenter on the right-hand side.

than those of other systems. This fact leads to a slight degradation of the performance of the Spanish-English sentence-based MT system for Audio models compared to the Text model.

Table 2.11 shows BLEU scores when the ASR output is replaced by the reference transcription, so that errors are only due to the segmenter and the MT systems. These results follow the trend of those in Table 2.10, with improvements of Audio over Text models, and no significant differences between both Audio models. Unlike in Table 2.10 when ASR output was considered, the Spanish-English Audio w/o RNN system does improve the results of the Text model when reference transcriptions are provided. Interestingly enough, in this case the oracle segmentation allows us to observe the performance degradation specifically due to the segmentation model without the interference of

Table 2.11: BLEU scores on the test sets provided by a cascade ST system with reference transcriptions.

Segmenter	En-De	En-Es	En-Fr	Es-En	Fr-En	De-En
Text	33.3	43.3	35.6	37.8	38.1	30.0
Audio w/o RNN	34.2	44.2	36.2	38.2	38.8	30.3
Audio w/ RNN	34.1	44.1	36.2	37.4	38.7	30.3
Oracle	37.2	47.4	38.9	41.3	41.5	35.6

the noisy ASR output, that is, between 2.7 and 5.3 BLEU points. Those oracle results show the best-case scenario that can be achieved with the current MT systems, using the reference transcriptions and the reference segmentation. As the addition of the RNN to process the acoustic features does not improve the performance, the simpler Audio w/o RNN will be used in the remaining experiments.

Comparison with previous work

In this section, we compare our results with previous work in the literature described in Section 2.2: the n-gram LM based segmenter included in the SRILM toolkit (Stolcke 2002), and the monolingual MT segmentation (E. Cho, Jan Niehues, and Waibel 2017) whose implementation is also publicly available[3].

Table 2.12 shows BLEU scores of a cascade ST system for the English-German Europarl-ST test set, comparing the two segmenters mentioned above, the Audio w/o RNN model proposed in this work, and the oracle segmenter that provides the reference segmentation. Except for the oracle, these segmenters were trained using only the Europarl-ST (EP-ST) training set, or the Europarl-ST training set plus additional training data from the IWSLT 2012 evaluation campaign (Cettolo, Girardi, and Federico 2012) , in order to study the performance of the segmenter when additional, out-of-domain text data is available. Results translating both, ASR hypotheses as well as reference transcriptions, are provided.

The results show the same trend across inputs to the MT system, ASR outputs, and reference transcriptions; but differences in BLEU over segmenters are more noticeable when segmenting the references. The LM based segmenter provides the lowest BLEU scores and is not able to take advantage of additional IWSLT training data. The monolingual MT model is at a middle ground between the

[3]https://github.com/jniehues-kit/SLT.KIT

Table 2.12: Comparison with previous work in terms of BLEU score on the English-German test set of the Europarl-ST corpus.

Segmenter	Train data	ASR	References
LM based	EP-ST	27.0	32.9
	+ IWSLT	26.5	31.7
Mono MT	EP-ST	28.0	33.8
	+ IWSLT	28.1	34.1
This work	EP-ST	28.4	34.2
	+ IWSLT	28.5	35.0
Oracle	–	31.6	37.2

LM based segmenter and our segmenter, but it is able to take advantage of the additional IWSLT training data. However, our segmenter outperforms all other segmenters in both training data settings. More precisely, when incorporating IWSLT training data, our segmenter outperforms by 0.4 BLEU (ASR output) and 0.9 BLEU (reference transcriptions) the best results of previous work obtained using the monolingual MT model, mainly thanks to the ability to use acoustic information. Additionally, our proposed model shrinks the gap with respect to the oracle segmentation to 3.1 BLEU points working with ASR output, and 2.2 BLEU points when reference transcriptions are provided.

Latency evaluation

We will now measure the latency of our cascade ST system in a streaming scenario. Following (Li et al. 2020), we define accumulative chunk-level latencies at three points in the system, as the time elapsed between the last word of a chunk being spoken, and: 1) The moment the consolidated hypothesis for that chunk is provided by the ASR system; 2) The moment the segmenter defines that chunk on the ASR consolidated hypothesis; 3) The moment the MT system translates the chunk defined by the segmenter. These three latency figures, in terms of mean and standard deviation, are shown in Table 2.25. It should be noticed that this ST system is working with ASR consolidated hypotheses in the sense that these hypotheses will not change as the audio stream is further processed.

The difference of 1.1 seconds between the ASR and the segmenter is mostly due to the need to wait for the words in the future window to be consolidated, as the time taken by the segmenter to decide whether to split or not is negligible ($\simeq$ 0.01s). Lastly, the MT system has a delay of 0.5 seconds. The total latency

Table 2.13: Accumulative chunk-level latencies in seconds (mean ± std. dev.) for the ASR, Segmenter and MT components of the Es-En ST cascade model.

	Latency (seconds)
ASR	4.1 ± 1.6
+ Seg.	5.2 ± 2.2
+ MT	5.7 ± 2.2

is dominated by the ASR system, since the long-range dependencies of the RNN-based LM delay the consolidation of the hypothesis, which is needed by the segmenter and the MT system in order to output the definitive translation.

In practice, however, the ST system could work with non-consolidated hypotheses, since these hypotheses very rarely change with respect to those consolidated. In this case, the latency of the ASR system is significantly reduced to 0.8 ± 0.2 seconds, while the latency experienced by the user for the whole ST system is 1.3 ± 0.4 seconds, as the segmenter does not wait for the words in the future window to be consolidated.

2.6 Conclusions

This work introduces a statistical framework for the problem of ASR output segmentation in streaming ST, as well as three possible models to instantiate this framework. In contrast to previous works, these models not only consider text, but also acoustic information. The experimental results reported provide two key insights. Firstly, we have confirmed how the preprocessing of the MT training data has a significant effect for ST, and how a special preprocessing that is closer to the inference conditions is able to obtain significant improvements. Secondly, we have shown the importance of including acoustic information in the segmentation process, as the inclusion of these features improves system performance. The proposed model improves the results of previous works on the Europarl-ST test set when evaluated with two training data setups.

In terms of future work, there are many ways of improving the direct segmentation model that has been presented here. We plan to look into additional acoustic features as well as possible ways to incorporate ASR information into the segmentation process. In addition, the segmentation model itself could also benefit from the incorporation of additional text data as well as pre-training procedures. We also devise two supplementary research lines, the integration of the segmentation into the translation process, so the system learns how to

segment and translate at the same time, and moving from an offline MT system to a streaming MT system to improve response time, but without performance degradation.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 761758 (X5Gon); the Spanish Government's research project Multisub, ref. RTI2018-094879-B-I00 (MCIU/AEI/FEDER,EU) and FPU scholarship FPU18/04135; the Generalitat Valenciana's research project Classroom Activity Recognition, ref. PROMETEO/2019/111., and predoctoral research scholarship ACIF/2017/055. The authors also wish to thank the anonymous reviewers for their criticisms and suggestions.

2.7 Reproducibility

The source code of the Direct Segmentation Model, as well as the ASR hypothesis and acoustic features used in the experiments are attached as supplementary materials. Combined with the instructions provided for training the MT systems, this allows for faithful reproduction of our experiments.

2.8 ASR Systems

The acoustic models were trained using the datasets listed on Table 2.14, and the architecture of the models is summarized in Table 2.15.

The language models were trained using the datasets listed on Table 2.16. The number of English words includes 294G words from Google Books counts. As for the models themselves, they are an interpolation between 4-gram LM and a RNNLM. For German and French, the RNN is trained with the RNNLM toolkit and has a hidden layer of 400 units. For Spanish and English, the RNN is a LSTM trained with the CUED toolkit, with an embbeding layer of 256 units and a hidden layer of 2048 units. The vocabulary was limited to the most common 200K words.

Table 2.14: Statistics of the speech resources used for acoustic model training.

English		Spanish	
Corpus	Hours	Corpus	Hours
Crawled Data	3313	Crawled Data	3466
LibriSpeech	960	PM	261
TED-LIUM v3	454	EPPS	157
CommonVoice	243	Voxforge	21
SWC	154		
VL.NET	110		
Voxforge	109		
AMI	96		
EPPS	79		
ELFA	48		
VCTK	44		

German		French	
Corpus	Hours	Corpus	Hours
Crawled Data	716	Crawled Data	592
GSC-TUDa	158	TEDx	39
Audiobooksfr	28		
Voxforge	21		

Table 2.15: Details of the acoustic models architecture.

	English	Spanish
MFCC	80	85
Input size	80	85
Standard Model (1-pass)	8x1024(BLSTM)	8x1024(BLSTM)
Output states (1-pass)	16132	10041
fCMLLR model (2-pass)	–	–
Output states (2-pass)	–	–
	German	French
MFCC	48	48
Input size	48x11	48x11
Standard Model (1-pass)	6x2048(DNN)	6x2048(DNN)
Output states (1-pass)	18867	6282
fCMLLR model (2-pass)	5x1024(BLSTM)	6x2048(DNN)
Output states (2-pass)	18867	6651

2.9 MT Systems

The models were trained using the datasets listed on Table 2.17.

The following fairseq command was used to train the systems:

```
fairseq-train $CORPUS_FOLDER \
 -s $SOURCE_LANG_SUFFIX \
 -t $TARGET_LANG_SUFFIX \
 --arch transformer \
 --share-all-embeddings \
 --optimizer adam \
 --adam-betas '(0.9, 0.98)' \
 --clip-norm 0.0 \
 --lr-scheduler inverse_sqrt \
 --warmup-init-lr 1e-07 \
 --warmup-updates 4000 \
 --lr 0.0005 \
 --min-lr 1e-09 \
 --dropout 0.3 \
 --weight-decay 0.0 \
 --criterion \
```

Table 2.16: Statistics of text resources used for language modelling.

English		French	
Corpus	MWords	Corpus	MWords
News-Discuss	3650	OpenSubtitles	1146
Wikipedia	2266	Ufal	910
News Crawl	1120	Wikipedia	586
LibriSpeech	804	United Nations	343
GIGA	617	News Crawl	298
United Nations	334	Crawled data	116
HAL	92	Comm. Crawl	41
Europarl	54		
DGT-TM	45		
News comm.	6		
WIT-3	3		
COSMAT	1		
EuroParl TV	1		

German		French	
Corpus	MWords	Corpus	MWords
Wikipedia	642	Giga	665
Europarl	46	Wikipedia	375
Comm. Crawl	45	UN	358
News-Crawl	30	OpenSubs	263
Reuters	38	DGT	79
Tatoeba	3	Europarl	55
		COSMAT	29
		TT2	13
		News comm.	5
		TED	4
		AMARA fr	1
		EUTV	1

Table 2.17: Satistics of the text resources used for training MT systems.

Corpus	Samples(M)		
	De-En	Fr-En	Es-En
DGT	5.1	–	–
EUbookshop	9.3	–	5.2
TildeMODEL	4.2	–	–
Wikipedia	2.4	–	1.8
UN	–	11.0	–
GIGA	–	22.5	–
newscommentary	–	1.0	–
commoncrawl	–	3.2	1.8
EU-TT2	–	–	1.0

```
 label_smoothed_cross_entropy \
--label-smoothing 0.1 \
--max-tokens 4000 \
--update-freq 8 \
--save-dir $OUTPUT_FOLDER \
--no-progress-bar \
--log-interval 100 \
--save-interval-updates 10000 \
--keep-interval-updates 20 \
--ddp-backend=no_c10d \
--fp16
```

For finetuning, we change the following:

```
--optimizer sgd \
--lr-scheduler fixed \
--lr 5e-5 \
```

2.10 Segmentation Systems

The different hyperparameters values that were tried for the segmentation models are shown on Table 2.18. In total, no more than 75 combinations were tested in order to conduct the experiments reported on this paper.

Table 2.18: Segmentation model hyperparameter exploration. Selected values are shown in bold.

Hyperparameter	Values
Embedding size	128,**256**,512,1024
RNN size	128,**256**,512,1024
FF layers	1,**2**,3
FF size	128,**256**,512
Batch size	128,**256**,512
Learning rate	0.001,**0.0001**
Optimizer	**Adam**
Dropout	**0.3**,0.5
History size	0,1,2,5,**10**,15,20
Future window	0,1,2,**4**,8

References

Abadi, Martin et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org (cit. on p. 60).

Agua, Miguel A. del et al. (2014). "The Translectures-UPV Toolkit". In: *Advances in Speech and Language Technologies for Iberian Languages*. Ed. by Juan Luis Navarro Mesa et al. Springer International Publishing, pp. 269–278. ISBN: 978-3-319-13623-3. DOI: 10.1007/978-3-319-13623-3_28 (cit. on p. 60).

Arivazhagan, Naveen et al. (2019). "Monotonic Infinite Lookback Attention for Simultaneous Machine Translation". In: *Proc. of ACL*. ACL, pp. 1313–1323. DOI: 10.18653/v1/P19-1126 (cit. on p. 52).

Bahar, Parnia, Tobias Bieschke, and Hermann Ney (Dec. 2019). "A comparative study on end-to-end speech to text translation". In: *Proc. of IEEE ASRU*, pp. 792–799. DOI: 10.1109/ASRU46091.2019.9003774 (cit. on p. 52).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*. Ed. by Yoshua Bengio and Yann LeCun (cit. on p. 52).

Berard, Alexandre et al. (2018). "End-to-End Automatic Speech Translation of Audiobooks". In: *Proc. of ICASSP*. IEEE, pp. 6224–6228. DOI: 10.1109/ICASSP.2018.8461690 (cit. on p. 52).

Cettolo, Mauro and Marcello Federico (2006). "Text Segmentation Criteria for Statistical Machine Translation". In: *In Proc. of Advances in Natural Language Processing, FinTAL*. Ed. by Tapio Salakoski et al. Vol. 4139. Lecture

Notes in Computer Science. Springer, pp. 664–673. DOI: 10.1007/11816508_66 (cit. on p. 53).

Cettolo, Mauro, Christian Girardi, and Marcello Federico (2012). "WIT^3: Web Inventory of Transcribed and Translated Talks". In: *Proc. of EAMT*, pp. 261–268 (cit. on p. 65).

Chan, William et al. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. of ICASSP*. IEEE, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621 (cit. on p. 51).

Chen, Xi et al. (2016). "CUED-RNNLM — An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *ICASSP 2016* (cit. on p. 60).

Cho, Eunah, Jan Niehues, Kevin Kilgour, et al. (2015). "Punctuation insertion for real-time spoken language translation". In: *Proc. of IWSLT*. ISCA (cit. on pp. 53, 55).

Cho, Eunah, Jan Niehues, and Alex Waibel (2012). "Segmentation and punctuation prediction in speech language translation using a monolingual translation system". In: *Proc. of IWSLT*. ISCA (cit. on pp. 53, 55).

– (2017). "NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation". In: *Proc. of Interspeech*. ISCA, pp. 2645–2649. DOI: 10.21437/Interspeech.2017-1320 (cit. on pp. 53, 55, 65).

Fügen, Christian, Alex Waibel, and Muntsin Kolss (2007). "Simultaneous translation of lectures and speeches". In: *Machine Translation* 21.4, pp. 209–252. DOI: 10.1007/s10590-008-9047-0 (cit. on p. 53).

Gangi, Mattia Antonino Di et al. (2019). "Enhancing Transformer for End-to-end Speech-to-Text Translation". In: *Proc. of MT Summit XVII Volume 1: Research Track*. EAMT, pp. 21–31 (cit. on p. 52).

Iranzo-Sánchez, Javier et al. (2020). "Europarl-ST: A Multilingual Corpus For Speech Translation Of Parliamentary Debates". In: *Proc. of ICASSP*. IEEE, pp. 8229–8233 (cit. on p. 59).

Irie, Kazuki et al. (2019). "Language Modeling with Deep Transformers". In: *Proc. Interspeech 2019*. ISCA, pp. 3905–3909. DOI: 10.21437/Interspeech.2019-2225 (cit. on p. 51).

Jia, Ye et al. (2019). "Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model". In: *Proc. of Interspeech*. Ed. by Gernot Kubin and Zdravko Kacic. ISCA, pp. 1123–1127. DOI: 10.21437/Interspeech.2019-1951 (cit. on p. 52).

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*, pp. 3820–3824. published (cit. on p. 52).

Jorge, Javier, Adrià Giménez, Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, et al. (Jan. 1, 2020). "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP*. IEEE (cit. on p. 52).

Kolss, Muntsin, Stephan Vogel, and Alex Waibel (2008). "Stream decoding for simultaneous spoken language translation". In: *Proc. of Interspeech*. ISCA, pp. 2735–2738 (cit. on p. 53).

Li, B. et al. (2020). "Towards Fast and Accurate Streaming End-To-End ASR". In: *Proc. of ICASSP*. IEEE. DOI: 10.1109/ICASSP40776.2020.9054715 (cit. on p. 66).

Ma, Mingbo et al. (2019). "STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework". In: *Proc. of ACL*. ACL, pp. 3025–3036. DOI: 10.18653/v1/P19-1289 (cit. on p. 52).

Matusov, Evgeny, Gregor Leusch, et al. (2005). "Evaluating machine translation output with automatic sentence segmentation". In: *Proc. of IWSLT*. ISCA (cit. on p. 61).

Matusov, Evgeny and Hermann Ney (2011). "Lattice-Based ASR-MT Interface for Speech Translation". In: *IEEE Trans. Audio, Speech & Language Processing* 19.4, pp. 721–732. DOI: 10.1109/TASL.2010.2060483 (cit. on p. 52).

Matusov, Evgeny, Patrick Wilken, et al. (2018). "Neural speech translation at AppTek". In: *Proc. of IWSLT*. ISCA, pp. 104–111 (cit. on p. 61).

Mikolov, T. (2011). *The RNNLM Toolkit*. http://www.fit.vutbr.cz/~imikolov/rnnlm/ (cit. on p. 60).

Ng, Raymond W. M. et al. (2016). "Groupwise learning for ASR k-best list reranking in spoken language translation". In: *Proc. of ICASSP*, pp. 6120–6124 (cit. on p. 52).

Niehues, J. et al. (2019). "The IWSLT 2019 Evaluation Campaign". In: *Proc. of IWSLT*. ISCA. DOI: 10.5281/zenodo.3525578 (cit. on p. 52).

Oda, Yusuke et al. (2014). "Optimizing Segmentation Strategies for Simultaneous Speech Translation". In: *Proc. of ACL*. ACL, pp. 551–556. DOI: 10.3115/v1/p14-2090 (cit. on p. 53).

Ott, Myle et al. (2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proc. of NAACL-HLT: Demonstrations*. ACL (cit. on p. 60).

Papineni, Kishore et al. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proc. of ACL*. ACL, pp. 311–318. DOI: 10.3115/1073083.1073135 (cit. on p. 59).

Park, Daniel S. et al. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. Interspeech*, pp. 2613–2617 (cit. on p. 52).

Peitz, Stephan et al. (2012). "Spoken language translation using automatically transcribed text in training". In: *Proc. of IWSLT*. ISCA, pp. 276–283 (cit. on p. 52).

Pino, Juan et al. (2019). "Harnessing Indirect Training Data for End-to-End Automatic Speech Translation: Tricks of the Trade". In: *Proc. of IWSLT*. ISCA. DOI: 10.5281/zenodo.3525032 (cit. on p. 52).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016a). "Improving Neural Machine Translation Models with Monolingual Data". In: *Proc. of ACL*. ACL, pp. 86–96. DOI: 10.18653/v1/p16-1009 (cit. on p. 52).

– (2016b). "Neural Machine Translation of Rare Words with Subword Units". In: *Proc. of ACL*. ACL, pp. 1715–1725. DOI: 10.18653/v1/p16-1162 (cit. on p. 52).

Siahbani, Maryam et al. (2018). "Simultaneous Translation using Optimized Segmentation". In: *Proc. of AMTA*. Ed. by Colin Cherry and Graham Neubig. AMTA, pp. 154–167 (cit. on p. 53).

Silvestre-Cerdà, Joan Albert et al. (Nov. 22, 2012). "Albayzin Evaluation: The PRHLT-UPV Audio Segmentation System". In: *Proc. of IberSPEECH 2012*, pp. 596–600. published (cit. on p. 60).

Sperber, Matthias, Graham Neubig, et al. (July 2019). "Self-Attentional Models for Lattice Inputs". In: *Proc. of ACL*. ACL, pp. 1185–1197. DOI: 10.18653/v1/P19-1115 (cit. on p. 52).

Sperber, Matthias, Jan Niehues, and Alex Waibel (2017). "Toward robust neural machine translation for noisy input sequences". In: *IWSLT 2017* (cit. on p. 52).

Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech*. Ed. by John H. L. Hansen and Bryan L. Pellom. ISCA, pp. 901–904 (cit. on pp. 60, 65).

Stolcke, Andreas and Elizabeth Shriberg (1996). "Automatic linguistic segmentation of conversational speech". In: *Proc. of ICSLP*. Vol. 2. ISCA, pp. 1005–1008 (cit. on pp. 53, 55).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS*. Ed. by Isabelle Guyon et al., pp. 5998–6008 (cit. on pp. 52, 60).

Waibel, Alex et al. (1991). "JANUS: a speech-to-speech translation system using connectionist and symbolic processing strategies". In: *Proc. of ICASSP*. IEEE, pp. 793–796. DOI: 10.1109/ICASSP.1991.150456 (cit. on p. 51).

Wang, Xiaolin, Andrew Finch, et al. (2016). "An Efficient and Effective Online Sentence Segmenter for Simultaneous Interpretation". In: *Proc. of WAT*. The COLING 2016 Organizing Committee, pp. 139–148 (cit. on pp. 53, 55).

Wang, Xiaolin, Masao Utiyama, and Eiichiro Sumita (2019). "Online Sentence Segmentation for Simultaneous Interpretation using Multi-Shifted Recurrent

Neural Network". In: *Proc. of MT Summit XVII Volume 1: Research Track*. EAMT, pp. 1–11 (cit. on pp. 53, 55).

Weiss, Ron J. et al. (2017). "Sequence-to-Sequence Models Can Directly Translate Foreign Speech". In: *Proc. of Interspeech*. ISCA, pp. 2625–2629. DOI: 10.21437/Interspeech.2017-503 (cit. on p. 52).

Zeyer, Albert, Patrick Doetsch, et al. (2017). "A comprehensive study of deep bidirectional LSTM RNNS for acoustic modeling in speech recognition". In: *Proc. of ICASSP*. IEEE, pp. 2462–2466. DOI: 10.1109/ICASSP.2017.7952599 (cit. on p. 60).

Zeyer, Albert, Ralf Schlüter, and Hermann Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Proc. of Interspeech 2016*. Ed. by Nelson Morgan. ISCA, pp. 3424–3428. DOI: 10.21437/Interspeech.2016-759 (cit. on p. 52).

Zheng, Baigong et al. (2019). "Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation". In: *Proc. of EMNLP-IJCNLP*. ACL, pp. 1349–1354. DOI: 10.18653/v1/D19-1137 (cit. on p. 52).

3 Streaming cascade-based speech translation leveraged by a direct segmentation model

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

Neural Networks, 142 , pp. 303–315, 2021

10.1016/j.neunet.2021.05.013

Streaming cascade-based speech translation leveraged by a direct segmentation model

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

Abstract

The cascade approach to Speech Translation (ST) is based on a pipeline that concatenates an Automatic Speech Recognition (ASR) system followed by a Machine Translation (MT) system. Nowadays, state-of-the-art ST systems are populated with deep neural networks that are conceived to work in an offline setup in which the audio input to be translated is fully available in advance. However, a streaming setup defines a completely different picture, in which an unbounded audio input gradually becomes available and at the same time the translation needs to be generated under real-time constraints. In this work, we present a state-of-the-art streaming ST system in which neural-based models integrated in the ASR and MT components are carefully adapted in terms of their training and decoding procedures in order to run under a streaming setup. In addition, a direct segmentation model that adapts the continuous ASR output to the capacity of simultaneous MT systems trained at the sentence level is introduced to guarantee low latency while preserving the translation quality of the complete ST system. The resulting ST system is thoroughly evaluated on the real-life streaming Europarl-ST benchmark to gauge the trade-off between quality and latency for each component individually as well as for the complete ST system.

Keywords: automatic speech recognition, speech translation, machine translation, segmentation model, streaming, cascade system, hybrid system, deep neural networks

3.1 Introduction

Deep Neural Networks (DNNs) are revolutionizing not only speech-related research fields, such as purely Automatic Speech Recognition (ASR) with significant breakthroughs (Chan et al. 2016; Irie et al. 2019; Park et al. 2019; Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020), but also other closely connected fields, such as Machine Translation (MT) (Bahdanau, K. Cho, and Bengio 2015; Sennrich, Haddow, and Birch 2016b; Sennrich, Haddow, and Birch 2016a; Vaswani et al. 2017) and consequently, Speech Translation (ST). Indeed, ST is gaining momentum due to the vast number of industry applications that could be exploited based on this technology, from person-to-person communication to subtitling of audiovisual content, just to mention the main two applications.

Nowadays, two approaches to ST, end-to-end and cascade, coexist. End-to-end models directly perform a mapping from speech in a source language into a text representation in a target language, without exploiting an intermediate discrete representation and jointly training model parameters (Weiss et al. 2017; Berard et al. 2018; Gangi et al. 2019; Jia et al. 2019). However, current end-to-end models are not usually well-suited for a streaming setup, since they need to process the entire input sequence before providing the corresponding translation.

Differently, the cascade approach considers a two-step process in which an ASR system transcribes the source language in speech form, and the automatically generated transcription is pipelined into an MT system that provides the target language in text form. There are well-known pros and cons of this approach with respect to that of end-to-end. On the one hand, it is possible to train strong independent ASR and MT systems in the cascade approach, since abundant manually transcribed audio and parallel data are widely available in high-resourced languages. In contrast, this is not usually the case in end-to-end models, since manually transcribed audio data in a source language aligned with text data in the desired target language is more expensive to produce and cannot be found in the same magnitude. Thus, end-to-end models resort to pre-training and data augmentation techniques to alleviate this problem (Bahar, Bieschke, and Hermann Ney 2019; Pino et al. 2019). On the other hand, the cascade approach tends to propagate ASR errors into the MT system, while training a single end-to-end model is theoretically more robust than two decoupled models, if sufficient data would be available. All in all, cascade systems still outperform end-to-end systems in standard setups (Pino et al. 2019; J. Niehues et al. 2019; Bahar, Wilken, et al. 2020).

Streaming cascade-based ST systems just started to become available (Bahar, Wilken, et al. 2020) thanks to recent advances in streaming hybrid and end-to-end ASR systems stating competitive results compared to offline systems (Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020; Albert Zeyer, Ralf Schlüter, and Hermann Ney 2016; Jorge, Giménez, Iranzo-Sánchez, Civera, et al. 2019; Albert Zeyer, Bahar, et al. 2019; Miao et al. 2020; Moritz, Hori, and Le 2020; Zhang, Lu, et al. 2020), and also due to the significant progress in simultaneous MT (Jan Niehues et al. 2018; Naveen Arivazhagan et al. 2019; M. Ma et al. 2019; Zheng et al. 2019; N. Arivazhagan et al. 2020). However, the segmentation of the ASR output is still essential to deal with the simultaneous translation of long audio streams (Fügen, Waibel, and Kolss 2007; Rangarajan Sridhar et al. 2013; Oda et al. 2014; E. Cho, Jan Niehues, Kilgour, et al. 2015; Gu et al. 2017; Iranzo-Sánchez, Giménez, et al. 2020). Indeed, state-of-the-art simultaneous MT models are Transformer-based models trained on sentence pairs with a limited length. These vanilla Transformer models cannot capture any longer-term dependency beyond the predefined sentence length observed in training (Popel and Bojar 2018; Dai et al. 2019). Thus, the translation accuracy of these simultaneous Transformed-based MT models rapidly degrades as the source sentence length goes beyond that observed in training. This fact leads to the need of a text segmenter that splits the ASR output into hopefully semantically self-contained chunks[4] that can be successfully translated by a simultaneous MT system.

In this work we present a state-of-the-art streaming cascade-based ST system evaluated on the Europarl-ST task (Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020), a real streaming ST benchmark including parliamentary speeches of up to 10-minute long. This benchmark along with the limitation of simultaneous Transformed-based MT models to adequately translate an unbounded sequence of words, motivates the crucial role played by the segmenter when translating continuous text streams provided by an ASR system under real-time constrains. For this reason, this work, in addition to review the ASR and MT components of the ST system developed in previous work, puts special emphasis on the description of our neural-based Direct Segmentation (DS) model followed by a thoroughly evaluation of its impact on a streaming cascade-based ST system in terms of accuracy and latency.

In contrast to our previous work in which the DS model was initially presented (Iranzo-Sánchez, Giménez, et al. 2020), here off-line MT systems are replaced by simultaneous MT systems. This fact has very important implications in this work. First, words provided by the ASR system are translated as

[4] A chunk must be understood as a sequence of words.

they become available without waiting for the end-of-chunk token to appear in the input of the MT system. This is a significant difference with our previous work mentioned above in which translations were only generated once a complete chunk was available, which is the expected chunk-level behavior of off-line MT systems. Second, simultaneous MT systems work at the word level, this allows to compute word-level latencies in contrast to the chunk-level latencies reported in our previous work. Finally, word-level latencies define a more realistic and challenging evaluation closer to the user experience and thus, specific experimental conditions for this work. As opposed to our previous work in which the ST system was only optimised for translation accuracy, in this work a joint optimization of the DS and translation models is carefully performed and reported not only for translation accuracy but also for word-level latency.

This paper is organized as follows. The neural-based stream-adapted ASR and MT components of the ST system are reviewed in Sections 3.2 and 3.3, respectively. Next, Section 3.4 describes in full detail the DS model seamlessly integrated between the ASR and MT components to allow streaming ST decoding. Then, in Section 3.5, ASR and MT components are individually assessed on the Europarl-ST task before going into an extensive evaluation in terms of accuracy and latency of the ST system when the DS model is integrated into the pipeline. Finally, conclusions are drawn and future work is foreseen in Section 4.

3.2 Streaming automatic speech recognition

Nowadays, state-of-the-art hybrid ASR systems use DNNs to approximate acoustic and language model probabilities. However, when we move from the offline to the streaming (online) setup, it is necessary to take into account a series of constraints imposed by the streaming scenario to efficiently manage DNNs. First, the acoustic information comes gradually over time, so the input sequence $\mathbf{x}_1^T$ is not fully available at decoding time. Second, output must be provided under tight real-time constraints, so efficient inference procedures must be devised to guarantee system usability.

More precisely, the limited access to the input poses some challenges to transcribe an audio stream, since the system needs to wait for enough acoustic information in the input to be available to perform the next decoding step. This fact introduces a trade-off between the quality of the acoustic scores provided by the DNN and the latency of the decoding step that directly impacts the response-time of the system. As acoustic models (AMs) based on DNNs work at sequence level, we should adapt their behavior to include these constraints.

On the other hand, most of the LMs can naturally work on a streaming setup, since model dependencies involve conditioning on previous words (history) to provide the posterior probability of the next one. In this case, the challenge resides on the inference speed when it comes to state-of-the-art neural-based LMs. In the following sections we will review how these neural-based models are adapted to perform streaming ASR.

Acoustic model

Over the last decade, deep Feed-Forward Networks (FFNs) (G. 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 contributed to improve acoustic modeling with respect to the historical approach based on Gaussian Mixture Model (GMM) (Bourlard and Wellekens 1990; Russell and Moore 1985). Indeed, RNNs based on the Long-Short Term Memory (LSTM) unit (Hochreiter and Schmidhuber 1997) have been successfully applied in ASR (Graves, Jaitly, and Mohamed 2013; Albert Zeyer, Doetsch, et al. 2017). More precisely, the so-called Bidirectional LSTM (BLSTM) architecture has been widely applied and studied for AM in ASR (Albert Zeyer, Doetsch, et al. 2017).

As expected in an offline setup, the BLSTM architecture observes the complete acoustic sequence to estimate the score for each frame in this sequence. However, this is not feasible in a streaming scenario under tight real-time constraints, as we need to minimize the time elapsed between the speaker utterance, and the corresponding transcription of that utterance by the system. For this reason, following the study performed in (A. Zeyer, R. Schlüter, and H. Ney 2016), we introduce the concept of *lookahead context* of a given frame, as the sequence of frames following this given frame that need to be processed to compute the acoustic score (Jorge, Adria Giménez, et al. 2020). The length of the lookahead context $n_{\text{lookahead}}$ allows us to control the trade-off between accuracy and latency, adjusting it to a minimum length that allows the BLSTM network to gather enough acoustic information. In practice, a sliding window with a limited lookahead context is moved over the infinite sequence of frames one frame at a time. The acoustic score of a given frame is a weighted average of the posterior probabilities of that frame computed over overlapping windows.

There is a final consideration that is missing in this adaptation of the BLSTM architecture to the streaming setup, that is, the normalization of acoustic features. Acoustic features are mean normalized over the full sequence, but again

this is not possible in a streaming setup. We alleviate this problem by introducing a configurable delay when starting to process an audio stream in order to gather enough acoustic evidence to compute the required statistics to normalize input features (Jorge, Adria Giménez, et al. 2020).

Language model

Similarly to AMs, LMs have significantly evolved from the count-based n-gram model (Shannon 1948; S. F. Chen and Goodman 1999) to continuous neural LMs based on LSTM RNN (Bengio et al. 2003; Schwenk 2007) and the Transformer architecture (Vaswani et al. 2017) with impressive results in ASR (Irie et al. 2019). However, the integration of neural-based LMs into a streaming ASR decoder requires an adaptation of their training procedure (Baquero-Arnal et al. 2020). First, the full computation of the softmax function cannot be afforded under real-time constraints. Instead, a variance regularization (VR) term is added during training, so that the sum of the softmax deviates minimally from a constant value (Shi et al. 2014). This constant value is assumed to be invariant during inference, significantly reducing the high computational cost of the softmax function. Second, a key idea applicable specifically to Transformer LMs is to limit the size of the word history. Though Transformer LMs are robust dealing with long-range dependencies due to their ability to attend to all previous words in a direct manner, this implies that every time a next word is predicted the whole word history needs to be processed. For the streaming case, we limit the size of the history to n words, avoiding an unbounded growth of memory requirements for the internal state.

Decoding

In the hybrid approach, decoding is performed with the conventional beam-search Viterbi algorithm (Viterbi 1967; Hermann Ney 1984). This algorithm combines scores provided by the AM, the LM and the pronunciation dictionary or lexicon, in order to find the most likely sequence of spoken words. Due to this combination of external models, the decoding process becomes much more complex than in end-to-end models. Nowadays, there are two predominant approaches to decoding, 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). Both approaches convert the AM and LM into data structures to perform a time-synchronous search, to which several pruning

techniques will be applied in order to reduce the exponentially growing search space.

These decoders leverage the discrete nature of count-based LMs to create the skeleton of the aforementioned search space before decoding. However, with the advent of neural-based LMs, it became unclear how to integrate these continuous models into the discrete search space. For this reason, multi-pass decoders benefited from neural-based LMs performing an additional rescoring step, in which n-best hypotheses stored in a word-graph structure were re-ranked to obtain significant accuracy improvements (Sundermeyer et al. 2014; Xie Chen et al. 2017; Xu et al. 2018).

Obviously, this additional rescoring step increases the response time of ASR systems compared to those based on one-pass decoders. In addition, search errors propagated in previous passes cannot be fixed in the rescoring step. One-pass decoders integrating neural-based LM have been proposed, such as those in (Arısoy et al. 2014; Singh, Oualil, and Klakow 2017; K. Lee et al. 2018), but few of them have reached the technology readiness level for a production environment under streaming conditions. In (Jorge, Adrià Giménez, et al. 2019), we proposed a novel one-pass decoder that allows a seamless integration of neural-based LMs, while keeping the system fast enough to perform real-time streaming decoding.

Our decoder follows a similar structure to the HCS decoder proposed in (Nolden 2017), where hypotheses are organized according to the LM history. To do that, we precompute a lookahead finite state model from a heavily pruned n-gram model. This model provides the main static search structure at word and triphoneme levels on which the decoding is based. Unlike other WFST-based decoders, no finite-state reduction algorithm is applied. Moreover, Hidden Markov Model (HMM) states are dynamically expanded on-demand during decoding to reduce memory consumption.

Apart from conventional beam search decoding parameters, such as beam width or the maximum number of active hypothesis, we enriched our decoder with two additional LM-related decoding parameters to specifically deal with neural-based LMs: the Language Model History Recombination (LMHR) and the Language Model Histogram Pruning (LMHP).

Regarding LMHR, it is important to remark that, unlike count-based LMs, neural-based LMs benefit from the unlimited context condensed in their internal state, a continuous vector representation, that potentially contains the previous context observed so far. Not including any limitation on the previ-

ous context leads to the generation of similar hypotheses that only differ in words far from the current time step. This fact limits the effectiveness of beam search, reducing hypothesis diversity and exploration. The LMHR parameter sets the length of the LM context, in terms of words, that is considered before performing hypothesis recombination. For example, setting LMHR to 3 means that hypotheses whose context is longer than three words are recombined. LMHR differs from the conventional recombination induced by WFSTs, as in that case recombination is limited to the precomputed transducer resulted from combining the HCLG model (Mohri and Riley 1999). Indeed, LMHR values can easily go beyond the usual 4 or 5-grams commonly used to compute WFSTs. In other words, the LMHR parameter enforces a structural limitation to the previous LM context, while the internal continuous state of neural-based LMs is preserved.

On the other hand, inference in neural-based LMs is computationally demanding. In this sense, the LMHP parameter provides an additional mechanism to control the number of active hypotheses at word level, that is, the number of hypotheses that will be expanded when a word-end node is reached. Thus, LMHP limits the number of queries (inferences) performed by the neural-based LM speeding up the decoding process.

3.3 Simultaneous machine translation

Current state-of-the-art MT systems (Barrault et al. 2020) are based on the Transformer architecture. However, this architecture was originally envisioned to entirely process a sentence before generating the corresponding translation, and thus it is not well-suited for a streaming scenario under real-time constraints. Recently, some variants of attention-based architectures that are able to carry out simultaneous translation have been presented (Naveen Arivazhagan et al. 2019; M. Ma et al. 2019; Raffel et al. 2017; X. Ma et al. 2020; Elbayad, Besacier, and Verbeek 2020). Basically, these variants limit the attention mechanism to those input words available in the stream, since the complete sentence cannot be observed. However, we focus on two Transformer-based variants: the Monotonic Multi-Head Attention (MMA) framework (X. Ma et al. 2020) and the efficient multi-path wait-k approach (Elbayad, Besacier, and Verbeek 2020), since both have achieved competitive results on reference tasks.

On the one hand, two attention mechanisms are discussed in (X. Ma et al. 2020), the Hard MMA (MMA-H) that attends only a single position in the input (Raffel et al. 2017), and the Infinite Lookback MMA (MMA-IL) that keeps

track of all previous positions in the input (Naveen Arivazhagan et al. 2019). However, MMA-IL attention mechanism is very computationally demanding and streaming response-time requirements cannot be met. Therefore, we resort to the MMA-H in this work. In MMA-H, a hyperparameter λ must be adjusted in order to control the balance between system latency and quality. Higher values of λ will bias the multiple heads in the attention mechanism towards reading synchrony, increasing translation speed but degrading accuracy.

On the other hand, wait-k models read k words from the input sentence before alternating write/read operations of one word at a time, being the write operation the generation of a target word (M. Ma et al. 2019). Wait-k models have proved to obtain better performance when trained for the specific k value employed in decoding (Zheng et al. 2019). This is exactly what is tackled by (Elbayad, Besacier, and Verbeek 2020) when proposing their efficient multi-path wait-k models. These models are trained across multiple values of k, allowing to perform the decoding with different latency constraints.

The conventional offline MT decoding is carried out using beam search. However, the streaming setup conditions the decoding process in simultaneous MT. First, the beam-search decoder is replaced by a greedy decoder to work under real-time constraints. Secondly, the decoding algorithm must carry out simultaneous translation, and start translating without having received the entire source sentence.

3.4 Direct segmentation model

Our streaming cascade-based ST system integrates the ASR and MT components described above in Sections 3.2 and 3.3. However, streaming ST poses additional challenges that combine those of translating error-prone ASR output with those of simultaneous MT processing unbounded word sequences. As discussed in Section 3.1, simultaneous Transformer-based MT systems trained at the sentence level are not able to properly produce longer translations than those observed in training. Besides, an additional related problem with Transformer models, already mentioned in Section 3.2, is the significant increase of computational requirements as a function of the length of the input sequence. Thus, an intermediate preprocessing step that perfectly accommodates the continuous output of the streaming ASR system to the current capabilities of simultaneous MT systems is needed to achieve real-time streaming ST. In this regard, this section introduces a novel state-of-the-art segmentation model specially suited for streaming cascade-based ST.

The goal of a segmenter in a ST pipeline is to split the continuous stream of words generated by the upstream ASR system into non-overlapping chunks that maximize the accuracy of the downstream MT system. This is necessary in order to transform unbounded-length ASR transcriptions into sentence-like chunks that can be processed by MT models, which have been trained using sentence-aligned data. Every time the segmenter emits an end-of-segment event, the MT encoder and decoder are reset to make a fresh start of the translation process. Although recent advances in document-level MT could alleviate the need for a segmenter (Junczys-Dowmunt 2019) in the future, as discussed above it still remains a necessary component for streaming cascade-based ST. In this work, the DS model is reviewed (Iranzo-Sánchez, Giménez, et al. 2020). This model innovatively considers a word context not only into the past, but also into the future, as well as acoustic information to take segmentation decisions on the ASR output.

Formally, the segmentation problem is the task of splitting a sequence of input words w_1^J into non-overlapping chunks. We represent this with a sequence of split/non-split decisions, y_1^J, with $y_j = 1$ if the associated word w_j is the word that ends a chunk; and $y_j = 0$, otherwise. In this work, as mentioned above, we incorporate acoustic word-based features $\check{\mathbf{x}}_1^J$ aligned with the sequence of words output by the ASR system. From a statistical viewpoint, the sequence of split/non-split decisions is taken on the basis of

$$\begin{aligned}\hat{y}_1^J &= \arg\max_{y_1^J} p(y_1^J \mid w_1^J, \check{\mathbf{x}}_1^J) \\ &= \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_1^{j-1}, w_1^J, \check{\mathbf{x}}_1^J). \end{aligned} \tag{2.14}$$

However, in a streaming setup, we need to bound the sequence to d words into the future (hereafter, *future window*) to meet latency requirements

$$\hat{y}_1^J = \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_1^{j-1}, w_1^{j+d}, \check{\mathbf{x}}_1^{j+d}). \tag{2.15}$$

Indeed, for computational reasons and to prevent an ever-growing unbounded history, the word sequence w_1^{j+d} is limit to n words into the past, and the acoustic sequence $\check{\mathbf{x}}_1^{j+d}$ drops its history as

$$\hat{y}_1^J = \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_{j-n}^{j-1},\, w_{j-n}^{j+d},\, \check{\mathbf{x}}_j^{j+d}). \tag{2.16}$$

The DS model estimates the probabilistic term in Eq. 2.16 as schematically depicted from bottom to top in Figure 2.3. First, the input word sequence w_{j-n}^{j+d} is replaced by an extended version that incorporates previous split decisions y_{j-n}^{j-1}. The new sequence w'^{j+d}_{j-n} inserts an end-of-chunk token into the text input sequence every time a split decision has been taken. Next, the corresponding word embeddings $\mathbf{h}_{j-n}^{j+d}$ of the input tokens are computed and input into a GRU-based RNN (K. Cho, Merriënboer, et al. 2014), represented by function $f_1(\,)$. So, the resulting text state vectors are defined as

$$\mathbf{s}_j^{j+d} = f_1(\mathbf{h}_{j-n}^{j+d}). \tag{2.17}$$

Acoustic word-based vectors $\check{\mathbf{x}}_j^{j+d}$ are obtained from three acoustic features associated to each word: duration of the current word, duration of the previous silence (if any), and duration of the next silence (if any). Next, the split probability is computed by concatenating text and audio vectors of the current word and those in the future window, and passing them through a FFN, represented by function $f_2(\,)$, as

$$p(y_j \mid y_{j-n}^{j-1}, w_{j-n}^{j+d}, \check{\mathbf{x}}_j^{j+d}) \approx f_2([\mathbf{s}_j^{j+d}; \check{\mathbf{x}}_j^{j+d}]). \tag{2.18}$$

The incorporation of the acoustic word-based vectors into the segmentation model has been shown to outperform a version of the segmentation model that only depends on the word sequence w_{j-n}^{j+d} in order to decide whether to split or not (Iranzo-Sánchez, Giménez, et al. 2020).

At training time, the components inside the dashed boundary in Figure 2.3 are first pre-trained using only text data, and this allows training with more data, as there is a limited amount of audio datasets that include explicit sentence-level segmentation. Then, the RNN is frozen and training of the FFN continues with the addition of acoustic word-based vectors.

The segmenter just described is a streaming-ready model, so no specific adaptation needs to be performed to the decoder in a streaming setup. A greedy and a beam-search decoders were implemented to search for the most probable sequence of split decisions according to Eq. 2.16, but no significant differences

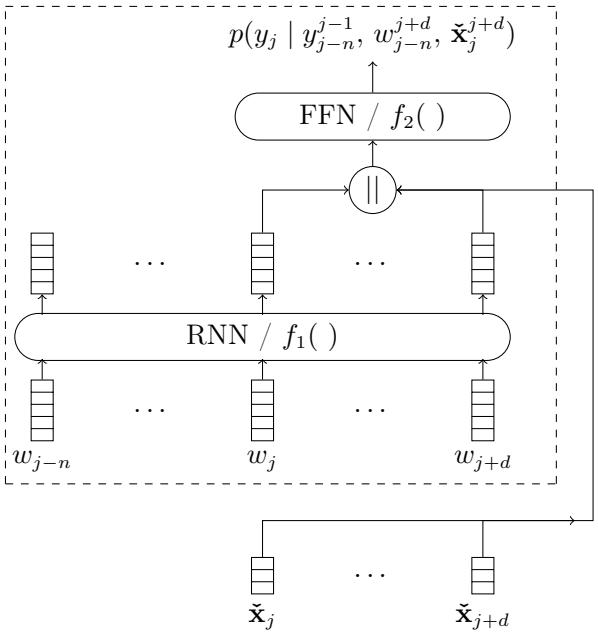

Figure 2.3: Architectural overview of the DS model. At the bottom, input acoustic word-based vectors $\check{\mathbf{x}}_j^{j+d}$ are found. Then, inside the dashed boundary, the input word sequence w_{j-n}^{j+d} is processed by an RNN and concatenated with the acoustic word-based vectors before passing through a FFN to output $p(y_j \mid y_{j-n}^{j-1},\, w_{j-n}^{j+d},\, \check{\mathbf{x}}_j^{j+d})$.

in performance were observed between them (Iranzo-Sánchez, Giménez, et al. 2020). Basically, this decoder moves a sliding window over the ASR output in order to decide whether to split or not after the current word. If a split decision is taken, an end-of-chunk token is inserted right after the current word w_j, and the decoding process continues.

3.5 Evaluation

In this section, after describing the experimental streaming setup, the ASR and MT components are independently assessed. Then, the complete ST pipeline concatenating the ASR system, the segmentation model and the simultaneous MT system is finally evaluated in terms of accuracy and latency.

Experimental setup

In order to properly evaluate the accuracy and latency of the proposed ST system, the testing conditions must mirror as close as possible those of a real streaming ST use case. This is why we have decided to use the Europarl-ST corpus (Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020) for evaluation purposes. The Europarl-ST corpus is a collection of interventions carried out by Members of the European Parliament (MEP) between 2008 and 2012, jointly with their corresponding transcriptions and translations. Unlike other corpus, the data is provided aligned at both, segment and intervention levels. Therefore, the segment-aligned data can be used during training, and entire interventions can be used at testing time in order to simulate real streaming conditions. This is the experimental setup that has been selected for this work. The advantage of this setup is that, by using the entire interventions at testing time, we are able to properly measure the performance of the streaming ST system in its intended setting. As mentioned in Section 3.1, each intervention is several minutes long, and this motivates the inclusion of the segmenter component, as the MT system would be unable to translate the entire recording otherwise. Additionally, the ST task of parliamentary debates is a realistic and challenging task that currently receives much interest due to the actual need to find an accurate enough solution in the near future (European Parliament and DG Translation 2019).

In this work, a state-of-the-art streaming Spanish ASR system is cascaded with simultaneous MT systems to perform ST from Spanish (Es) into French (Fr) and English (En). The statistics of the language pairs of the Europarl-ST corpus involved in our evaluation are shown in Table 2.19. For the purpose of

Table 2.19: Basic statistics of the Europarl-ST corpus for the training, development and test sets for the Es-En and Es-Fr language pairs.

	Training					Dev				
Lang Pairs	Vids	Chks	Hrs	Kwords Src	Kwords Trg	Vids	Chks	Hrs	Kwords Src	Kwords Trg
Es-En	727	7402	21.6	203	200	202	1947	5.7	53	53
Es-Fr	439	4673	13.7	129	149	121	1115	3.2	30	35

	Test				
Lang Pairs	Vids	Chks	Hrs	Kwords Src	Kwords Trg
Es-En	206	1816	5.3	50	50
Es-Fr	124	1082	3.2	31	36

these statistics, an oracle segmentation based on end-of-sentence punctuation marks was applied to split videos into chunks. Consequently, the average length of the resulting chunks was 10 seconds, and 27 to 28 words for Spanish and English, and 32 to 33 words for French.

Automatic Speech Recognition

The AMs integrated in our ASR system follow the hybrid approach introduced in Section 3.2. First, a GMM-HMM is used to initialize the required alignments to train the subsequent DNN architectures. Then, context-dependent FFN-HMMs with three left-to-right states are trained. More precisely, our ASR system uses 48-dimensional feature vectors as a result of preprocessing with a Hamming window of 25ms shifted at 10ms intervals into 16 Mel-frequency cepstral coefficients (MFCC) plus deltas and accelerations (Zolnay, Schluter, and Hermann Ney 2005). The input to the FFN is a context of 11 frames unrolled into a 528-dimensional vector. This FFN includes 8 layers containing 2048 hidden units each followed with Rectified Linear Units (ReLU), and a final softmax layer with 10K labels corresponding to the number of clustered subphonetic units considered in this task. This network is trained using plain backpropagation to optimize cross-entropy. The feature extraction process, the training of the GMM-HMM and the FFN-HMM systems were performed with the transLectures UPV toolkit (TLK) (Agua et al. 2014).

The FFN was used to bootstrap a BLSTM-HMM with 8 layers and 512 units per direction. Differently from the FFN, the BLSTM network uses 85-dimensional filter-bank features (Aggarwal and Dave 2012). This network is trained us-

Table 2.20: Statistics of transcribed Spanish speech data sources used to train AMs.

Data source	Hours
Internal: TV, entertainment	3034
Internal: education	306
Internal: user-generated content	202
Internal: politics	158
Internal: audiobooks	21
RTVE2018(Lleida et al. 2019)	205
TOTAL	3926

ing TensorFlow (Abadi et al. 2015) with cross-entropy loss during 16 epochs. Dropout (G. E. Hinton et al. 2012) and specaugment (Park et al. 2019) regularization techniques were used to improve the generalization of the model. We performed Back-Propagation Through Time (BPTT) limited to 50 frames according to (Albert Zeyer, Doetsch, et al. 2017).

Table 2.20 shows statistics of the transcribed speech data sources used to train our AMs. Figures of internal, private speech data sources are provided organized by domain. Overall, almost four thousand hours were used for training. In addition, a multi-domain dev set of 41 hours, which included the dev set of Europarl-ST, was used to tune model hyperparameters.

Table 2.21 shows statistics of data sources adding up to over 3.4 billion of words devoted to LM training. First, we trained a 4-gram model with Kneser-Ney discount (Kneser and Hermann Ney 1995) using the SRILM toolkit (Stolcke 2002). We limited the system vocabulary to the most probable 255K words. Next, concerning the neural LMs, we trained both a LSTM-based and a Transformer-based LMs (TLM). On the one hand, our LSTM LM, with a 256-dimensional embedding and two layers of 2048 hidden units, was trained using the CUED-RNNLM toolkit (Xi Chen et al. 2016) for 6 epochs. BPTT was set to consider the 6 previous words. Training criterion was based on the Noise Contrastive Estimation (NCE) (Mnih and Teh 2012) to accelerate the training process. Also, VR was used to speed up the inference process. For this model, we sampled a 500M words subset from the available training data to accelerate the training process. On the other hand, we trained a Transformer LM using a customized version of the FairSeq (Ott et al. 2019) toolkit, with the same training dataset as the LSTM, using a configuration consisting of a 24 layer network with 768 units per layer, 4096-unit FFN, 12 attention heads, and an embedding of 768 dimensions. This model was trained during 8 epochs, with batches limited to 512 tokens, 512 sentences, and 512 words per

Table 2.21: Statistics of Spanish text resources used for language modelling. S=Sentences, RW=Running words, V=Vocabulary. Units are 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 (Tiedemann 2012)	212635	1146861	1576
UFAL (*UFAL Medical Corpus* 2018)	92873	910728	2179
Wikipedia (*Wikipedia* 2015)	32686	586068	3373
UN (Callison-Burch, Koehn, Monz, et al. 2012)	11196	343594	381
News Crawl (*News Crawl corpus* 2015)	7532	198545	648
eldiario.es (*Eldiario.es* 2017)	1665	47542	247
El Periódico (*ElPeriodico.com* 2017)	2677	46637	291
Common Crawl (*CommonCrawl* 2014)	1719	41792	486
News Commentary (*News Crawl corpus* 2015)	207	5448	83
TOTAL	369434	3423146	5785

sentence. Model parameters were updated every 32 batches. During inference, VR was also used to speed up TLM score computation. Both neural LMs and the 4-gram LM used the same vocabulary. The out-of-vocabulary (OOV) ratio in this task for this vocabulary was less than 0.4%, in both dev and test sets.

Table 2.22 shows the figures of baseline experiments with perplexities, weights for the interpolated models, and Word Error Rate (WER) for the three types of LMs considered in this work: n-gram (NG), LSTM, and Transformer (TLM), and their interpolated combinations. Hyperparameters were tuned on the dev set as defined in (Baquero-Arnal et al. 2020). These baseline experiments allow us to select the best LM combination for the streaming setup.

Regarding these results, while the difference in terms of perplexity is significant when considering neural LMs and its combinations, this improvement is not reflected in terms of WER, where the interpolated models provided very similar figures. The conclusion that can be drawn after these results is that the AM is sound, and it can depict promising paths during the decoding, not requiring much help from the LM. This is reflected in the fact that a small relative reduction of 6.7% in WER is the difference between the LMs with the highest (n-gram) and the lowest (three-way interpolation) perplexity. Therefore, in favor of studying the history limitation of the TLM, and to keep the decoding process as lightweight as possible, we have selected for the following experiments the interpolated model combining the n-gram and the TLM. The

Table 2.22: PPLs, interpolation weights and WERs for EuroParl dev and test sets.

Model	PPL		Weights	WER	
	dev	test		dev	test
NG	70.3	78.4	-	10.5	11.3
LSTM	46.3	54.4	-	10.2	10.9
TLM	32.1	37.6	-	9.9	10.7
NG+LSTM	41.7	48.0	(0.20/0.80)	10.0	10.8
NG+TLM	30.2	34.8	(0.10/0.90)	9.8	10.5
LSTM+TLM	32.0	37.5	(0.07/0.93)	9.9	10.6
NG+LSTM+TLM	30.2	34.8	(0.09/0.04/0.87)	9.8	10.5

interpolation of n-gram and TLMs was also proved in (Baquero-Arnal et al. 2020) to be an essential ingredient of our streaming ASR systems for English when positively compared in well-established benchmarks to other state-of-the-art streaming ASR systems (Moritz, Hori, and Le 2020; Zhang, Lu, et al. 2020; Zhou et al. 2020).

The following set of experiments are devoted to study the impact of the streaming parameters presented in Section 3.2 on the system performance. These parameters are the $n_{\text{lookahead}}$ that defines the length in seconds of the sliding window and has a direct impact on the baseline latency, the history size for the TLM, and finally the LMHR and LMHP, that are related to the pruning process in order to minimize the computational requirements of the neural LMs.

Figure 2.4 shows results on WER as a function of the $n_{\text{lookahead}}$ in seconds on the Europarl-ST dev set. The rest of the parameters are fixed as defined in the baseline experiments. As expected, lower WER figures are achieved as the length of the lookahead window grows to consider more future frames to compute the acoustic score. However, we need our system to work under real-time constrains, meaning that we should ensure a reasonable trade-off between WER and latency. In this case, setting this parameter to a particular value introduces a fixed delay equal to $n_{lookahead}$ seconds, that again, is the length of the sliding window that is applied over the input stream. Taking into account our previous work in streaming ASR (Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020), an $n_{lookahead}$ value of 0.6 seconds is a reasonable baseline delay, since subsequent components of the cascade ST system will introduce additional delays. Hence, we fixed this value for the following experiments.

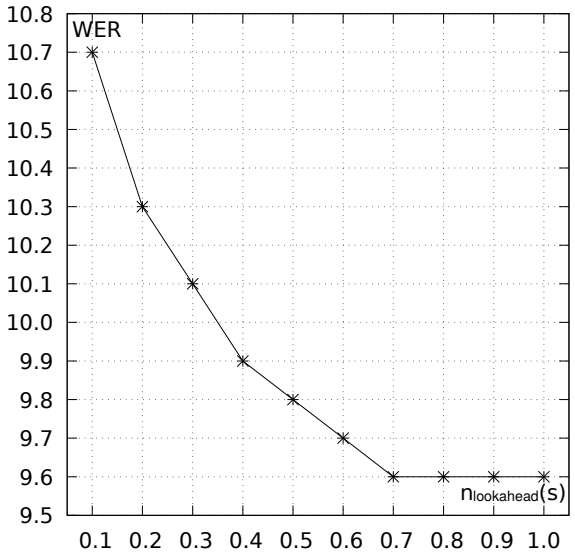

Figure 2.4: WER vs $n_{\text{lookahead}}$ in seconds on the EuroParl-ST dev set.

As mentioned in Section 3.2, the computational cost of the TLM requires to limit its history size in order to perform streaming decoding. For this reason, Figure 2.5 explores the impact of the TLM history size, in terms of the number of words n, evaluating the perplexity on the dev set.

As observed in Figure 2.5, increasing the history size consistently decreases perplexity, reaching a minimum value with a history size of 50 words. Additionally, we have validated this parameter in terms of WER, but differences were not significant on the Europarl-ST dev set. Therefore, we decided to adopt a history size of 50 words.

Regarding decoding pruning parameters, we have also studied the effect of the LMHR parameter that controls the length of the previous context to perform hypothesis recombination during decoding. In line with the effect of history size discussed above, LMHR had little impact in WER on the Europarl-ST dev set. For this reason, and considering that a shorter context involves earlier hypothesis recombination and consolidation reducing system latency, LMHR was set to 3.

The second pruning-related parameter is the LMHP, that controls the number of active hypotheses at word level during decoding. Consequently, this param-

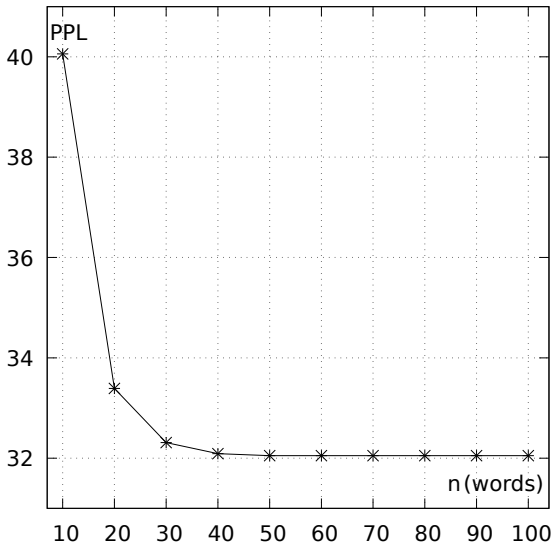

Figure 2.5: Perplexity as a function of the TLM history size measured in n words on the EuroParl-ST dev set.

eter affects WER and the Real Time Factor (RTF) of the system. The RTF is defined as the ratio between the decoding time of an audio input and its duration. Figure 2.6 illustrates WER vs. RTF results varying beam width without limiting LMHP (LMHP=Inf) and fixing that value to 80. As observed, limiting the number of active hypotheses to 80 has no negative impact on WER in line with our previous work (Jorge, Giménez, Iranzo-Sánchez, Silvestre-Cerdà, et al. 2020). This means that the information provided by the AM is definitively enough to figure out the best path, so the number of active hypotheses querying the LM are somehow limited in the Europarl-ST task. Considering this, LMHP equal to 80 is adopted for the rest of the experiments, to ensure that the performance of the decoder is kept even in difficult parts of the decoding.

To sum up, in this section we have defined the ASR system that will provide the transcriptions to the MT system. The final ASR system is based on a BLSTM acoustic model with a lookahead context window of 0.6 seconds, using the interpolation of the n-gram model and the TLM with a limited history of 50, with the pruning parameters LMHR equal to 3 and LMHP equal to 80. This system provides 9.8 and 10.5 WER points on Europarl-ST dev and

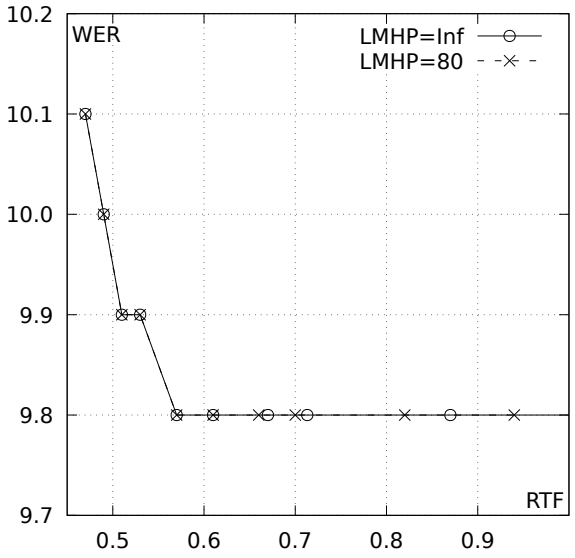

Figure 2.6: WER vs. RTF as a function of the beam width without limiting the value of LMHP (LMHP=Inf) and considering LMHP equal to 80 on the Europarl-ST dev set.

test sets, respectively. These figures are good enough to provide high-quality transcriptions to ease the downstream MT process.

Simultaneous Machine Translation

Offline and simultaneous MT systems were trained for each of the translation directions using the Transformer BASE configuration (Vaswani et al. 2017) implemented with the Fairseq toolkit (Ott et al. 2019). The initial models are general out-of-domain systems trained with the data shown in Table 2.23. After training finishes, domain adaptation by finetuning (Luong and Manning 2015) was carried out using the Europarl-ST training data. Finetuning is performed using the SGD optimizer and a fixed learning rate, equal to that used in the general-domain model when training finished. Early stopping is carried out by measuring performance against the Europarl-ST dev set.

In the case of the MMAH models, the trade-off between accuracy and latency has been explored by training several models varying the hyperparameter λ. Wait-k models are trained using the multi-path strategy sampling different values of k at each training step, and therefore the value of k can be specified at

Table 2.23: Training data used for the general-domain neural MT systems in millions of sentence pairs.

Corpus	Es-En (M)	Es-Fr (M)
Common Crawl (*CommonCrawl* 2014)	1.8	–
DGT (Tiedemann 2012)	–	4.8
EU Bookshop (Tiedemann 2012)	5.2	4.9
EU Bulletin (*EU Bulletin* 2009)	1.0	–
JRC-Acquis (Tiedemann 2012)	–	1.6
UN (Callison-Burch, Koehn, Monz, et al. 2012)	11.2	25.8
Wikipedia (Tiedemann 2012)	1.8	–

decoding time. The accuracy of MT systems is evaluated in terms of Bilingual Evaluation Understudy (BLEU) (Papineni et al. 2002). BLEU gauges the degree of n-gram overlapping between the automatic and reference translations ranging n from 1 to 4. In addition, a penalization factor is included if the automatic translation is shorter than the reference translation.

In addition to BLEU, the theoretical latency of simultaneous MT systems is usually evaluated in terms of delay of the output with respect to the input by three measures: Average Proportion (AP) (K. Cho and Esipova 2016), Average Lagging (AL) (M. Ma et al. 2019) and Differentiable Average Lagging (DAL) (Cherry and Foster 2019). To define AP, AL and DAL we need to introduce function $g(\ )$, that for each output position i, $g(i)$ indicates how many words from the input had been read when output word e_i was written. AP is an average delay, in terms of input words, taking into account the source sentence length $|\mathbf{w}|$, that is,

$$\mathrm{AP} = \frac{1}{|\mathbf{w}| \cdot |\mathbf{e}|} \sum_{i=1}^{|\mathbf{e}|} g(i). \tag{2.19}$$

AL can be understood as the average delay, in terms of input words, of the system with respect to an ideal translator that does not need to wait for input words to generate the next word (wait-0 policy) (M. Ma et al. 2019), that is,

$$\mathrm{AL} = \frac{1}{\tau} \sum_{i=1}^{\tau} g(i) - \frac{i-1}{\gamma} \tag{2.20}$$

where $\gamma = |\mathbf{e}|/|\mathbf{w}|$.

In order to account for differences in source and target length, the measure is computed only up to the input position at which the entire source sentence has been fully read

$$\tau = \underset{i:g(i)=|\mathbf{w}|}{\arg\min}\, g(i) \tag{2.21}$$

Both AP and AL, specially the former, present some issues (Cherry and Foster 2019). DAL tries to solve those issues by assigning a cost to write operations, while at the same time presenting a measure that is differentiable and could be part of a loss function. In order to do this, a modified delay $g'(i)$ including the cost of writing operations is computed as

$$g'(i) = \begin{cases} g(i) & i = 1 \\ \max\left(g(i), g'(i-1) + \frac{1}{\gamma}\right) & i > 1 \end{cases} \tag{2.22}$$

So, the DAL is defined as

$$\text{DAL} = \frac{1}{|\mathbf{e}|} \sum_{i=1}^{|\mathbf{e}|} g'(i) - \frac{i-1}{\gamma} \tag{2.23}$$

First, we evaluate the performance of MT systems by themselves, using the Europarl-ST dev set reference transcriptions and oracle segmentation based on end-of-sentence punctuation marks, in order to measure the accuracy gap incurred between offline and simultaneous MT systems. Table 2.24 reports comparative BLEU and latency measures as a function of the hyperparameter λ for MMAH and k for wait-k when translating from Spanish into English and French. Values of λ and $k \leq 4$ were selected so that similar latency figures were obtained for MMAH and wait-k systems, and a fair comparison in terms of BLEU is possible. In the case of wait-k, an exceptionally high value for k ($k = 32$) simulating offline behavior was additionally tested to compare BLEU scores with the offline systems. As observed, in this latter comparison, the offline system supersedes the wait-k system by 4.9 and 3.7 BLEU points for Es-En and Es-Fr, respectively. This gap in BLEU is the baseline translation quality degradation of deploying simultaneous vs. offline MT systems in order to guarantee low-latency ST.

In the case of MMAH systems, latency is correlated with λ, since as λ increases, latency decreases. When comparing BLEU scores across λ values, we observe

Table 2.24: BLEU, AP, AL and DAL results on the Europarl-ST dev set for Es-En and Es-Fr with reference transcriptions and oracle segmentation as a function of the hyperparameter λ.

Model		Es-En				Es-Fr			
	λ/k	BLEU	AP	AL	DAL	BLEU	AP	AL	DAL
Offline	-	44.3	1.00	29.68	29.68	33.5	1.00	31.09	31.09
MMAH	0.1	31.0	0.69	5.58	9.79	25.6	0.62	3.28	7.24
	0.2	29.9	0.63	3.79	7.84	23.0	0.60	2.44	6.42
	0.4	30.5	0.62	3.37	6.64	24.4	0.59	2.23	5.98
Wait-k	1	32.6	0.57	2.14	2.89	27.0	0.62	3.80	4.76
	2	35.2	0.59	2.90	3.49	29.1	0.65	4.61	5.46
	4	37.6	0.65	4.47	5.05	29.9	0.70	6.15	7.00
	32	39.4	0.99	25.4	25.29	29.8	0.99	25.99	26.23

only a slight improvement from $\lambda = 0.4$ to $\lambda = 0.1$, since heads gain in freedom to align far apart one from the others. In wait-k systems, higher values of k have larger delays as the model must wait longer before starting to translate, but this results in better translation quality. The value of k has a significant effect on quality for Es-En, whereas for Es-Fr, k values higher than 4 have very similar performance.

As reported in Table 2.24, for the smallest latency values, wait-k models outperform the equivalent MMAH models. As we increase k allowing for slightly longer delays, wait-k models are much better than MMAH models. Specifically, the wait-4 configuration is 6.6 and 4.3 BLEU points better than the best MMAH model in Es-En and Es-Fr, respectively. As a result of this comparison, the multi-path wait-k approach was selected in the rest of the experiments.

Speech Translation

Once the ASR system has been adjusted for streaming conditions in Section 3.5 and the simultaneous MT system was selected in Section 3.5, we move on to evaluate the complete ST pipeline, including the segmentation model which allows for a seamlessly connection between the ASR and MT systems under a streaming setup.

The architectural overview of the segmentation model provided in Section 3.4 is instantiated here. First, an embedding layer of size 512 followed by a GRU-based RNN of the same size is used in order to process the ASR text output. The generated word state vectors are then combined with the acoustic word-

based feature vectors and fed into a two-layer FFN with ReLU activation, followed by a softmax output layer. During training, dropout of 0.2 is applied after the text-based RNN as well as after each layer of the FFN. Chunks belonging to the split class are upsampled so that, on average, one third of the samples of each batch are split samples. Otherwise, the model has trouble converging, as the data is heavily unbalanced. As previously mentioned, training is carried out by first using a model with only a text RNN, and once it achieves convergence, its weights are frozen and training continues with the addition of the acoustic features.

This segmentation model was trained using the Europarl-ST data, as well as additional data from the Europarl corpus (Koehn 2005) from years not covered by Europarl-ST. Its hyperparameters, history size and future window length, were tuned on the Europarl-ST dev set. As in (Iranzo-Sánchez, Giménez, et al. 2020), longer history sizes improve BLEU scores of the ST system up to a certain point, but similar BLEU scores are obtained beyond a history size of 10 words. Thus, history size was fixed to 10 words in these experiments. However, the future window length has a significant impact on the performance of the ST system, not only in terms of BLEU scores, but also in latency. It should be reminded that the future window length d is the number of future words the segmenter needs to see in order to make a split decision after the current word. So, a trade-off between translation quality and latency in the segmentation model needs to be found in conjunction with the simultaneous MT model.

Indeed, there are two main factors that contribute to the quality-latency trade-off of the ST system: the already mentioned future window length d of the segmentation model and the hyperparameter k of the wait-k models, that is, the number of source words that the simultaneous MT system needs to read before starting the translation. At the beginning, given an input ASR stream, the segmenter and wait-k models accumulatively wait for $d+k-1$ words before the MT system starts writing the translation. Then, the MT system will write a translated word each time a new input word is received from the segmenter, following the wait-k schedule.

In Section 3.5, we have discussed theoretical latencies in terms of how many words the output is behind the input for simultaneous MT systems. However, the MT system must also be able to work fast enough not to fall behind the ASR system in a streaming setup. High latencies in a streaming ST system can be caused by waiting too long for the ASR input, or due to a high computational cost of the MT system itself that makes it unfeasible to process the ASR output under real-time constraints. This is specially crucial in the ST case, because due to the dependency on the ASR input, and the realistic

testing conditions, the system must keep an adequate throughput during the entire streaming session. A simultaneous MT system that translates significantly ahead of the ASR output does nothing to improve the response time, while at the same time, if at any point the MT system falls behind the ASR stream, it will need to catch up at some point in the future. This means that short bursts of translation speed can only be used to catch up and make up for previous slowdowns, but otherwise offer no advantage. We will test the behaviour of our cascade ST system by measuring its latency in our realistic streaming scenario.

To this purpose, we define accumulative word-level latencies at three points in the system, as the time elapsed between a word spoken, and: 1) The moment the consolidated hypothesis for that word is provided by the ASR system; 2) The moment the segmenter has processed that word on the ASR consolidated hypothesis; 3) The moment the MT system translates that word after being processed by the segmenter. In addition, some considerations should be done about how latencies have been estimated. On the one hand, it should be noticed that this ST system is working with ASR consolidated hypotheses in the sense that these hypotheses will not change as the audio stream is further processed. Working with non-consolidated hypotheses it is also possible, and in fact it reduces drastically the ASR latency. However, although in our experience non-consolidated hypotheses are suitable for an ASR streaming scenario, we realised that when combined with an MT system it produces an annoying flickering effect. For this reason, despite the increase in ASR latency it was decided to work only on consolidated hypotheses. On the other hand, determining when a spoken word has been translated is not a trivial task since translation is not a monotonic process, and hence, a correspondence between input and output words is required. Although it would be possible to retrieve alignments from the translation process, in order to simplify the estimation of the MT latency, it was decided to use the approximation recently proposed in (N. Arivazhagan et al. 2020). In this approach, the estimation of the alignment between words is approximated by assuming a uniform monotonic alignment. More precisely, for a given output word e_i the position j of its corresponding word in the input sentence is calculated as $j = i \cdot |\mathbf{w}|/|\mathbf{e}|$.

Table 2.25 reports accumulative word-level latencies, in terms of mean and standard deviation, for the ASR system plus the segmentation model, before detailing latencies of the complete ST system when incorporating the MT system. As previously mentioned, latencies were computed using complete MEP interventions. All reported latencies are measured on a machine with a i7-3820 CPU and a RTX 2080Ti GPU.

Table 2.25: Accumulative word-level latencies in seconds (mean ± std. dev.) for the ASR and segmenter components of the ST system on the Europarl-ST dev set.

	Latency (seconds)
ASR	1.6 ± 0.5
+ Seg. (d=0)	2.0 ± 0.6
+ Seg. (d=1)	2.4 ± 0.7
+ Seg. (d=2)	2.8 ± 0.8
+ Seg. (d=4)	3.5 ± 0.9

As observed in Table 2.25, the ASR system introduces a latency of 1.6s. Around 0.9s is due to the lookahead context and other decoding aspects, while the other 0.7s is related to the fact we are working with consolidated hypotheses. The segmenter adds an additional latency that ranges from 0.4s to 1.9s depending on the future window length. This latency is mostly due to the need to wait for the words in the future window to be consolidated by the ASR, as the time taken by the segmenter to decide whether to split or not is negligible ($\simeq 0.01$s). In the case of $d = 0$, the additional delay of $0.4s$ with respect to the ASR system is due to the fact that the segmenter needs to wait for the consolidation of the next silence phoneme in order to compute the corresponding acoustic features.

Next, we focus on the trade-off between latency and quality of the complete ST system. Figure 2.7 shows BLEU scores vs. average word-level latency in seconds on the Europarl-ST dev set for Es-En (top) and Es-Fr (bottom) translation directions. Each curve represents a fixed value for the future window length $d = \{0, 1, 2, 4\}$ of the segmenter, while each point on this curve from left to right correspond to $k = \{1, 2, 4, 8\}$ of the wait-k model behind.

As observed, in general terms, for a given latency, it is more beneficial to use a lower value of d such as 1 or 2, paired with a higher value of k, than to use configurations with higher d but lower k. Therefore, we can conclude that, at lower latency regimes, increasing k has a bigger positive impact than increasing d. However, if we continue to increase k we quickly start to get diminishing returns, at which point it is more efficient to increase latency by giving more context to the segmenter. Overall, it can be said that the MT decoding strategy has a bigger impact on translation quality than the segmenter context, but segmentation quality remains a limiting factor for downstream MT performance. This means that sometimes it will be necessary to increase d if a certain MT quality threshold must be reached.

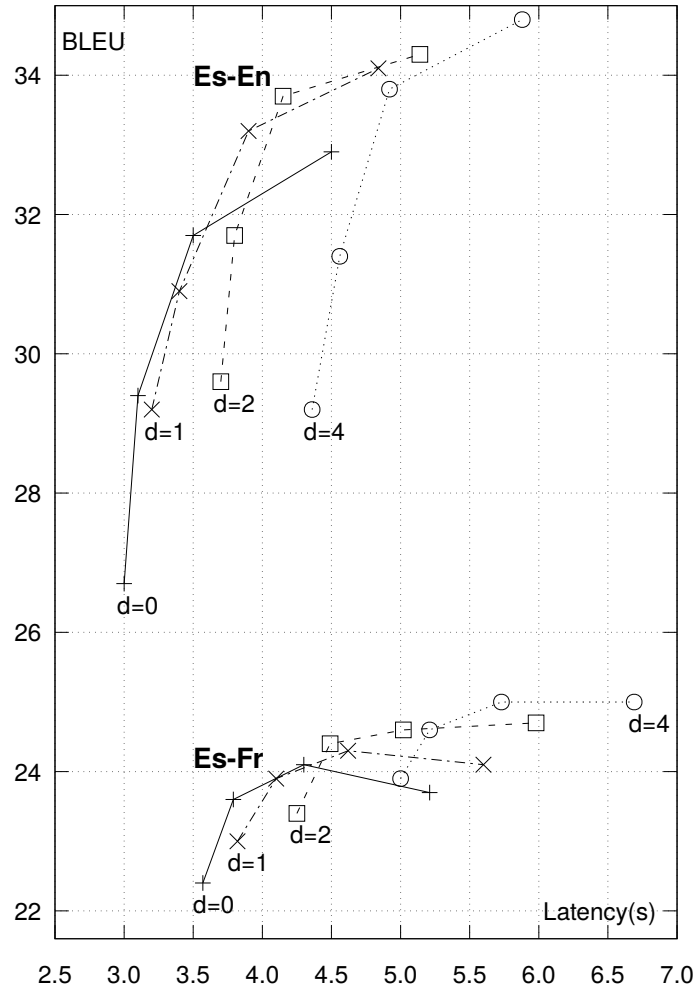

Figure 2.7: BLEU vs average word-level latency for Es-En (top) and Es-Fr (bottom) with future window length $d = \{0, 1, 2, 4\}$ of the segmentation model on Europarl-ST dev set. Points on each curve from left to right represent increasing values of $k = \{1, 2, 4, 8\}$ in the wait-k MT system.

Table 2.26: BLEU scores under the input settings for the wait-k MT system evaluated on the Es-En and Es-Fr Europarl-ST test sets.

Input	Es-En	Es-Fr
Ref. + Oracle Seg.	34.8	27.5
ASR + Oracle Seg.	31.8	24.7
ASR + DS	30.0	22.9

When looking for a final configuration to use, it would be ideal to choose one that maximizes quality without adding so much latency that the user experience is negative. We propose to use a latency similar to that of a professional (human) interpreter, so that our (artificial) interpreter can be used in a similar way. Fortunately, there exists ample literature about measuring the Ear-Voice Span (EVS) of human interpreters, which is the delay between a chunk being spoken and its corresponding translation being produced. Many factors have been shown to affect EVS (Yagi 2000), but a delay of 2-4 seconds is a reasonable expected value (Barik 1973; Lederer 1978; T.-H. Lee 2002). Therefore, we select a combination of d and k values that maximizes quality but does not exceed a latency of 4 seconds. Based on this, we have chosen $d = 1$ with $k = 4$ for Es-En, and $d = 0$ with $k = 2$ for Es-Fr.

Next, we evaluate the proposed ST system on the Europarl-ST test set in terms of accuracy and latency. First, in order to measure the degradation of the ST performance introduced by upstream ASR and/or segmentation errors, we compare the system accuracy depending on the input configuration: ASR output segmented with the DS model, ASR output with oracle segmentation and reference transcription with oracle segmentation. The oracle segmentation is based on end-of-sentence punctuation marks. Table 2.26 reports BLEU scores for the input configurations mentioned above on the Es-En and Es-Fr Europarl-ST test sets. First, the configuration of reference transcription plus oracle segmentation defines an upper bound for BLEU scores when neither ASR nor segmentation errors are present. Then, the ASR output plus oracle segmentation allows to know how much translation accuracy degradation is introduced by the DS model shown in the last row. As observed, the degradation in BLEU scores with respect to the ASR output and DS model due to the effect of segmentation errors is 1.8 points in both language pairs, while the impact of segmentation plus ASR errors goes from 4.6 points for Es-Fr to 4.8 points for Es-En.

Table 2.27 shows accumulative word-level latencies for the three components of the ST system on the Es-En and Es-Fr Europarl-ST test sets. As expected from

Table 2.27: Accumulative word-level latencies in seconds (mean ± std. dev.) for the ASR, segmenter and MT components of the ST system on Es-En and Es-Fr Europarl-ST test sets.

	Es-En	Es-Fr
ASR	1.7 ± 0.5	
+ Seg.	2.4 ± 0.7	2.0 ± 0.6
+ MT	4.0 ± 1.8	3.9 ± 1.9

hyperparameter tuning on the dev set, the average latency of the complete ST systems is 4 seconds for both translation directions. Almost half of the latency is explained by the ASR, while the other half is due to the segmenter plus the MT system, though in different proportion depending on the language. As observed, the MT component introduces the greater amount of variability in the latency of the ST system.

Finally, we compare the translation accuracy and word-level latency of the DS model with other streaming segmentation schemes, integrating them into our streaming ST pipeline and tuning them on the dev set under the same maximum 4-second word-level latency constraint. Three segmentation schemes were initially considered: a VAD-based segmenter (Silvestre-Cerdà et al. 2012), a monolingual MT segmenter (E. Cho, Jan Niehues, and Waibel 2017) and an ASR-based segmenter grounded on the by-product of the ASR decoding when the special end-of-chunk token is recognized. However, the VAD-segmenter was discarded because it makes the ST system to work in an off-line manner. In other words, the VAD segmenter prevents the ST system from translating simultaneously since the ASR output only becomes available to the MT system as complete chunks, and consequently only chunk-level latencies could be measured. Table 2.28 shows comparative BLEU scores and accumulative word-level latencies for the ST system as a function of the segmentation scheme on the Europarl-ST test sets. As observed, the DS model and the ASR-based segmenter achieved similar BLEU scores, but both better than the monolingual MT segmenter. However, the DS model clearly exhibits a lower latency variance than the ASR-based segmenter, since the latter defines chunks that are approximately 60% longer than the former leading the simultaneous MT system to incur in a greater latency variability. In addition, it should be reminded that the ASR-based segmenter is a by-product of the ASR system. On the one hand, this means that the ASR-based segmenter is taking advantage of the large amount of data devoted to train the ASR system, indeed one order of magnitude larger than the DS model. But, on the other hand, the ASR-based segmenter is fully dependent on the ASR system in contrast to the high

Table 2.28: Comparative BLEU scores and accumulative word-level latencies across segmentation schemes evaluated on the Es-En and Es-Fr Europarl-ST test sets.

	Es-En		Es-Fr	
	BLEU	Latency	BLEU	Latency
Mono. MT	28.1	3.6 ± 4.7	21.4	3.7 ± 4.4
ASR-based	29.9	3.2 ± 3.1	23.1	3.9 ± 3.9
DS	30.0	4.0 ± 1.8	22.9	3.9 ± 1.9

flexibility provided by the DS model, that can be trained independently from the ASR system in terms of both, input features and model architecture.

3.6 Conclusions

In this work we have presented a state-of-the-art streaming ST system under the cascade approach. After revisiting from a streaming viewpoint the neural-based models behind the ASR and MT components, special attention is devoted to the direct segmentation model that allows to accommodate the continuous ASR output to the limited-length capacity of state-of-the-art simultaneous MT systems.

In ASR, the BLSTM network employed for acoustic modeling was modified in order to consider a lookahead context of future frames to deal with the progressive access to the input acoustic sequence in a streaming setup. Indeed, WER figures proved the impact of the lookahead context in the ASR system. On the other hand, neural-based LMs exploited the idea of the VR term to minimize inference time, while limiting the history size in training and via the LMHR and LMHP parameters in decoding had a minor effect in terms of WER, but allowed for low latencies on the Europarl-ST benchmark under real-time constraints.

Next, two state-of-the-art simultaneous MT systems, MMAH and multi-path wait-k, were assessed before deploying a streaming cascade-based ST pipeline. This pipeline integrating our DS model was extensively evaluated in conjunction with the best performing wait-k MT systems to guarantee low latency for usability purposes while preserving translation quality. In this respect, the DS model proved to play a crucial role in a streaming ST system to manage unbounded audio streams.

In terms of future work, performance improvements could be obtained by a closer coupling of the components of the cascade system. Currently, the si-

multaneous MT system has a translation policy that is independent from the ASR input stream. A dynamic policy that takes into account how many ASR words are ready could provide improvements in quality with little to none additional latency. Another research line to improve performance is to consider segmentation and translation as a joint problem, therefore avoiding a source of cascading errors. Finally, an adequately modified document-level MT model could carry out simultaneous translation without the need for a segmentation model.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 761758 (X5Gon) and 952215 (TAILOR); the Government of Spain's research project Multisub, ref. RTI2018-094879-B-I00 (MCIU/AEI/FEDER,EU) and FPU scholarships FPU14/03981 and FPU18/04135; and the Generalitat Valenciana's research project Classroom Activity Recognition, ref. PROMETEO/2019/111 and predoctoral research scholarship ACIF/2017/055.

References

Abadi, Martin et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org (cit. on p. 95).

Aggarwal, Rajesh and Mayank Dave (2012). "Filterbank optimization for robust ASR using GA and PSO". In: *International Journal of Speech Technology* 15, pp. 191–201 (cit. on p. 94).

Agua, Miguel A. del et al. (2014). "The Translectures-UPV Toolkit". In: *Advances in Speech and Language Technologies for Iberian Languages*. Ed. by Juan Luis Navarro Mesa et al. Springer International Publishing, pp. 269–278. ISBN: 978-3-319-13623-3. DOI: 10.1007/978-3-319-13623-3_28 (cit. on p. 94).

Arivazhagan, N. et al. (2020). "Re-Translation Strategies for Long Form, Simultaneous, Spoken Language Translation". In: *Proc. of ICASSP*, pp. 7919–7923 (cit. on pp. 83, 105).

Arivazhagan, Naveen et al. (2019). "Monotonic Infinite Lookback Attention for Simultaneous Machine Translation". In: *Proc. of ACL*. ACL, pp. 1313–1323. DOI: 10.18653/v1/P19-1126 (cit. on pp. 83, 88, 89).

Arısoy, Ebru et al. (2014). "Converting neural network language models into back-off language models for efficient decoding in automatic speech recog-

nition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1, pp. 184–192 (cit. on p. 87).
Bahar, Parnia, Tobias Bieschke, and Hermann Ney (Dec. 2019). "A comparative study on end-to-end speech to text translation". In: *Proc. of IEEE ASRU*, pp. 792–799. DOI: 10.1109/ASRU46091.2019.9003774 (cit. on p. 82).
Bahar, Parnia, Patrick Wilken, et al. (2020). "Start-Before-End and End-to-End: Neural Speech Translation by AppTek and RWTH Aachen University". In: *Proc. of IWSLT* (cit. on pp. 82, 83).
Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*. Ed. by Yoshua Bengio and Yann LeCun (cit. on p. 82).
Baquero-Arnal, Pau et al. (2020). "Improved Hybrid Streaming ASR with Transformer Language Models". In: *Proc. of Interspeech*, pp. 2127–2131 (cit. on pp. 86, 96, 97).
Barik, H. (1973). "Simultaneous Interpretation: Temporal and Quantitative Data". In: *Language and Speech* 16, pp. 237–270 (cit. on p. 108).
Barrault, Loïc et al. (2020). "Findings of the 2020 Conference on Machine Translation (WMT20)". In: *WMT* (cit. on p. 88).
Bengio, Yoshua et al. (2003). "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3, pp. 1137–1155 (cit. on p. 86).
Berard, Alexandre et al. (2018). "End-to-End Automatic Speech Translation of Audiobooks". In: *Proc. of ICASSP*. IEEE, pp. 6224–6228. DOI: 10.1109/ICASSP.2018.8461690 (cit. on p. 82).
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. 85).
Bozheniuk, Vitalii et al. (2020). "A comprehensive study of Residual CNNs for acoustic modeling in ASR". In: *Proc. of ICASSP*. IEEE (cit. on p. 85).
Callison-Burch, Chris, Philipp Koehn, Christof Monz, et al. (2012). "Findings of the 2012 Workshop on Statistical Machine Translation". In: *Proc. of WMT*. ACL, pp. 10–51 (cit. on pp. 96, 101).
Chan, William et al. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. of ICASSP*. IEEE, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621 (cit. on p. 82).
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. 86).
Chen, Xie et al. (2017). "Future word contexts in neural network language models". In: *Proc of ASRU*. IEEE Signal Processing Society, pp. 97–103 (cit. on p. 87).

Chen, Xi et al. (2016). "CUED-RNNLM — An open-source toolkit for efficient training and evaluation of recurrent neural network language models". In: *ICASSP 2016* (cit. on p. 95).

Cherry, Colin and George Foster (2019). "Thinking Slow about Latency Evaluation for Simultaneous Machine Translation". In: *arXiv:1906.00048* (cit. on pp. 101, 102).

Cho, Eunah, Jan Niehues, Kevin Kilgour, et al. (2015). "Punctuation insertion for real-time spoken language translation". In: *Proc. of IWSLT*. ISCA (cit. on p. 83).

Cho, Eunah, Jan Niehues, and Alex Waibel (2017). "NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation". In: *Proc. of Interspeech*. ISCA, pp. 2645–2649. DOI: 10.21437/Interspeech.2017-1320 (cit. on p. 109).

Cho, Kyunghyun and Masha Esipova (2016). "Can neural machine translation do simultaneous translation?" In: *arXiv preprint arXiv:1606.02012* (cit. on p. 101).

Cho, Kyunghyun, Bart van Merriënboer, et al. (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proc. of EMNLP*. ACL, pp. 1724–1734 (cit. on p. 91).

CommonCrawl (2014). http://commoncrawl.org/ (cit. on pp. 96, 101).

Dai, Zihang et al. (2019). "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proc. of ACL*, pp. 2978–2988 (cit. on p. 83).

Elbayad, Maha, Laurent Besacier, and Jakob Verbeek (2020). "Efficient Wait-k Models for Simultaneous Machine Translation". In: *Proc. of Interspeech*, pp. 1461–1465 (cit. on pp. 88, 89).

Eldiario.es (2017). https://www.eldiario.es/ (cit. on p. 96).

ElPeriodico.com (2017). https://www.elperiodico.com/ (cit. on p. 96).

European Parliament and DG Translation (2019). *Live Speech to Text and Machine Translation Tool for 24 Languages* (cit. on p. 93).

EU Bulletin (2009). https://ec.europa.eu/archives/bulletin/en/welcome.htm (cit. on p. 101).

Fügen, Christian, Alex Waibel, and Muntsin Kolss (2007). "Simultaneous translation of lectures and speeches". In: *Machine Translation* 21.4, pp. 209–252. DOI: 10.1007/s10590-008-9047-0 (cit. on p. 83).

Gangi, Mattia Antonino Di et al. (2019). "Enhancing Transformer for End-to-end Speech-to-Text Translation". In: *Proc. of MT Summit XVII Volume 1: Research Track*. EAMT, pp. 21–31 (cit. on p. 82).

Graves, Alex, Navdeep Jaitly, and Abdel-rahman Mohamed (2013). "Hybrid speech recognition with deep bidirectional LSTM". In: *Proc. of ASRU*. IEEE. IEEE Signal Processing Society, pp. 273–278 (cit. on p. 85).

Gu, Jiatao et al. (2017). "Learning to Translate in Real-time with Neural Machine Translation". In: *Proc. of EACL*. ACL, pp. 1053–1062 (cit. on p. 83).

Hinton, Geoffrey E et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (cit. on p. 95).

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 p. 85).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 85).

Iranzo-Sánchez, Javier, Adrià Giménez, et al. (2020). "Direct Segmentation Models for Streaming Speech Translation". In: *Proc. of EMNLP*, pp. 2599–2611 (cit. on pp. 83, 90, 91, 93, 104).

Iranzo-Sánchez, Javier, Joan Albert Silvestre-Cerdà, et al. (2020). "Europarl-ST: A Multilingual Corpus For Speech Translation Of Parliamentary Debates". In: *Proc. of ICASSP*. IEEE, pp. 8229–8233 (cit. on pp. 83, 93).

Irie, Kazuki et al. (2019). "Language Modeling with Deep Transformers". In: *Proc. Interspeech 2019*. ISCA, pp. 3905–3909. DOI: 10.21437/Interspeech.2019-2225 (cit. on pp. 82, 86).

Jia, Ye et al. (2019). "Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model". In: *Proc. of Interspeech*. Ed. by Gernot Kubin and Zdravko Kacic. ISCA, pp. 1123–1127. DOI: 10.21437/Interspeech.2019-1951 (cit. on p. 82).

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*, pp. 3820–3824. published (cit. on p. 83).

Jorge, Javier, Adrià Giménez, Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, et al. (Jan. 1, 2020). "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP*. IEEE (cit. on pp. 82, 83, 97, 99).

Jorge, Javier, Adria Giménez, et al. (2020). "LSTM-Based One-Pass Decoder for Low-Latency Streaming". In: *Proc. of ICASSP*. IEEE, pp. 7814–7818 (cit. on pp. 85, 86).

Jorge, Javier, Adrià Giménez, et al. (2019). "Real-Time One-Pass Decoder for Speech Recognition Using LSTM Language Models". In: *Proc. of Interspeech*, pp. 3820–3824 (cit. on p. 87).

Junczys-Dowmunt, Marcin (2019). "Microsoft Translator at WMT 2019: Towards Large-Scale Document-Level Neural Machine Translation". In: *Proc. of WMT*, pp. 225–233 (cit. on p. 90).

Kneser, Reinhard and Hermann Ney (1995). "Improved backing-off for m-gram language modeling". In: *Proc .of ICASSP*. Vol. 1. IEEE, pp. 181–184 (cit. on p. 95).

Koehn, Philipp (2005). "Europarl: A Parallel Corpus for Statistical Machine Translation". In: *Proc. of MT Summit*. AAMT, pp. 79–86 (cit. on p. 104).

Lederer, Marianne (1978). "Simultaneous Interpretation — Units of Meaning and other Features". In: *Language Interpretation and Communication*. Ed. by David Gerver and H. Wallace Sinaiko. Springer US, pp. 323–332 (cit. on p. 108).

Lee, Kyungmin et al. (2018). "Accelerating recurrent neural network language model based online speech recognition system". In: *arXiv:1801.09866* (cit. on p. 87).

Lee, Tae-Hyung (2002). "Ear Voice Span in English into Korean Simultaneous Interpretation". In: *Meta* 47.4, pp. 596–606 (cit. on p. 108).

Lleida, Eduardo et al. (2019). "Albayzin 2018 Evaluation: The IberSpeech-RTVE Challenge on Speech Technologies for Spanish Broadcast Media". In: *Applied Sciences* 9.24. ISSN: 2076-3417. DOI: 10.3390/app9245412 (cit. on p. 95).

Luong, Minh-Thang and Christopher D. Manning (2015). "Stanford Neural Machine Translation Systems for Spoken Language Domain". In: *Proc. of IWSLT* (cit. on p. 100).

Ma, Mingbo et al. (2019). "STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework". In: *Proc. of ACL*. ACL, pp. 3025–3036. DOI: 10.18653/v1/P19-1289 (cit. on pp. 83, 88, 89, 101).

Ma, Xutai et al. (2020). "Monotonic Multihead Attention". In: *Proc. ICLR 2020*. OpenReview.net (cit. on p. 88).

Miao, H. et al. (2020). "Transformer-Based Online CTC/Attention End-To-End Speech Recognition Architecture". In: *Proc. of ICASSP*, pp. 6084–6088 (cit. on p. 83).

Mnih, Andriy and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models". In: *arXiv preprint arXiv:1206.6426* (cit. on p. 95).

Mohri, Mehryar and Michael Riley (1999). "Integrated context-dependent networks in very large vocabulary speech recognition". In: *Proc. of ECSCT* (cit. on pp. 86, 88).

– (2001). "A weight pushing algorithm for large vocabulary speech recognition". In: *Proc. of ECSCT* (cit. on p. 86).

Moritz, N., T. Hori, and J. Le (2020). "Streaming Automatic Speech Recognition with the Transformer Model". In: *Proc. of ICASSP*, pp. 6074–6078 (cit. on pp. 83, 97).

News Crawl corpus (2015). http://www.statmt.org/wmt15/translation-task.html (cit. on p. 96).

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. 86).

Ney, Hermann and Stefan Ortmanns (2000). "Progress in dynamic programming search for LVCSR". In: *Proc. of IEEE* 88.8, pp. 1224–1240 (cit. on p. 86).

Niehues, J. et al. (2019). "The IWSLT 2019 Evaluation Campaign". In: *Proc. of IWSLT*. ISCA. DOI: 10.5281/zenodo.3525578 (cit. on p. 82).

Niehues, Jan et al. (2018). "Low-Latency Neural Speech Translation". In: *Proc. of Interspeech*. ISCA, pp. 1293–1297 (cit. on p. 83).

Nolden, D. (2017). "Progress in Decoding for Large Vocabulary Continuous Speech Recognition". PhD thesis. RWTH Aachen University, Germany (cit. on pp. 86, 87).

Oda, Yusuke et al. (2014). "Optimizing segmentation strategies for simultaneous speech translation". In: *Proc. of ACL*, pp. 551–556 (cit. on p. 83).

Ott, Myle et al. (2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proc. of NAACL-HLT: Demonstrations*. ACL (cit. on pp. 95, 100).

Papineni, Kishore et al. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proc. of ACL*. ACL, pp. 311–318. DOI: 10.3115/1073083.1073135 (cit. on p. 101).

Park, Daniel S. et al. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. Interspeech*, pp. 2613–2617 (cit. on pp. 82, 95).

Pino, Juan et al. (2019). "Harnessing Indirect Training Data for End-to-End Automatic Speech Translation: Tricks of the Trade". In: *Proc. of IWSLT*. ISCA. DOI: 10.5281/zenodo.3525032 (cit. on p. 82).

Popel, Martin and Ondřej Bojar (2018). "Training tips for the Transformer model". In: *The Prague Bulletin of Mathematical Linguistics* 110.1, pp. 43–70 (cit. on p. 83).

Povey, Daniel et al. (2011). "The Kaldi Speech Recognition Toolkit". In: *Proc. of ASRU*. IEEE Signal Processing Society (cit. on p. 86).

Raffel, Colin et al. (2017). "Online and Linear-Time Attention by Enforcing Monotonic Alignments". In: *Proc. of ICML*. Vol. 70. PMLR, pp. 2837–2846 (cit. on p. 88).

Rangarajan Sridhar, Vivek Kumar et al. (2013). "Segmentation Strategies for Streaming Speech Translation". In: *Proc. of NAACL-HLT*. ACL, pp. 230–238 (cit. on p. 83).

Russell, M. and R. Moore (1985). "Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition". In: *Proc. of ICASSP*. Vol. 10, pp. 5–8 (cit. on p. 85).

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. 85).

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. 85).

Schwenk, Holger (2007). "Continuous Space Language Models". In: *Computer Speech and Language* 21.3, pp. 492–518 (cit. on p. 86).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016a). "Improving Neural Machine Translation Models with Monolingual Data". In: *Proc. of ACL*. ACL, pp. 86–96. DOI: 10.18653/v1/p16-1009 (cit. on p. 82).

– (2016b). "Neural Machine Translation of Rare Words with Subword Units". In: *Proc. of ACL*. ACL, pp. 1715–1725. DOI: 10.18653/v1/p16-1162 (cit. on p. 82).

Shannon, C. E. (1948). "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3, pp. 379–423 (cit. on p. 86).

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 p. 86).

Silvestre-Cerdà, Joan Albert et al. (Nov. 22, 2012). "Albayzin Evaluation: The PRHLT-UPV Audio Segmentation System". In: *Proc. of IberSPEECH 2012*, pp. 596–600. published (cit. on p. 109).

Singh, Mittul, Youssef Oualil, and Dietrich Klakow (2017). "Approximated and Domain-Adapted LSTM Language Models for First-Pass Decoding in Speech Recognition." In: *Proc. of Interspeech*, pp. 2720–2724 (cit. on p. 87).

Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech*. Ed. by John H. L. Hansen and Bryan L. Pellom. ISCA, pp. 901–904 (cit. on p. 95).

Sundermeyer, Martin et al. (2014). "Lattice decoding and rescoring with long-span neural network language models". In: *Proc. of ISCA* (cit. on p. 87).

Tiedemann, Jörg (2012). "Parallel Data, Tools and Interfaces in OPUS". In: *Proc. of LREC*. ELRA, pp. 2214–2218 (cit. on pp. 96, 101).

UFAL Medical Corpus (2018). http://ufal.mff.cuni.cz/ufal_medical_corpus (cit. on p. 96).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS*. Ed. by Isabelle Guyon et al., pp. 5998–6008 (cit. on pp. 82, 86, 100).

Viterbi, Andrew (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 p. 86).

Weiss, Ron J. et al. (2017). "Sequence-to-Sequence Models Can Directly Translate Foreign Speech". In: *Proc. of Interspeech*. ISCA, pp. 2625–2629. DOI: 10.21437/Interspeech.2017-503 (cit. on p. 82).

Wikipedia (2015). https://www.wikipedia.org/ (cit. on p. 96).
Xu, H. et al. (2018). "A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition". In: *Proc. of ICASSP*, pp. 5929–5933 (cit. on p. 87).
Yagi, Sane (2000). "Studying style in simultaneous interpretation". In: *Meta: Journal des traducteurs/Meta: Translators' Journal* 45.3, pp. 520–547 (cit. on p. 108).
Zeyer, A., R. Schlüter, and H. Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Proc. of Interspeech*, pp. 3424–3428 (cit. on p. 85).
Zeyer, Albert, Parnia Bahar, et al. (2019). "A comparison of Transformer and LSTM encoder decoder models for ASR". In: *Proc. of ASRU*. IEEE Signal Processing Society, pp. 8–15 (cit. on p. 83).
Zeyer, Albert, Patrick Doetsch, et al. (2017). "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition". In: *Proc. of ICASSP*. IEEE, pp. 2462–2466 (cit. on pp. 85, 95).
Zeyer, Albert, Ralf Schlüter, and Hermann Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models". In: *Proc. of Interspeech 2016*. Ed. by Nelson Morgan. ISCA, pp. 3424–3428. DOI: 10.21437/Interspeech.2016-759 (cit. on p. 83).
Zhang, Q., H. Lu, et al. (2020). "Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss". In: *Proc. of ICASSP*, pp. 7829–7833 (cit. on pp. 83, 97).
Zheng, Baigong et al. (2019). "Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation". In: *Proc. of EMNLP-IJCNLP*. ACL, pp. 1349–1354. DOI: 10.18653/v1/D19-1137 (cit. on pp. 83, 89).
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. 97).
Zolnay, András, Ralf Schluter, and Hermann Ney (2005). "Acoustic feature combination for robust speech recognition". In: *Proc. of ICASSP 2005*. Vol. 1. IEEE, pp. I–457 (cit. on p. 94).

4 Stream-level Latency Evaluation for Simultaneous Machine Translation

Javier Iranzo-Sánchez, Jorge Civera, Alfons Juan

Findings of the Association for Computational Linguistics: EMNLP 2021, pp. 664-670

Punta Cana (Dominican Republic)

10.18653/v1/2021.findings-emnlp.58

7–11 November 2021

Stream-level Latency Evaluation for Simultaneous Machine Translation

Javier Iranzo-Sánchez, Jorge Civera, Alfons Juan

Abstract

Simultaneous machine translation has recently gained traction thanks to significant quality improvements and the advent of streaming applications. Simultaneous translation systems need to find a trade-off between translation quality and response time, and with this purpose multiple latency measures have been proposed. However, latency evaluations for simultaneous translation are estimated at the sentence level, not taking into account the sequential nature of a streaming scenario. Indeed, these sentence-level latency measures are not well suited for continuous stream translation, resulting in figures that are not coherent with the simultaneous translation policy of the system being assessed. This work proposes a stream-level adaptation of the current latency measures based on a re-segmentation approach applied to the output translation, that is successfully evaluated on streaming conditions for a reference IWSLT task.

4.1 Introduction

Simultaneous speech translation systems just started to become available (Bahar et al. 2020; Elbayad, Nguyen, et al. 2020; Han et al. 2020; Pham et al. 2020) thanks to recent developments in streaming automatic speech recognition and simultaneous machine translation. These systems seamlessly translate a continuous audio stream under real-time latency constraints. However, current translation latency evaluations (Ansari et al. 2020) are still performed at the sentence-level based on the conventional measures, Average Proportion (AP) (Cho and Esipova 2016), Average Lagging (AL) (Ma et al. 2019) and Differentiable Average Lagging (DAL) (Cherry and Foster 2019). These measures compute the translation latency for each sentence independently without

taking into account possible interactions that lead to accumulated delays in a real-world streaming scenario. Additionally, the current measures cannot be used by systems that do not use explicit sentence-level segmentation (Schneider and Waibel 2020).

In this work, we first revisit the conventional translation latency measures in Section 4.2 to motivate their adaptation to the streaming scenario in Section 4.3. Then, these adapted latency measures are computed and reported on an IWSLT task in Section 4.4. Finally, conclusions and future work are presented in Section 5.5.

4.2 Related work

Current latency measures for simultaneous translation can be characterised as a normalisation of the number of read-write word operations required to generate a translation $\boldsymbol{y}$ from a source sentence $\boldsymbol{x}$

$$L(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{y})} \sum_i C_i(\boldsymbol{x}, \boldsymbol{y}) \tag{2.24}$$

with Z being a normalisation function, i an index over the target positions and C_i a cost function for each target position i.

Depending on the latency measure, C_i is defined as

$$C_i(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} g(i) & \text{AP} \\ g(i) - \frac{i-1}{\gamma} & \text{AL} \\ g'(i) - \frac{i-1}{\gamma} & \text{DAL} \end{cases} \tag{2.25}$$

with

$$g'(i) = \max \begin{cases} g(i) \\ g'(i-1) + \frac{1}{\gamma} \end{cases} \tag{2.26}$$

where $g(i)$ is the number of source tokens read when a token is written at position i and γ is target-to-source length ratio $\frac{|\boldsymbol{y}|}{|\boldsymbol{x}|}$. Note that the AP cost function considers the absolute number of source tokens that has been read to output the i-th word, while AL and DAL cost functions account for the

number of source words the model lags behind a wait-0 oracle. This oracle simply accumulates a uniform distribution of source words over target positions according to the ratio $\frac{1}{\gamma}$. In the case of DAL, the recurrent definition of $g'(i)$ guarantees that the most expensive read-write operation is considered.

On the other hand, the normalisation function Z depends on the measure according to

$$Z(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} |\boldsymbol{x}| \cdot |\boldsymbol{y}| & \text{AP} \\ \underset{i:g(i)=|\boldsymbol{x}|}{\arg\min}\ i & \text{AL} \\ |\boldsymbol{y}| & \text{DAL} \end{cases} \tag{2.27}$$

The term in AP normalises the sum over the target sentence of absolute source tokens, while AL and DAL does over the number of target positions, which in the case of AL is limited to those target positions reading new source tokens. Indeed, the normalization term of AL is referred to as τ. The sentence-level latency measures just described are reported as an average value over an evaluation set of multiple sentence pairs, each one evaluated independently from the others.

However, the latency evaluation of a continuous paired stream of sentences has not received much attention, with the exception of the strategy proposed by (Schneider and Waibel 2020). This evaluation strategy considers the straightforward approach of concatenating all sentences into a single source-target pair in order to compute the corresponding latency measure. Next section outlines some drawbacks of this strategy (hereafter *Concat-1*) to motivate the discussion on how the current sentence-level latency measures could be adapted to the streaming scenario.

4.3 Stream-level evaluation

Let us consider the translation of a stream of two sentences, the first sentence has two input and two output tokens, while the second one has two input and four output tokens with ratios $\gamma_1 = 1$ and $\gamma_2 = 2$, respectively. The translation process is performed with a sentence-based wait-k system with catch-up characterised by a function $g(i) = \lfloor k + \frac{i-1}{\gamma} \rfloor$ with $k = 1$.

Table 2.29 compares the computation of the latency measures for the Concat-1 strategy (top) with the conventional strategy that considers independent sentences (bottom). Note that the translation process has only been carried

Table 2.29: Comparison of the latency metric computation between the Concat-1 (top) and the conventional sentence-level (bottom) strategy when using a wait-1 system.

									L
Concat-1		i	1	2	3	4	5	6	
		$g(i)$	1	2	3	3	4	4	
		$\frac{i-1}{\gamma}$	0	0.6	1.3	2.0	2.6	3.3	
		AP	1	2	3	3	4	4	0.7
	C_i	AL	1	1.3	1.6	1	1.3	-	1.2
		DAL	1	1.3	1.6	1.6	1.6	1.6	1.5
Ind. Sent.		i	1	2	1	2	3	4	
		$g(i)$	1	2	1	1	2	2	
		$\frac{i-1}{\gamma}$	0.0	1.0	0.0	0.5	1.0	1.5	
		AP	1	2	1	1	2	2	0.8
	C_i	AL	1	1	1	0.5	1	-	0.9
		DAL	1	1	1	1	1	1	1.0

out once, but both strategies are just interpreting the results differently as first denoted by their i and $g(i)$ values. The wait-0 oracle $\frac{i-1}{\gamma}$ of Concat-1, with a single global $\gamma = \frac{3}{2}$ underestimates the actual writing rate, and the system accumulates more delay than in the evaluation strategy of independent sentences, which uses a sentence-level estimation for γ.

These differences in results are magnified when computing latencies on a real streaming evaluation set. On the one hand, AL and DAL tend to obtain scores that do not reflect the real behaviour of the system when using a Concat-1 strategy with a single global γ, since the source-target length ratio varies wildly between different sentences. Therefore, the wait-0 oracle will sometimes overestimate the actual writing rate, and sometimes it will underestimate it. Moreover, the definition of DAL keeps the system from recovering from previously incurred delays, and therefore, every time the writing rate is underestimated, the system falls further and further behind the wait-0 oracle. On the other hand, AP turns out to be little informative when the stream is long enough, since AP always tends to be 0.5 because the delay incurred by a system with a reasonable k is always negligible compared with the total source length.

The accuracy of AL and DAL could be improved if sentence-level estimations for γ would be available somehow in a streaming scenario. With the availability of these estimations in mind, we formulate a streaming version of the cost functions in Eq. 2.43 based on a global $G(i)$ function, which returns the number

Table 2.30: Estimation of stream-level latencies measures on the same example proposed in Table 2.29.

								L
	i	1	2	1	2	3	4	
	$G(i+\lvert\boldsymbol{y}_1^{n-1}\rvert)$	1	2	3	3	4	4	
	$g_n(i)$	1	2	1	1	2	2	
	$\frac{i-1}{\gamma_n}$	0.0	1.0	0.0	0.5	1.0	1.5	
	AP	1	2	1	1	2	2	0.8
C_i	AL	1	1	1	0.5	1	-	0.9
	DAL	1	1	1	1	1	1	1.0

of source tokens (including those from previous sentences) that have been read as in the Concat-1 strategy:

$$C_i(\boldsymbol{x}_n, \boldsymbol{y}_n) = \begin{cases} g_n(i) & \text{AP} \\ g_n(i) - \frac{i-1}{\gamma_n} & \text{AL} \\ g'_n(i) - \frac{i-1}{\gamma_n} & \text{DAL} \end{cases} \tag{2.28}$$

with $g'_n(i)$ defined as

$$\max \begin{cases} g_n(i) \\ \begin{cases} g'_{n-1}(|\boldsymbol{x}_{n-1}|) + \frac{1}{\gamma_{n-1}} & i = 1 \\ g'_n(i-1) + \frac{1}{\gamma_n} & i > 1 \end{cases} \end{cases} \tag{2.29}$$

where $g_n(i) = G(i + |\boldsymbol{y}_1^{n-1}|) - |\boldsymbol{x}_1^{n-1}|$. Thus, the global delay is converted to a local representation so that it can be compared with the local sentence oracle.

Table 2.30 shows the computation of the stream-level latency measures as proposed in Eq. 2.45 for the same example calculated in Table 2.29. As observed, unlike with the Concat-1 strategy, we obtain the same results as in the conventional sentence-level estimation, while at the same time we keep the property that previous delays affect future sentences by basing our computations on the global delay $G(i)$.

If we use a segmentation-free model whose output is a single text stream, stream-level latency measures can be still computed by re-segmenting the output into sentence-like units (chunks). Formally, a segmenter takes an input stream Y and a set of reference sentences to compute a re-segmentation $\hat{\boldsymbol{y}}_1^N$ of Y. Once the re-segmentation is obtained, stream-level latency measures are estimated by considering paired input-output segments $(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n)$. In our case,

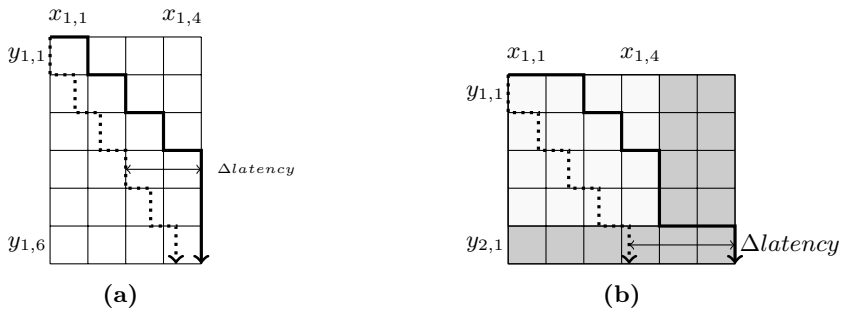

Figure 2.8: The examples shown above illustrate how a model which follows a wait-k policy can obtain AL/DAL values that differ from k. The bold lines show the behaviour of the model, the dotted lines show the oracle policy. Left: writing rate error with $k = 1$; the model uses $\hat{\gamma} = 1$, but the actual value is $\gamma = 1.5$. Right: segmentation error with $k = 2$; the first translated word of the second sentence is wrongly assigned to the first sentence during resegmentation, i.e. $\hat{y}_1 = (y_{1,1}, y_{1,2}, y_{1,3}, y_{1,4}, y_{2,1})$.

we re-segment by minimizing the edit distance between the stream hypothesis and the reference translations, analogously to the translation quality evaluation widely-used in speech translation (Matusov et al. 2005). Likewise, we can resegment the output to compute latency measures if our system uses a different segmentation than the reference.

Moreover, stream-level AL and DAL measures computed for a wait-k system are coherently close to k with two caveats. First, there can be deviation from the theoretical value of k due to a inaccurate estimation of the writing rate. Given that the wait-k policy uses a fixed γ, there will be some sentences in which this results in lower or higher writing rates than desirable. This is a feature inherent to the fixed policy itself. Second, a deviation could also occur due to re-segmentation errors. For instance, a word that is part of the translation of the n-th segment can be wrongly included into the previous $n-1$-th segment causing an increase of the latency. Both sources of latency are illustrated in Figure 2.8.

These two caveats given the definition of DAL imply that a system can never recover from previous delays, which might be an acceptable solution when computing latency measures at the sentence level, but it seems too strict and unrealistic when computing latency measures for streams comprising tens of thousands of words. To alleviate this problem, we propose to multiply the cost of a write operation $\frac{1}{\gamma_n}$ in $g'_n(i)$ by a scaling factor $s \in [0, 1]$. In practice, for values of s close to 1, this means that the write operation costs slightly less for

the real system than for the oracle. We believe this is an acceptable practical solution given that there are many ways that this could be achieved in real-world tasks, such as rendering subtitles slightly faster or, in the case of cascade speech-to-speech, slight reducing the duration of TTS segments or increasing the playback speed. Finally, the scaling factor s can be also understood as a hyperparameter that bridges the gap between AL ($s = 0$) and DAL ($s = 1$) and it can be adjusted depending on the actual writing cost of the translation task.

4.4 Experiments

The stream-level latency measures proposed in Section 4.3 are now computed and evaluated on the IWSLT 2010 German-English dev set (Paul, Federico, and Stüker 2010). To simulate a streaming scenario, all source sentences are concatenated into a single input stream. Then, they are segmented into sentences and translated with a wait-k fixed policy. As a result, it is expected that a well-behaved latency measure should rank the systems by increasing order of k.

Our streaming simultaneous translation system is based on a direct segmentation (DS) model (Iranzo-Sánchez et al. 2020) followed by a Transformer BASE model (Vaswani et al. 2017) trained with the multi-k approach (Elbayad, Besacier, and Verbeek 2020). The DS model was trained on TED talks (Cettolo, Girardi, and Federico 2012) with a future window of length 0 and history size of 10, while the translation model was trained on the IWSLT 2020 German-English data (Ansari et al. 2020). This system, which we will refer to as *Real*, uses catch-up with $\gamma = 1.24$. In addition to the Real system, three experimental setups based on different oracles are considered:

- *In. Seg.*: The input segmentation provided by the DS model is replaced by the reference segmentation to gauge segmentation errors.
- *Out. Seg.*: The reference segmentation is used to link each translation with its corresponding source sentence, therefore avoiding the need of re-segmentation by minimum edit distance.
- *Policy*: The translation model is replaced by an oracle model that outputs the reference translation with the appropriate writing rate for each sentence to account for errors due to a global γ.

Table 2.31: Stream-level AL as a function of k, computed using the Concat-1 approach on the IWSLT 2010 German-English dev set.

System	1	2	3	4	5
Real	-9.7	-12.0	-45.2	-23.7	-8.5
+In. Seg.	-42.9	-29.0	17.4	-10.1	25.5
+ Policy	14.2	15.1	16.0	16.8	17.6

AL (Table 2.31) and DAL (Table 2.32) have been computed using the Concat-1 approach, to serve as a baseline for the developed measures. These results confirm the problems of the Concat-1 approach, which have been identified and discussed on Section 4.3. AP results have been excluded from the tables, as no matter which setup is used, the computed AP is always 0.5. Likewise, the obtained AL and DAL values offer little insight about the latency behaviour of the model. These results are not only uninterpretable, but they also alter the ranking of the models. This could be specially worrisome if the Concat-1 approach was used to compare systems with adaptative policies that lack a explicit latency control such as k, as it might be harder to detect wheter the incoherent results are due to the adaptative policy or the latency measure itself. The only setup which returns the correct ranking is the one using the In. Seg. and Policy Oracles, but the latency results do not reflect the real behaviour of the model. The full AL and DAL results, for values up to $k = 10$ are reported in the appendix.

Table 2.32: Stream-level DAL as a function of k, computed using the Concat-1 approach on the IWSLT 2010 German-English dev set.

System	1	2	3	4	5
Real	15.0	11.0	17.4	11.3	20.3
+In. Seg.	4.5	8.5	37.1	24.6	52.3
+ Policy	85.8	86.7	87.7	88.7	89.7

Now that we have experimentally shown that the Concat-1 approach is unable to properly compute latencies, we move onto computing the stream-adapted version of the latency measures. The computation of stream-level AP (left), AL (center) and, DAL (right) with $s = 1.0$ and $s = 0.95$ (dashed lines) as a function of k in the multi-k approach are shown in Figure 2.9. The behaviour of AP and AL is that expected for the four experimental setups defined above, but the conventional DAL measure ($s = 1.0$) abruptly suffers the effect of not being able to recover from accumulated delays due to the cost of write operations. In contrast, DAL with $s = 0.95$ exhibits a smooth interpretable behaviour as a

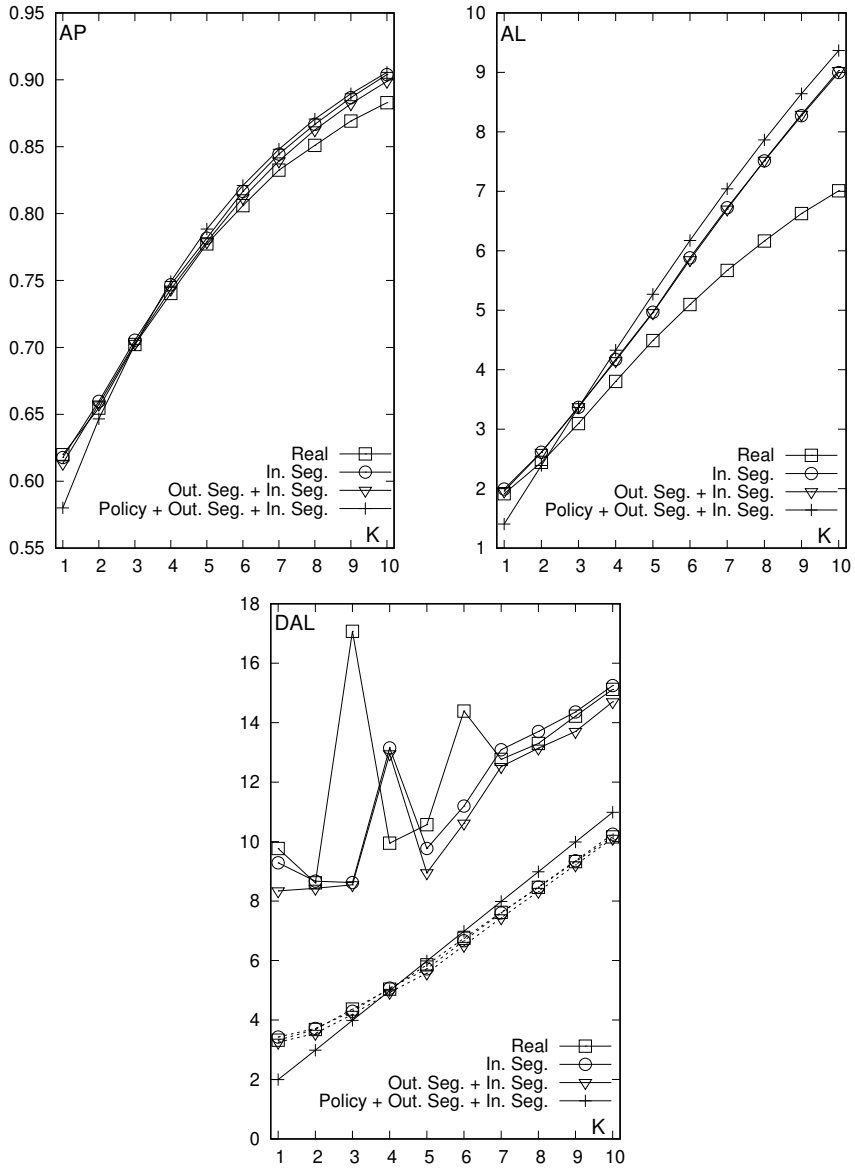

Figure 2.9: Stream-level AP (top-left), AL (top-right) and DAL (bottom) with $s = 1.0$ and $s = 0.95$ (dashed lines) as a function of k in the multi-k approach for four experimental setups on the IWSLT 2010 German-English dev set.

result of compensating for re-segmentation errors. Moreover, the gap between "In. Seg." and "In. Seg. + Out. Seg." is not significant, therefore we believe that, if the translation quality is good enough, the automatic re-segmentation process is an acceptable way of computing stream-level latencies. Lastly, as expected, if we use an oracle system that outputs the reference translation with the appropriate writing rate for each sentence ("Policy + In. Seg. + Out. Seg."), the obtained AL and DAL values are very close to the theoretical value k. If we compute DAL using $s = 0.95$, we obtain similar values without the need of using any oracle, while accounting for the additional cost of write operations.

Thus, unlike the Concat-1 approach, our stream-level approach is highly effective for providing interpretable and accurate latency measures.

4.5 Conclusions

In this work, an adaptation of the current latency measures to a streaming setup is proposed motivated by the lack of interpretability of sentence-level latency measures in this setup.

This adaptation basically consists in the modification of the conventional latency measures to move from a sentence-level evaluation based on a local delay function to a stream-level estimation by using a global delay function that keeps track of delays across the whole translation process. At the same time, a re-segmentation approach has been proposed to compute these latency measures on any arbitrary segmentation of the input stream used by the translation model. The resulting measures are highly interpretable and coherent accounting for the actual behaviour of the simultaneous translation system in a real streaming scenario.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 761758 (X5Gon) and 952215 (TAILOR) and Erasmus+ Education program under grant agreement no. 20-226-093604-SCH; the Government of Spain's research project Multisub, ref. RTI2018-094879-B-I00 (MCIU/AEI/FEDER,EU) and FPU scholarships FPU18/04135; and the Generalitat Valenciana's research project Classroom Activity Recognition, ref. PROMETEO/2019/111.

Table 2.33: Stream-level AL as a function of k, computed using the Concat-1 approach on the IWSLT 2010 German-English dev set.

System	1	2	3	4	5	6	7	8	9	10
Real	-9.7	-12.0	-45.2	-23.7	-8.5	-4.4	-17.4	-13.6	-14.2	-12.2
+In-Seg.	-42.9	-29.0	17.4	-10.1	25.5	3.8	9.7	5.3	2.7	4.7
+Policy	14.2	15.1	16.0	16.8	17.6	18.2	18.9	19.5	20.1	20.6

Table 2.34: Stream-level DAL as a function of k, computed using the Concat-1 approach on the IWSLT 2010 German-English dev set.

System	1	2	3	4	5	6	7	8	9	10
Real	15.0	11.0	17.4	11.3	20.3	25.1	11.3	14.4	17.7	17.9
+In-Seg.	4.5	8.5	37.1	24.6	52.3	27.8	21.5	33.9	31.2	31.8
+Policy	85.8	86.7	87.7	88.7	89.7	90.7	91.7	92.7	93.6	94.6

4.6 Reproducibility of proposed measures

The code for the proposed latency measures, as well as all the translations have been published [5]. A script is included to reproduce the results reported in the paper. The full results for the Concat-1 method are reported on Tables 2.33 and 2.34

Table 2.35: Corpus used for MT model training

		tokens(M)	
Corpus	sentences(M)	German	English
News Commentary	0.3	7.4	7.2
WikiTitles	1.3	2.7	3.1
Europarl	1.8	42.5	45.5
Rapid	1.5	26.0	26.9
MuST-C	0.2	3.9	4.2
Ted	0.2	3.3	3.6
LibriVox	0.1	0.9	1.1
Paracrawl	31.4	465.2	502.9

4.7 MT System

Table 2.35 lists the corpus that were selected for training out of the IWSLT 2020 allowed data [6].

The multi-k system has been trained with the official implementation [7]. The model was trained for 0.5M steps on a machine with 2 2080Ti GPUs, which took 6 days. The following command was used to train it:

```
fairseq-train $CORPUS_FOLDER \
-s $SOURCE_LANG_SUFFIX \
-t $TARGET_LANG_SUFFIX \
--user-dir $FAIRSEQ/examples/waitk \
--arch waitk_transformer_base \
--share-decoder-input-output-embed \
--left-pad-source False \
--multi-waitk \
--optimizer adam \
--adam-betas '(0.9, 0.98)' \
--clip-norm 0.0 \
--lr-scheduler inverse_sqrt \
--warmup-init-lr 1e-07 \
--warmup-updates 4000 \
--lr 0.0005 \
--min-lr 1e-09 \
--dropout 0.3 \
--weight-decay 0.0 \
--criterion label_smoothed_cross_entropy \
--label-smoothing 0.1 \
--max-tokens 4000 \
--update-freq 4 \
--save-dir $MODEL_OUTPUT_FOLDER \
--no-progress-bar \
--log-interval 100 \
--max-update 500000 \
--save-interval-updates 10000 \
--keep-interval-updates 20 \
--ddp-backend=no_c10d \
--fp16
```

[5] https://github.com/jairsan/Stream-level_Latency_Evaluation_for_Simultaneous_Machine_Translation

[6] http://iwslt2020.ira.uka.de/doku.php?id=offline_speech_translation

[7] https://github.com/elbayadm/attn2d

4.8 Segmenter System

The Direct Segmentation system has been trained with the official implementation [8]. The ted corpus was used as training data (See Table 2.35). The following command was used to train the segmenter system:

```
len=11
window=0
python3 train_text_model.py \
--train_corpus train.ML$len.WS$window.txt \
--dev_corpus  dev.ML$len.WS$window.txt \
--output_folder $output_folder \
--vocabulary $corpus_folder/train.vocab.txt \
--checkpoint_interval 1 \
--epochs 15 \
--rnn_layer_size 256 \
--embedding_size 256 \
--n_classes 2 \
--batch_size 256 \
--min_split_samples_batch_ratio 0.3 \
--optimizer adam \
--lr 0.0001 \
--lr_schedule reduce_on_plateau \
--lr_reduce_patience 5 \
--dropout 0.3 \
--model_architecture ff-text \
--feedforward_layers 2 \
--feedforward_size 128 \
--sample_max_len $len \
--sample_window_size $window
```

References

Ansari, Ebrahim et al. (2020). "FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN". In: *Proc. of IWSLT*. Online: ACL, pp. 1–34 (cit. on pp. 121, 127).

Bahar, Parnia et al. (2020). "Start-Before-End and End-to-End: Neural Speech Translation by AppTek and RWTH Aachen University". In: *Proc. of IWSLT* (cit. on p. 121).

Cettolo, Mauro, Christian Girardi, and Marcello Federico (2012). "WIT^3: Web Inventory of Transcribed and Translated Talks". In: *Proc. of EAMT*, pp. 261–268 (cit. on p. 127).

[8] https://github.com/jairsan/Speech_Translation_Segmenter

Cherry, Colin and George Foster (2019). "Thinking Slow about Latency Evaluation for Simultaneous Machine Translation". In: *arXiv:1906.00048* (cit. on p. 121).

Cho, Kyunghyun and Masha Esipova (2016). "Can neural machine translation do simultaneous translation?" In: *arXiv preprint arXiv:1606.02012* (cit. on p. 121).

Elbayad, Maha, Laurent Besacier, and Jakob Verbeek (2020). "Efficient Wait-k Models for Simultaneous Machine Translation". In: *Proc. of Interspeech*, pp. 1461–1465 (cit. on p. 127).

Elbayad, Maha, Ha Nguyen, et al. (2020). "ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020". In: *Proc. of IWSLT*. ACL, pp. 35–43 (cit. on p. 121).

Han, Hou Jeung et al. (2020). "End-to-End Simultaneous Translation System for IWSLT2020 Using Modality Agnostic Meta-Learning". In: *Proc. of IWSLT*. ACL, pp. 62–68 (cit. on p. 121).

Iranzo-Sánchez, Javier et al. (2020). "Direct Segmentation Models for Streaming Speech Translation". In: *Proc. of EMNLP*, pp. 2599–2611 (cit. on p. 127).

Ma, Mingbo et al. (2019). "STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework". In: *Proc. of ACL*. ACL, pp. 3025–3036. DOI: 10.18653/v1/P19-1289 (cit. on p. 121).

Matusov, Evgeny et al. (2005). "Evaluating machine translation output with automatic sentence segmentation". In: *Proc. of IWSLT*. ISCA (cit. on p. 126).

Paul, Michael, Marcello Federico, and Sebastian Stüker (2010). "Overview of the IWSLT 2010 evaluation campaign". In: *Proc. of IWSLT*. ISCA, pp. 3–27 (cit. on p. 127).

Pham, Ngoc-Quan et al. (2020). "KIT's IWSLT 2020 SLT Translation System". In: *Proc. of IWSLT*. ACL, pp. 55–61 (cit. on p. 121).

Schneider, Felix and Alexander Waibel (2020). "Towards Stream Translation: Adaptive Computation Time for Simultaneous Machine Translation". In: *Proc. of IWSLT*. ACL, pp. 228–236 (cit. on pp. 122, 123).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS*. Ed. by Isabelle Guyon et al., pp. 5998–6008 (cit. on p. 127).

5 From Simultaneous to Streaming Machine Translation by Leveraging Streaming History

Javier Iranzo-Sánchez, Jorge Civera, Alfons Juan

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6972-6985

Dublin (Ireland)

10.18653/v1/2022.acl-long.480

22–27 May 2022

From Simultaneous to Streaming Machine Translation by Leveraging Streaming History

Javier Iranzo-Sánchez, Jorge Civera, Alfons Juan

Abstract

Simultaneous Machine Translation is the task of incrementally translating an input sentence before it is fully available. Currently, simultaneous translation is carried out by translating each sentence independently of the previously translated text. More generally, Streaming MT can be understood as an extension of Simultaneous MT to the incremental translation of a continuous input text stream. In this work, a state-of-the-art simultaneous sentence-level MT system is extended to the streaming setup by leveraging the streaming history. Extensive empirical results are reported on IWSLT Translation Tasks, showing that leveraging the streaming history leads to significant quality gains. In particular, the proposed system proves to compare favorably to the best performing systems.

5.1 Introduction

Simultaneous Machine Translation (MT) is the task of incrementally translating an input sentence before it is fully available. Indeed, simultaneous MT can be naturally understood in the scenario of translating a text stream as a result of an upstream Automatic Speech Recognition (ASR) process. This setup defines a simultaneous Speech Translation (ST) scenario that is gaining momentum due to the vast number of industry applications that could be exploited based on this technology, from person-to-person communication to subtitling of audiovisual content, just to mention two main applications.

These real-world streaming applications motivate us to move from simultaneous to streaming MT, understanding streaming MT as the task of simultaneously translating a potentially unbounded and unsegmented text stream. Streaming MT poses two main additional challenges over simultaneous MT.

First, the MT system must be able to leverage the streaming history beyond the sentence level both at training and inference time. Second, the system must work under latency constraints over the entire stream.

With regard to exploiting streaming history, or more generally sentence context, it is worth mentioning the significant amount of previous work in offline MT at sentence level (Tiedemann and Scherrer 2017; Agrawal, Turchi, and Negri 2018), document level (Scherrer, Tiedemann, and Loáiciga 2019; S. Ma, D. Zhang, and Zhou 2020; Z. Zheng et al. 2020; Li et al. 2020; Maruf, Saleh, and Haffari 2021; B. Zhang et al. 2021), and in related areas such as language modelling (Dai et al. 2019) that has proved to lead to quality gains. Also, as reported in (Li et al. 2020), more robust ST systems can be trained by taking advantage of the context across sentence boundaries using a data augmentation strategy similar to the prefix training methods proposed in (Niehues et al. 2018; M. Ma et al. 2019). This data augmentation strategy was suspected to boost re-translation performance when compared to conventional simultaneous MT systems (Arivazhagan, Cherry, Macherey, and Foster 2020).

Nonetheless, with the notable exception of (Schneider and Alexander Waibel 2020), sentences in simultaneous MT are still translated independently from each other ignoring the streaming history. (Schneider and Alexander Waibel 2020) proposed an end-to-end streaming MT model with a Transformer architecture based on an Adaptive Computation Time method with a monotonic encoder-decoder attention. This model successfully uses the streaming history and a relative attention mechanism inspired by Transformer-XL (Dai et al. 2019). Indeed, this is an MT model that sequentially translates the input stream without the need for a segmentation model. However, it is hard to interpret the latency of their streaming MT model because the authors observe that the current sentence-level latency measures, Average Proportion (AP) (K. Cho and Esipova 2016), Average Lagging (AL) (M. Ma et al. 2019) and Differentiable Average Lagging (DAL) (Cherry and Foster 2019) do not perform well on a streaming setup. This fact is closely related to the second challenge mentioned above, which is that the system must work under latency constraints over the entire stream. Indeed, current sentence-level latency measures do not allow us to appropriately gauge the latency of streaming MT systems. To this purpose, (Iranzo-Sánchez, Civera Saiz, and Juan 2021) recently proposed a stream-level adaptation of the sentence-level latency measures based on the conventional re-segmentation approach applied to the ST output in order to evaluate translation quality (Matusov et al. 2005).

In this work, the simultaneous MT model based on a unidirectional encoder-decoder and training along multiple wait-k paths proposed by (Elbayad, Be-

sacier, and Verbeek 2020) is evolved into a streaming-ready simultaneous MT model. To achieve this, model training is performed following a sentence-boundary sliding-window strategy over the parallel stream that exploits the idea of prefix training, while inference is carried out in a single forward pass on the source stream that is segmented by a Direct Segmentation (DS) model (Iranzo-Sánchez et al. 2020). In addition, a refinement of the unidirectional encoder-decoder that takes advantage of longer context for encoding the initial positions of the streaming MT process is proposed. This streaming MT system is thoroughly assessed on IWSLT translation tasks to show how leveraging the streaming history provides systematic and significant BLEU improvements over the baseline, while reported stream-adapted latency measures are fully consistent and interpretable. Finally, our system favourably compares in terms of translation quality and latency to the latest state-of-the-art simultaneous MT systems (Ansari et al. 2020).

This paper is organized as follows. Next section provides a formal framework for streaming MT to accommodate streaming history in simultaneous MT. Section 5.3 presents the streaming experimental setup whose results are reported and discussed in Section 5.4. Finally, conclusions and future work are drawn in Section 5.5.

5.2 Streaming MT

In streaming MT, the source stream X to be translated into Y comes as an unsegmented and unbounded sequence of tokens. In this setup, the decoding process usually takes the greedy decision of which token appears next at the i-th position of the translation being generated

$$\hat{Y}_i = \arg\max_{y \in \mathcal{Y}} p\Big(y \mid X_1^{G(i)}, Y_1^{i-1}\Big) \tag{2.30}$$

where $G(i)$ is a global delay function that tells us the last position in the source stream that was available when the i-th target token was output, and $\mathcal{Y}$ is the target vocabulary.

However, taking into account the entire source and target streams can be prohibitive from a computational viewpoint, so the generation of the next token can be conditioned to the last $H(i)$ tokens of the stream as

$$\hat{Y}_i = \arg\max_{y \in \mathcal{Y}} p\Big(y \mid X_{G(i)-H(i)+1}^{G(i)}, Y_{i-H(i)}^{i-1}\Big)\,. \tag{2.31}$$

Nevertheless, for practical purposes, the concept of sentence segmentation is usually introduced to explicitly indicate a monotonic alignment between source and target sentences in streaming MT. Let us consider for this purpose the random variables **a** and **b** for the source and target segmentation of the stream, respectively. Variables **a** and **b** can be understood as two vectors of equal length denoting that the n-th source sentence starts at position a_n, while the n-th target sentence does so at position b_n.

In the next sections, we reformulate simultaneous MT in terms of the more general framework of streaming MT. This reformulation allows us to consider opportunities for improvement of previous simultaneous MT models.

Simultaneous MT with streaming history

In the conventional simultaneous MT setup, the aforementioned variables **a** and **b** are uncovered at training and inference time, while in streaming MT **a** and **b** are considered hidden variables at inference time that may be uncovered by a segmentation model. In fact, in conventional simultaneous MT the history is limited to the current sentence being translated, while in streaming MT we could exploit the fact that the history could potentially span over all the previous tokens before the current sentence.

To this purpose, the global delay function $G(i)$ introduced above would replace the sentence-level delay function $g(i)$ commonly used in simultaneous MT. However, it should be noticed that we could express $g(i)$ as $G(i) - a_n$ with $b_n \leq i < b_{n+1}$. Delay functions are defined as a result of the policy being applied. This policy decides what action to take at each timestep, whether to read a token from the input or to write a target token. Policies can be either fixed (M. Ma et al. 2019; Dalvi et al. 2018) depending only on the current timestep, or adaptive (Arivazhagan, Cherry, Macherey, Chiu, et al. 2019; X. Ma et al. 2020; B. Zheng et al. 2020) being also conditioned on the available input source words. Among those fixed policies, the sentence-level wait-k policy proposed by (M. Ma et al. 2019) is widely used in simultaneous MT with the simple local delay function

$$g(i) = k + i - 1. \tag{2.32}$$

This policy initially reads k source tokens without writing a target token, and then outputs a target token every time a source token is read. This is true in the case that the ratio between the source and target sentence lengths is one.

However, in the general case, a catch-up factor γ computed as the inverse of the source-target length ratio defines how many target tokens are written for every read token, that generalises Eq. 2.32 as

$$g(i) = \left\lfloor k + \frac{i-1}{\gamma} \right\rfloor . \tag{2.33}$$

The wait-k policy can be reformulated in streaming MT so that the wait-k behaviour is carried out for each sentence as

$$G(i) = \left\lfloor k + \frac{i-b_n}{\gamma} \right\rfloor + a_n - 1 \tag{2.34}$$

where $b_n \leq i < b_{n+1}$.

In streaming MT, we could take advantage of the streaming history by learning the probability distribution stated in Eq. 2.31, whenever streaming samples would be available. However, training such a model with arbitrarily long streaming samples poses a series of challenges that need to be addressed. Firstly, it would be necessary to carefully define $G(i)$ and $H(i)$ functions so that, at each timestep, the available source and target streams are perfectly aligned. Given that the source-target length ratio may vary over the stream, if one uses a wait-k policy with a fixed γ, there is a significant chance that source and target are misaligned at some points over the stream. Secondly, every target token can potentially have a different $G(i)$ and $H(i)$, so the encoder-decoder representation and contribution to the loss would need to be recomputed for each target token at a significant computational expense. Lastly, current MT architectures and training procedures have evolved conditioned by the availability of sentence-level parallel corpora for training, so they need to be adapted to learn from parallel streams.

To tackle the aforementioned challenges in streaming MT, a compromise practical solution is to uncover the source and target sentence segmentations. At training time, parallel samples are extracted by a sentence-boundary sliding window spanning over several sentences of the stream that shifts to the right one sentence at a time. In other words, each sentence pair is concatenated with its corresponding streaming history that includes previous sentence pairs simulating long-span prefix training. Doing so, we ensure that source and target streams are properly aligned at all times, and training can be efficiently carried out by considering a limited history. The inference process is performed in a purely streaming fashion in a single forward pass as defined in Eq. 2.31 with

$H(i)$ being consistently defined in line with training, so that the streaming history spans over previous sentences already translated.

Partial Bidirectional Encoder

In simultaneous MT, the conventional Transformer-based bidirectional encoder representation (of the l-th layer) of a source token at any position j is constrained to the current n-th sentence

$$e_j^{(l)} = \text{Enc}\left(e_{a_n:G(i)}^{(l-1)}\right) \tag{2.35}$$

where $a_n \leq j \leq G(i)$, while the decoder can only attend to previous target words and the encoding of those source words that are available at each timestep

$$s_i^{(l)} = \text{Dec}\left(s_{b_n:i-1}^{(l-1)}, e_{a_n:G(i)}^{(l-1)}\right). \tag{2.36}$$

As a result, the encoder and decoder representations for positions j and i, respectively, could be computed taking advantage of subsequent positions to position j up to position $G(i)$ at inference time. However, at training time, this means that this bidirectional encoding-decoding of the source sentence has to be computed for every timestep, taking up to $|y|$ times longer than the conventional Transformer model.

To alleviate this problem, (Elbayad, Besacier, and Verbeek 2020) proposes a wait-k simultaneous MT model based on a modification of the Transformer architecture that uses unidirectional encoders and multiple values of k at training time. In this way, the model is consistent with the limited-input restriction of simultaneous MT at inference time. The proposed unidirectional encoder can be stated as

$$e_j^{(l)} = \text{Enc}\left(e_{a_n:j}^{(l-1)}\right), \tag{2.37}$$

that is more restrictive than that in Eq. 2.35, and it consequently conditions the decoder representation, since $G(i)$ in Eq. 2.36 depends on the specific k value employed at each training step.

As mentioned above, the unidirectional encoder just requires a single forward pass of the encoder at training time, and therefore there is no additional computational cost compared with a conventional Transformer. However, it does not take into account all possible input tokens for different values of k. Indeed, the encoding of the j-th input token will not consider those tokens beyond the j-th position, even if including them into the encoding process does not prevent us from performing a single forward pass.

A trade-off between the unidirectional and bidirectional encoders is what we have dubbed Partial Bidirectional Encoder (PBE), which modifies the unidirectional encoder to allow the first $k-1$ source positions to have access to succeeding tokens according to

$$e_j^{(l)} = \text{Enc}\left(e_{a_n:\max(a_n+k-1,j)}^{(l-1)}\right). \tag{2.38}$$

PBE allows for a longer context when encoding the initial positions and is consistent with Eq. 2.36. At training time a single forward pass of the encoder-decoder is still possible as in the unidirectional encoder, and therefore no additional training cost is incurred. At inference time, we fall back to the bidirectional encoder.

Figure 2.10 shows a graphical comparison of the attention mechanism in $j = 3$ across the bidirectional (top-left), unidirectional (top-right) and PBE (bottom) encoders with $k = 4$ for two consecutive timesteps $i = 1$ with $G(1) = 4$ and $i = 2$ with $G(2) = 5$. As observed, PBE can take advantage of additional positions from $j+1$ up to k with respect to the unidirectional encoder.

In a streaming setup, the bidirectional encoder-decoder of Eqs. 2.35 and 2.36 are not necessarily constrained to the current sentence and could exploit a streaming history of $H(i)$ tokens

$$e_j^{(l)} = \text{Enc}\left(e_{G(i)-H(i)+1:G(i)}^{(l-1)}\right) \tag{2.39}$$

$$s_i^{(l)} = \text{Dec}\left(s_{i-H(i):i-1}^{(l-1)}, e_{G(i)-H(i)+1:G(i)}^{(l-1)}\right). \tag{2.40}$$

Likewise, the proposed PBE with streaming history states as follows

$$e_j^{(l)} = \text{Enc}\left(e_{G(i)-H(i)+1:\max(G(i)-H(i)+k,j)}^{(l-1)}\right). \tag{2.41}$$

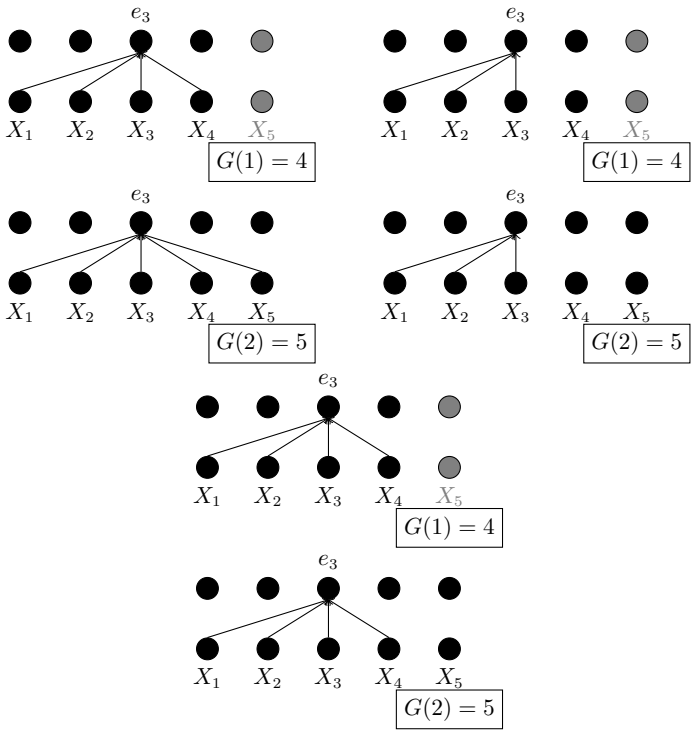

Figure 2.10: Comparison of attention positions in $j = 3$ for bidirectional (top-left), unidirectional (top-right) and PBE (bottom) encoders with $k = 4$ in two consecutive timesteps $i = 1$ with $G(1) = 4$ and $i = 2$ with $G(2) = 5$.

5.3 Experimental setup

Table 2.36: Basic statistics of the training data from the IWSLT 2020 Evaluation Campaign (M = Millions).

Corpus	Doc	Sents(M)	Tokens(M)	
			German	English
News-Comm.	✓	0.3	7.4	7.2
Wikititles		1.3	2.7	3.1
Europarl	✓	1.8	42.5	45.5
Rapid	✓	1.5	26.0	26.9
MuST-C	✓	0.2	3.9	4.2
TED	✓	0.2	3.3	3.6
LibriVox		0.1	0.9	1.1
Paracrawl		31.4	465.2	502.9

A series of comparative experiments in terms of translation quality and latency have been carried out using data from the IWSLT 2020 Evaluation Campaign (Ansari et al. 2020), for both German→English and English→German. For the streaming condition, our system is tuned on the 2010 dev set, and evaluated on the 2010 test set for comparison with (Schneider and Alexander Waibel 2020). Under this setting, words were lowercased and punctuation was removed in order to simulate a basic upstream ASR system. Also, a second non-streaming setting is used for the English→German direction to compare our system with top-of-the-line sentence-based simultaneous MT systems participating in the IWSLT 2020 Simultaneous Translation Task.

Table 2.36 summarizes the basic statistics of the IWSLT corpora used for training the streaming MT systems. Corpora for which document information is readily available are processed for training using the sliding window technique mentioned in Section 5.2. Specifically, for each training sentence, we prepend previous sentences, which are added one by one until a threshold h of history tokens is reached. Sentence boundaries are defined on the presence of special tokens (`[DOC]`,`[CONT]`,`[BRK]`,`[SEP]`) as in (Junczys-Dowmunt 2019). Byte Pair Encoding (Sennrich, Haddow, and Birch 2016) with 40K merge operations is applied to the data after preprocessing.

Our streaming MT system is evaluated in terms of latency and translation quality with BLEU (Papineni et al. 2002). Traditionally, latency evaluation in simultaneous MT has been carried out using AP, AL and DAL. However, these measures have been devised for sentence-level evaluation, where the latency of every sentence is computed independently from each other and as mentioned before, they do not perform well on a streaming setup. Thus, we revert to the stream-based adaptation of these measures proposed in (Iranzo-Sánchez, Civera Saiz, and Juan 2021) unless stated otherwise.

Latency measures for a sentence pair $(\boldsymbol{x}, \boldsymbol{y})$ are based on a cost function $C_i(\boldsymbol{x}, \boldsymbol{y})$ and a normalization term $Z(\boldsymbol{x}, \boldsymbol{y})$

$$L(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{y})} \sum_i C_i(\boldsymbol{x}, \boldsymbol{y}) \tag{2.42}$$

where

$$C_i(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} g(i) & \text{AP} \\ g(i) - \frac{i-1}{\gamma} & \text{AL} \\ g'(i) - \frac{i-1}{\gamma} & \text{DAL} \end{cases} \tag{2.43}$$

and

$$Z(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} |\boldsymbol{x}| \cdot |\boldsymbol{y}| & \text{AP} \\ \underset{i:g(i)=|\boldsymbol{x}|}{\arg\min}\ i & \text{AL} \\ |\boldsymbol{y}| & \text{DAL} \end{cases} \tag{2.44}$$

Latency measures can be computed in a streaming manner by considering a global delay function $G(i)$, that is mapped into a relative delay so that it can be compared with the sentence-level oracle delay. For the i-th target position of the n-th sentence, the associated relative delay can be obtained from the global delay function as $g_n(i) = G(i + b_n) - a_n$. So, the stream-adapted cost function of the latency measures is defined as

$$C_i(\boldsymbol{x}_n, \boldsymbol{y}_n) = \begin{cases} g_n(i) & \text{AP} \\ g_n(i) - \frac{i-1}{\gamma_n} & \text{AL} \\ g'_n(i) - \frac{i-1}{\gamma_n} & \text{DAL} \end{cases} \tag{2.45}$$

with $g'_n(i)$ defined as

$$\max \begin{cases} g_n(i) & \\ g'_{n-1}(|\boldsymbol{x}_{n-1}|) + \frac{1}{\gamma_{n-1}} & i = 1 \\ g'_n(i-1) + \frac{1}{\gamma_n} & i > 1 \end{cases} \tag{2.46}$$

This definition assumes that the source and target sentence segmentation of the stream are uncovered, but this is not always the case (Schneider and Alexander Waibel 2020) or they may not match that of the reference translations. However, sentence boundaries can be obtained by re-segmenting the system hypothesis following exactly the same procedure applied to compute translation quality in ST evaluation. To this purpose, we use the MWER segmenter (Matusov et al. 2005) to compute sentence boundaries according to the reference translations.

Our streaming MT models have been trained following the conventional Transformer BASE (German↔English streaming MT) and BIG (English→German simultaneous MT) configurations (Vaswani et al. 2017). As in (Schneider and Alexander Waibel 2020), after training is finished, the models are finetuned on the training set of MuST-C (Di Gangi et al. 2019).

The proposed model in Section 5.2 assumes that at inference time the source stream has been segmented into sentences. To this purpose, we opt for the text-based DS model (Iranzo-Sánchez et al. 2020), a sliding-window segmenter that moves over the source stream taking a split decision at each token based on a local-context window that extends to both past and future tokens. This segmenter is streaming-ready and obtains superior translation quality when compared with other segmenters (Stolcke 2002; E. Cho, Niehues, and Alex Waibel 2017). As the future window length of the DS segmenter conditions the latency of the streaming MT system, this length was adjusted to find a tradeoff between latency and translation quality. The DS segmenter was trained on the TED corpus (Cettolo, Girardi, and Federico 2012).

5.4 Evaluation

Figure 2.11 reports the evolution of BLEU scores on the German-English IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history lengths ($h = \{0, 20, 40, 60, 80\}$). We show results for the 3 encoders introduced previously. History lengths were selected taking into account that the average sentence length is 20 tokens. A history length of zero ($h = 0$) refers to the conventional sentence-level simultaneous MT model. The BLEU scores for the offline MT systems with a bidirectional encoder are also reported using horizontal lines, in order to serve as reference values. We report offline results for $h = 0$ and the best performing history configuration, $h = 60$. All systems used the reference segmentation during decoding.

As observed, BLEU scores of the simultaneous MT systems leveraging on the streaming history ($h > 0$) are systematically and notably higher than those of conventional sentence-based simultaneous MT system ($h = 0$) over the range of wait-k values. Indeed, as the streaming history increases, BLEU scores also do reaching what it seems the optimal history length at $h = 60$ and slightly degrading at $h = 80$. As expected, when replacing the unidirectional encoder by the PBE, BLEU scores improve as the wait-k value increases, since PBE has additional access to those tokens from $j + 1$ up to k. For instance, for $k = 32$ and $h = 60$, PBE is 0.7 BLEU points above the unidirectional encoder. On the other hand, it can be observed how using an encoder which is not fully

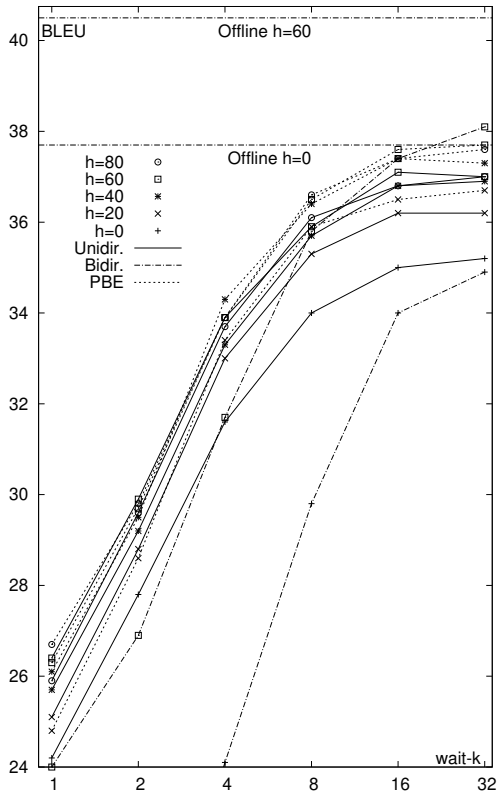

Figure 2.11: BLEU scores on the German-English IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history (h) lengths and encoder type (See Appendix 5.6 for a close-up).

bidirectional during training, creates a performance gap with respect to the offline bidirectional model when carrying out inference in an offline manner ($k \geq 32$). It can be also observed how the PBE model is better prepared for this scenario and shows a smaller gap. It is important to keep in mind that although both offline and PBE models behave the same way during inference for a large enough k, during training time the PBE model, trained using the multi-k with k randomly sampled for each batch, has been optimized jointly for low, medium and high latencies.

In general, the bidirectional encoder shows poor performance for simultaneous MT. This can be explained by the fact that there exists a mismatch between the training condition (whole source available) and the inference condition (only a

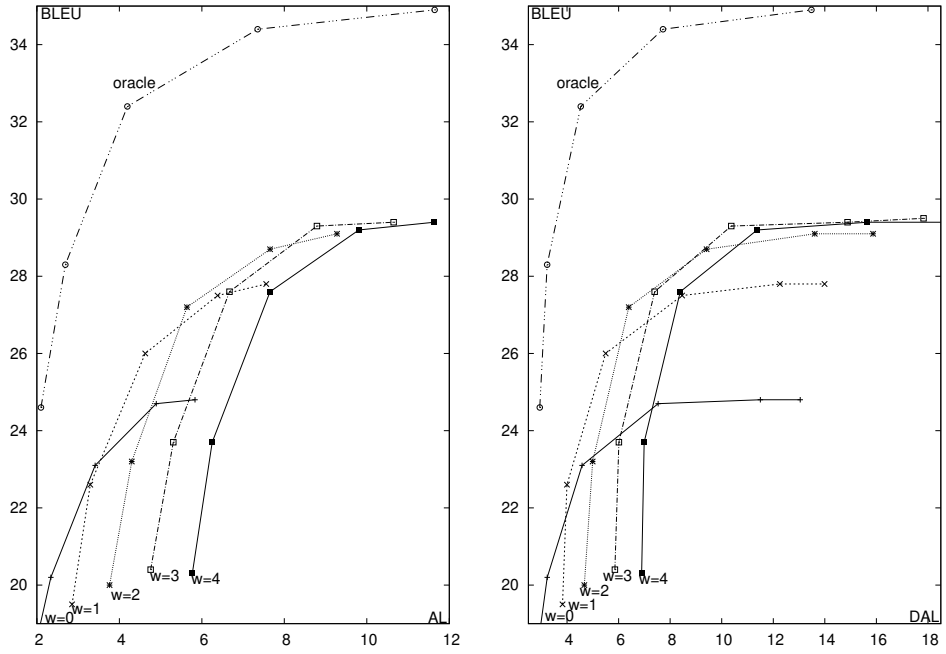

Figure 2.12: BLEU scores versus stream-adapted AL and DAL (scale s=0.85) with segmenters of future window length $w = \{0, 1, 2, 3, 4\}$ on the IWSLT 2010 test set. Points over each curve correspond to $k = \{1, 2, 4, 8, 16\}$ values of the wait-k policy used at inference time.

prefix of the source is available for $k < 32$). These results are consistent with (Elbayad, Besacier, and Verbeek 2020). Keep in mind that this bidirectional model is different from the offline one because it has been subject to the constraints of Eq. 2.36 during training. As a result of the BLEU scores reported in Figure 2.11, the streaming MT system with $h = 60$ and PBE was used in the rest of the German-English experiments.

Following (Schneider and Alexander Waibel 2020)'s setup, the test set is lowercased and concatenated into a single stream. In order to measure the latency of the pipeline defined by the segmenter followed by MT system, it is necessary to take into account not only the latency of the MT system but also that of the segmenter. Thankfully this is straightforward to do in our pipeline, as a segmenter with a future window of length w modifies the pipeline policy so that, at the start of the stream, w READ actions are carried out to fill up the future window. Then, every time the MT system carries out a READ action,

it receives one token from the segmenter. Thus, the integration of the segmenter into the pipeline is transparent from a latency viewpoint. Figure 2.12 shows BLEU scores versus stream-adapted AL and DAL (s scale = 0.85) figures reported with segmenters of future window length $w = \{0, 1, 2, 3, 4\}$ for a streaming evaluation on the IWSLT 2010 test set. Points over each curve correspond to $k = \{1, 2, 4, 8, 16\}$ values of the wait-k policy used at inference time. Results for a $w = 0$ oracle are also shown as an upper-bound.

As shown, stream-adapted AL and DAL figures achieved by our streaming MT system are reasonable, lagging 2-10 tokens behind the speaker for nearly maximum BLEU scores with a best BLEU score of 29.5 points. The same happens with AP figures ranging from 0.6 for $w = 0$ to 1.3 for $w = 4$. These figures highlight the advantages of tying together our translation policy with the sentence segmentation provided by the DS model. Every time the DS model emits an end-of-sentence event, the MT model is forced to catch-up and translate the entire input. In this way, the MT model never strays too far from the speaker, even if the source-target length ratio differs from the γ defined at inference time. See Appendix 5.6 for streaming translation results in the reverse direction (English $\rightarrow$ German).

Next, we compare our proposed streaming MT (STR-MT) model with the $\lambda = 0.3$ ACT system (Schneider and Alexander Waibel 2020) in terms of BLEU score and stream-adapted latency measures on Table 2.37. Stream-level AL and DAL indicate that the ACT models lags around 100 tokens behind the speaker. Although both MT systems achieve similar translation quality levels, they do so at significantly different latencies, since the ACT model lacks a catch-up mechanism to synchronize and keep the pace of the speaker.

The STR-MT model is now compared on the English-German IWSLT 2020 simultaneous text-to-text track (Ansari et al. 2020) with other participants: RWTH (Bahar et al. 2020), KIT (Pham et al. 2020) and ON-TRAC (Elbayad, Nguyen, et al. 2020). This comparison is carried out in order to assess whether the proposed streaming MT system is competitive with highly optimized systems for a simultaneous MT task. Given that the test set of this track remains

Table 2.37: Latency and quality comparison of ACT (Schneider and Alexander Waibel 2020) and the proposed STR-MT on the IWSLT 2010 De-En test set.

Model	BLEU	AP	AL	DAL
ACT	30.3	10.3	100.1	101.8
STR-MT	29.5	1.2	11.2	17.8

blind, we use the results reported on the MuST-C corpus as a reference. In order to evaluate all systems under the same conditions, the reference segmentation of the MuST-C corpus is used instead of the DS model. Additionally, given that all other participants translate each sentence independently, the conventional sentence-level AL latency measure is reported. Figure 2.13 shows the comparison of BLEU scores versus AL measured in terms of detokenized tokens. As defined in the IWSLT text-to-text track, three AL regimes, low ($AL \leq 3$), medium ($3 < AL \leq 6$) and high ($6 < AL \leq 15$) were considered.

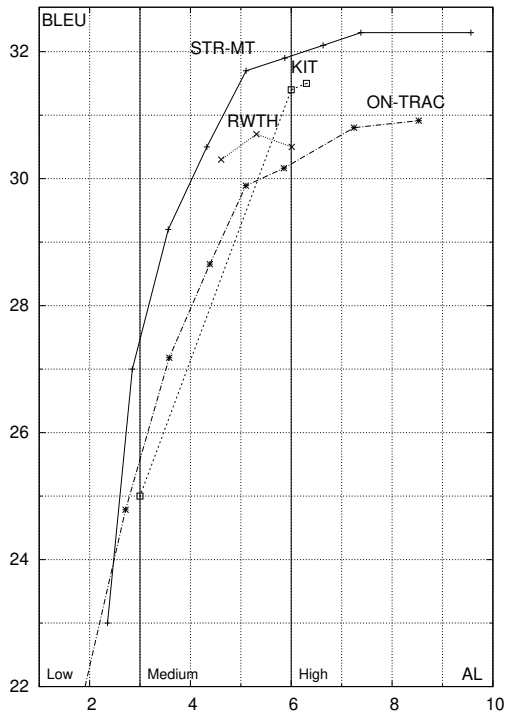

Figure 2.13: Comparative BLEU scores versus AL at three regimes, low, medium, and high latency, for IWSLT 2020 simultaneous text-to-text track participants, RWTH, ON-TRAC, KIT and our streaming MT (STR-MT) system on the MuST-C corpus.

ON-TRAC and our streaming MT system exhibit a similar progression, which is to be expected given that they are both based on the multi-k approach. However, our system consistently outperforms the ON-TRAC system by 1-2 BLEU. This confirms the importance of utilizing streaming history in order to significantly improve results, and how the proposed PBE model can take better advantage of the history.

RWTH and KIT systems are closer in translation quality to our proposal than ON-TRAC, for AL between 5 and 7. However, these systems do not show a flexible latency policy and are not comparable to our system at other regimes. Indeed, for that to be possible, these systems need to be re-trained, in contrast to our system in which latency is adjusted at inference time.

5.5 Conclusions

In this work, a formalization of streaming MT as a generalization of simultaneous MT has been proposed in order to define a theoretical framework in which our two contributions have been made. On the one hand, we successfully leverage streaming history across sentence boundaries for a simultaneous MT system based on multiple wait-k paths that allows our system to greatly improve the results of the sentence-level baseline. On the other hand, our PBE is able to take into account longer context information than its unidirectional counterpart, while keeping the same training efficiency.

Our proposed MT system has been evaluated under a realistic streaming setting being able to reach similar translation quality than a state-of-the-art segmentation-free streaming MT system at a fraction of its latency. Additionally, our system has been shown to be competitive when compared with state-of-the-art simultaneous MT systems optimized for sentence-level translation, obtaining excellent results using a single model across a wide range of latency levels, thanks to its flexible inference policy.

In terms of future work, additional training and inference procedures that take advantage of the streaming history in streaming MT are still open for research. One important avenue of improvement is to devise more robust training methods, so that simultaneous models can perform as well as their offline counterparts when carrying out inference at higher latencies. The segmentation model, though proved useful in a streaming setup, adds complexity and can greatly affect translation quality. Thus, the development of segmentation-free streaming MT models is another interesting research topic.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements no. 761758 (X5Gon) and 952215 (TAILOR), and Erasmus+ Education programme under grant agreement no. 20-226-093604-SCH (EXPERT); the Government of Spain's grant RTI2018-094879-B-I00 (Multisub) funded by MCIN/AEI/10.13039/501100011033 & "ERDF A way of making Europe", and FPU scholarships FPU18/04135; and the Generalitat Valenciana's research project Classroom Activity Recognition (ref. PROMETEO/2019/111). The authors gratefully acknowledge the computer resources at Artemisa, funded by the European Union ERDF and Comunitat Valenciana as well as the technical support provided by the Instituto de Física Corpuscular, IFIC (CSIC-UV).

5.6 Extended Streaming Translation Results

Figure 2.14 shows a close-up of Figure 2.11, which contains results for the German-English IWSLT 2010 dev set. We can observe how the PBE models obtain consistent quality improvements over their unidirectional counterparts.

Apart from the previously reported German → English streaming MT results, we have also conducted experiments in the reverse direction, English → German. These are shown in Figure 2.15. The results show a similar trend to previous experiments, with the addition of streaming history allowing our systems to obtain significant improvements over the sentence-based baseline. Unlike the previous case, the optimum history size in this case is $h = 40$ instead of $h = 60$.

In order to enable streaming translation, the best performing $h = 40$ systems has been combined with a German DS system. Similarly to previous experiments, we have conducted tests using different values of w and k in order to balance the latency-quality trade-off, shown in Figure 2.16. Under the streaming condition, the wait-k policy and DS model allow the model to follow closely the speaker while achieving good quality, with a latency that can be easily adjusted between 4 and 15 tokens depending on the requirements of the task. There are diminishing returns when increasing the latency above 6-7 tokens, as only marginal gains in quality are obtained.

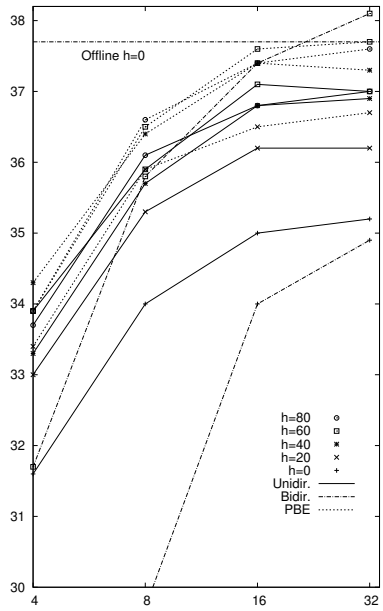

Figure 2.14: BLEU scores on the German-English IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history (h) lengths with a unidirectional encoder (solid lines), PBE (dashed line) or bidirectional (dashed line with points). This is a close-up of Figure 2.11.

5.7 Efficiency of the proposed models

During training of the unidirectional and PBE encoders, the constraints imposed by Eqs. 2.37 and 2.38 are efficiently implemented by full self-attention, as in the bidirectional encoder, followed by an attention mask, for each token to only attend those tokens fulfilling the constraints. The attention mask sets the weights of the other tokens to $-\infty$ before application of the self-attention softmax. This is exactly the same mechanism used in the standard Transformer decoder to prevent the auto-regressive decoder from accessing future information.

This means that the three encoder types have an identical computational behavior. We are not aware of alternative GPU-based acceleration techniques to speed up the training of the unidirectional encoder. If so, this could be also applicable to the training of the standard Transformer decoder.

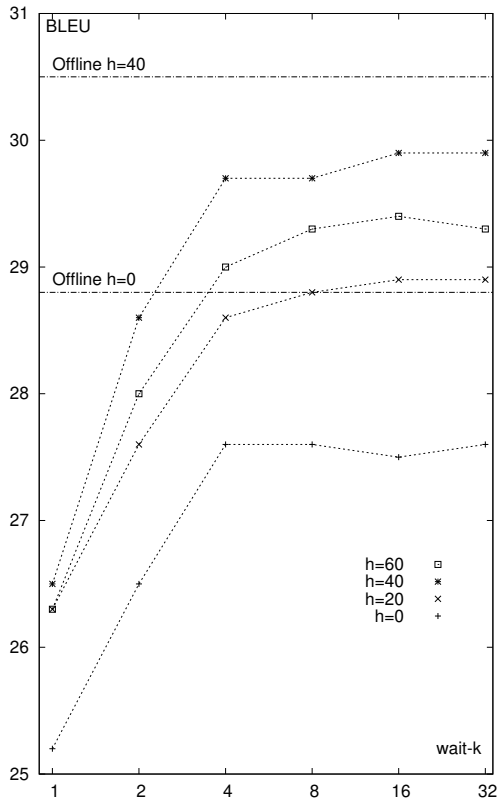

Figure 2.15: BLEU scores on the English-German IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history (h) lengths using a PBE encoder.

During inference time, however, the unidirectional encoder has some advantages. Given that the unidirectional encoder is incremental, meaning that the encodings of old tokens do not change when a new token becomes available, the process can be sped up by only computing the encoding of the newly available token. Although encoder self-attention still needs to be computed, a single vector is used as the query instead of the full matrix. Table 2.38 shows inference statistics for the different components of the En $\rightarrow$ De Transformer Big with h=60. Two setups have been tested: CPU-only inference, and GPU inference. Results were obtained on an Intel i9-7920X machine with an NVIDIA GTX 2080Ti.

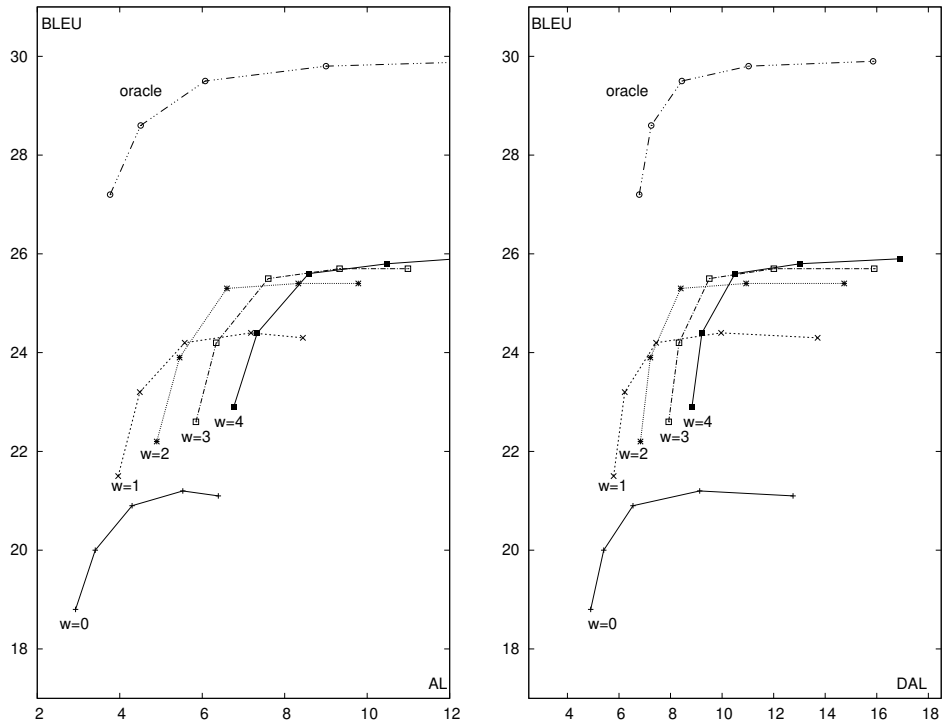

Figure 2.16: BLEU scores versus stream-adapted AL and DAL (scale s=0.85) with segmenters of future window length $w = \{0, 1, 2, 3, 4\}$ on the English-German IWSLT 2010 test set. Points over each curve correspond to $k = \{1, 2, 4, 8, 16\}$ values of the wait-k policy used at inference time.

The unidirectional encoder is four times faster than the bidirectional encoder when run on a CPU. However, both encoders perform the same when run on a GPU. For the streaming MT scenario considered in this work, no latency reduction is gained by not re-encoding previous tokens due to the GPU paralellization capability. When run on a GPU, the proposed model works seamlessly under real-time constraints.

Table 2.38: Latency of translating a token (in seconds) for the proposed En-De h=60 Transformer Big model.

Component	CPU	GPU
Unidir. Encoder	0.034s	0.002s
Bidir. Encoder	0.138s	0.002s
Decoder	0.242s	0.004s

5.8 MT System configuration

The multi-k systems have been trained with the official implementation (`https://github.com/elbayadm/attn2d`). Models are trained for 0.5M steps on a machine with 4 2080Ti GPUs. Total training time was 40h for BASE models, and 60h for BIG models. The following command was used to train them:

```
cri=label_smoothed_cross_entropy;
ex=simultaneous_translation
fairseq-train $CORPUS_FOLDER \
-s $SOURCE_LANG_SUFFIX \
-t $TARGET_LANG_SUFFIX \
--user-dir $FAIRSEQ/examples/$ex \
--arch $ARCH waitk_transformer_base \
--share-decoder-input-output-embed \
--left-pad-source False \
--multi-waitk \
--optimizer adam \
--adam-betas '(0.9, 0.98)' \
--clip-norm 0.0 \
--lr-scheduler inverse_sqrt \
--warmup-init-lr 1e-07 \
--warmup-updates 4000 \
--lr 0.0005 \
--min-lr 1e-09 \
--dropout 0.1 \
--weight-decay 0.0 \
--criterion $cri \
--label-smoothing 0.1 \
--max-tokens $TOK \
--update-freq 2 \
--save-dir $MODEL_OUTPUT_FOLDER \
--no-progress-bar \
--log-interval 100 \
--max-update 500000 \
--save-interval-updates 10000 \
```

```
  --keep-interval-updates 20 \
  --ddp-backend=no_c10d \
  --fp16
```

with

```
ARCH=waitk_transformer_base;
TOK=4000
```

for the BASE configuration, and

```
ARCH=waitk_transformer_big;
TOK=2000
```

for the BIG one.

For finetuning, we change to the following:

```
  --lr-scheduler fixed \
  --lr 4.47169e-05 \
```

For the streaming translation scenario, the data is lowercased and all punctuation signs are removed. For the simultaneous scenario (IWSLT 2020 simultaneous text- to-text), it is truecased and tokenized using Moses. We apply language identification to the training data using langid (Lui and Baldwin 2012) and discard those sentences that have been tagged with the wrong language. SentencePiece (Kudo and Richardson 2018) is used to learn the BPE units, and we use whitespace as a suffix in order to know when an entire target word has been written during decoding.

In order to obtain samples that can be used for training streaming MT models, a sliding window that moves over whole sentences is used to extract consistent source-target samples. Figure 2.17 shows an example of corpus construction using $h = 5$. The generated streaming data is upsampled to keep a 1-to-3 ratio with the regular sentence-level data.

Sentece pair	Source	Target
1	$x_{1,1}\ x_{1,2}$	$y_{1,1}\ y_{1,2}$
2	$x_{2,1}\ x_{2,2}\ x_{2,3}$	$y_{2,1}\ y_{2,2}$
3	$x_{3,1}\ x_{3,2}\ x_{3,3}$	$y_{3,1}\ y_{3,2}\ y_{3,3}$
4	$x_{4,1}\ x_{4,2}$	$y_{4,1}\ y_{4,2}$

Sentence pair	Source
1	[DOC] $x_{1,1}\ x_{1,2}$ [BRK]
2	[DOC] $x_{1,1}\ x_{1,2}$ [SEP] $x_{2,1}\ x_{2,2}\ x_{2,3}$ [BRK]
3	[DOC] $x_{1,1}\ x_{1,2}$ [SEP] $x_{2,1}\ x_{2,2}\ x_{2,3}$ [SEP] $x_{3,1}\ x_{3,2}\ x_{3,3}$ [BRK]
4	[CONT] $x_{3,1}\ x_{3,2}\ x_{3,3}$ [SEP] $x_{4,1}x_{4,2}$ [END]

Sentence pair	Target
1	[DOC] $y_{1,1}\ y_{1,2}$ [BRK]
2	[DOC] $y_{1,1}\ y_{1,2}$ [SEP] $y_{2,1}\ y_{2,2}$ [BRK]
3	[DOC] $y_{1,1}\ y_{1,2}$ [SEP] $y_{2,1}\ y_{2,2}$ [SEP] $y_{3,1}\ y_{3,2}\ y_{3,3}$ [BRK]
4	[CONT] $y_{3,1}\ y_{3,2}\ y_{3,3}$ [SEP] $y_{4,1}y_{4,2}$ [END]

Figure 2.17: Illustrated example of sample construction with history. Starting from a corpus of ordered sentence pairs (top), streaming samples are constructed (bottom) using $h = 5$. Past history is shown in light gray. Sentence boundary and document tokens (Junczys-Dowmunt 2019) are not counted for the history size limit. Notice how, for the last sample, the pair $(\boldsymbol{x}_2, \boldsymbol{y}_2)$ is not included in the sample, as the history size limit would have otherwise been exceeded on the source side.

5.9 Segmenter System configuration

The Direct Segmentation system has been trained with the official implementation (`https://github.com/jairsan/Speech_Translation_Segmenter`). The following command was used to train the segmenter system:

```
python3 train_text_model.py \
 --train_corpus train.$len_$window.txt \
 --dev_corpus  dev.$len_$window.txt \
 --output_folder $out_f \
 --vocabulary $corpus_f/train.vocab.txt \
 --checkpoint_interval 1 \
 --epochs 15 \
 --rnn_layer_size 256 \
 --embedding_size 256 \
 --n_classes 2 \
 --batch_size 256 \
 --min_split_samples_batch_ratio 0.3 \
 --optimizer adam \
 --lr 0.0001 \
 --lr_schedule reduce_on_plateau \
 --lr_reduce_patience 5 \
 --dropout 0.3 \
 --model_architecture ff-text \
 --feedforward_layers 2 \
 --feedforward_size 128 \
 --sample_max_len $len \
 --sample_window_size $window
```

with the following configurations:

```
(len=11; window=0)
(len=12; window=1)
(len=13; window=2)
(len=14, window=3)
(len=15, window=4)
```

References

Agrawal, Ruchit, M. Turchi, and M. Negri (2018). "Contextual Handling in Neural Machine Translation: Look Behind, Ahead and on Both Sides". In: *Proc. of EAMT*, pp. 11–20 (cit. on p. 138).

Ansari, Ebrahim et al. (2020). "FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN". In: *Proc. of IWSLT*. Online: ACL, pp. 1–34 (cit. on pp. 139, 145, 150).

Arivazhagan, Naveen, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, et al. (2019). "Monotonic Infinite Lookback Attention for Simultaneous Machine Translation". In: *Proc. of ACL*. ACL, pp. 1313–1323. DOI: 10.18653/v1/P19-1126 (cit. on p. 140).

Arivazhagan, Naveen, Colin Cherry, Wolfgang Macherey, and George Foster (2020). "Re-translation versus Streaming for Simultaneous Translation". In: *arXiv preprint arXiv:2004.03643* (cit. on p. 138).

Bahar, Parnia et al. (2020). "Start-Before-End and End-to-End: Neural Speech Translation by AppTek and RWTH Aachen University". In: *Proc. of IWSLT*. ACL, pp. 44–54 (cit. on p. 150).

Cettolo, Mauro, Christian Girardi, and Marcello Federico (2012). "WIT^3: Web Inventory of Transcribed and Translated Talks". In: *Proc. of EAMT*, pp. 261–268 (cit. on p. 147).

Cherry, Colin and George Foster (2019). "Thinking Slow about Latency Evaluation for Simultaneous Machine Translation". In: *arXiv:1906.00048* (cit. on p. 138).

Cho, Eunah, Jan Niehues, and Alex Waibel (2017). "NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation". In: *Proc. of Interspeech*. ISCA, pp. 2645–2649. DOI: 10.21437/Interspeech.2017-1320 (cit. on p. 147).

Cho, Kyunghyun and Masha Esipova (2016). "Can neural machine translation do simultaneous translation?" In: *arXiv preprint arXiv:1606.02012* (cit. on p. 138).

Dai, Zihang et al. (2019). "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proc. of ACL*, pp. 2978–2988 (cit. on p. 138).

Dalvi, Fahim et al. (2018). "Incremental Decoding and Training Methods for Simultaneous Translation in Neural Machine Translation". In: *Proc. of NAACL-HLT*. ACL, pp. 493–499 (cit. on p. 140).

Di Gangi, Mattia A. et al. (2019). "MuST-C: a Multilingual Speech Translation Corpus". In: *Proc. of NAACL-HLT*. ACM, pp. 2012–2017 (cit. on p. 147).

Elbayad, Maha, Laurent Besacier, and Jakob Verbeek (2020). "Efficient Wait-k Models for Simultaneous Machine Translation". In: *Proc. of Interspeech*, pp. 1461–1465 (cit. on pp. 138, 142, 149).

Elbayad, Maha, Ha Nguyen, et al. (2020). "ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020". In: *Proc. of IWSLT*. ACL, pp. 35–43 (cit. on p. 150).

Iranzo-Sánchez, Javier, Jorge Civera Saiz, and Alfons Juan (2021). "Stream-level Latency Evaluation for Simultaneous Machine Translation". In: *Findings of ACL: EMNLP*. ACL, pp. 664–670 (cit. on pp. 138, 145).

Iranzo-Sánchez, Javier et al. (2020). "Direct Segmentation Models for Streaming Speech Translation". In: *Proc. of EMNLP*, pp. 2599–2611 (cit. on pp. 139, 147).

Junczys-Dowmunt, Marcin (2019). "Microsoft Translator at WMT 2019: Towards Large-Scale Document-Level Neural Machine Translation". In: *Proc. of WMT*, pp. 225–233 (cit. on pp. 145, 159).

Kudo, Taku and John Richardson (2018). "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *Proc. of EMNLP: System Demonstrations*. ACL, pp. 66–71 (cit. on p. 158).

Li, Daniel et al. (2020). "Sentence Boundary Augmentation For Neural Machine Translation Robustness". In: *arXiv preprint arXiv:2010.11132* (cit. on p. 138).

Lui, Marco and Timothy Baldwin (2012). "langid.py: An Off-the-shelf Language Identification Tool". In: *Proc. of ACL: System Demonstrations*. ACL, pp. 25–30 (cit. on p. 158).

Ma, Mingbo et al. (2019). "STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework". In: *Proc. of ACL*. ACL, pp. 3025–3036. DOI: 10.18653/v1/P19-1289 (cit. on pp. 138, 140).

Ma, Shuming, Dongdong Zhang, and Ming Zhou (2020). "A Simple and Effective Unified Encoder for Document-Level Machine Translation". In: *Proc. of ACL*. Ed. by Dan Jurafsky et al. ACL, pp. 3505–3511 (cit. on p. 138).

Ma, Xutai et al. (2020). "Monotonic Multihead Attention". In: *Proc. ICLR 2020*. OpenReview.net (cit. on p. 140).

Maruf, Sameen, Fahimeh Saleh, and Gholamreza Haffari (2021). "A Survey on Document-level Machine Translation: Methods and Evaluation". In: *arXiv preprint arXiv:1912.08494* (cit. on p. 138).

Matusov, Evgeny et al. (2005). "Evaluating machine translation output with automatic sentence segmentation". In: *Proc. of IWSLT*. ISCA (cit. on pp. 138, 146).

Niehues, Jan et al. (2018). "Low-Latency Neural Speech Translation". In: *Proc. of Interspeech*. ISCA, pp. 1293–1297 (cit. on p. 138).

Papineni, Kishore et al. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proc. of ACL*. ACL, pp. 311–318. DOI: 10.3115/1073083.1073135 (cit. on p. 145).

Pham, Ngoc-Quan et al. (2020). "KIT's IWSLT 2020 SLT Translation System". In: *Proc. of IWSLT*. ACL, pp. 55–61 (cit. on p. 150).

Scherrer, Yves, Jörg Tiedemann, and Sharid Loáiciga (2019). "Analysing concatenation approaches to document-level NMT in two different domains". In: *Proc. of DiscoMT@EMNLP*. ACL, pp. 51–61 (cit. on p. 138).

Schneider, Felix and Alexander Waibel (2020). "Towards Stream Translation: Adaptive Computation Time for Simultaneous Machine Translation". In: *Proc. of IWSLT*. ACL, pp. 228–236 (cit. on pp. 138, 145–147, 149, 150).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proc. of ACL*. ACL, pp. 1715–1725 (cit. on p. 145).

Stolcke, Andreas (2002). "SRILM - an extensible language modeling toolkit." In: *Proc. of Interspeech*. Ed. by John H. L. Hansen and Bryan L. Pellom. ISCA, pp. 901–904 (cit. on p. 147).

Tiedemann, Jörg and Yves Scherrer (2017). "Neural Machine Translation with Extended Context". In: *Proc. of DiscoMT@EMNLP*. ACL, pp. 82–92 (cit. on p. 138).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proc. of NIPS*. Ed. by Isabelle Guyon et al., pp. 5998–6008 (cit. on p. 147).

Zhang, Biao et al. (2021). "Beyond Sentence-Level End-to-End Speech Translation: Context Helps". In: *Proc. of ACL*. ACL, pp. 2566–2578 (cit. on p. 138).

Zheng, Baigong et al. (2020). "Simultaneous Translation Policies: From Fixed to Adaptive". In: *Proc. of ACL*. ACL, pp. 2847–2853 (cit. on p. 140).

Zheng, Zaixiang et al. (2020). "Towards Making the Most of Context in Neural Machine Translation". In: *Proc. of IJCAI*, pp. 3983–3989 (cit. on p. 138).

Chapter 3

General discussion of the results

This chapter highlights some important aspects and provides an integrated discussion of the results, in relation with the scientific goals that had been defined for the thesis:

- **How can we obtain a ST dataset that can be used for realistic, multilingual development and evaluation of streaming ST systems?**
- **How can the output of the ASR system be processed and segmented in order to maximize the performance of the cascaded MT system?**
- **How can the performance of streaming ST systems be evaluated with a procedure that takes into account the sequential nature of the problem and is at the same time interpretable?**
- **How can a streaming ST system best take advantage of contextual information in order to improve translation quality?**

In order to obtain a reliable ST dataset that could serve as the backbone of further research, the Europarl-ST corpus was collected and presented in Paper 1. This sets up the foundation for the follow-up research carried out, as Europarl-ST provides a reliable and realistic benchmark for training and evaluation of streaming ST models. Out of all the contributions presented, the Europarl-ST corpus is by far the most impactful breakthrough achieved on this thesis. The corpus has become one of the de-facto benchmarks for ST, and as of June 2023, it has amassed more than 100 citations. We have also been made aware through informal conversations that this corpus is also used internally by some of the major technology players in the sector. Indeed, Europarl-ST is also being used as the main evaluation benchmark for both ASR and ST at the MLLP research group.

Originally envisioned as a benchmark for realistic, long-form ST, the evaluation procedure for Europarl-ST was described as follows:

1. Take the audio of the whole intervention of a Member of the European Parliament (MEP), and segment it into the type of segments expected by the ST system.
2. Transcribe and translate each segment.
3. Re-align the hypothesis with the reference, and use the re-aligned version for evaluation.

This matches the standard evaluation procedure for long-form ST (as well as ASR if you stop before the last step), and ensures a realistic evaluation under the same conditions as the real, inference-time scenario. However, the vast majority of papers using Europarl-ST for evaluation do not follow this procedure, and instead use the pre-computed segments that were released with the corpus. These segments were originally released in order to reduce the human effort needed to start training a system with Europarl-ST, but when used for system evaluation they are not as reliable. Whereas the MEP speeches have been filtered in order to exclude bad quality samples, the pre-computed segments are generated using automatic audio and sentence alignment models, and are then truncated so that no audio segment exceeds 20 seconds. Indeed, this automatic procedure adds an additional step of noise that makes segment-level evaluation not as reliable as the intended intervention-level evaluation. Furthermore, isolated segment translation is a much simpler and limited task than long-form translation, and research findings that have only been tested on short, isolated segments might not hold when tested with a real, long-form task.

Table 3.1: Overview of the Europarl-ST train set, version v1.1. The rows indicate the source language (audio, transcription), and the columns the target language (translations). Each entry indicates the amount of audio hours available.

src/tgt	en	fr	de	it	es	pt	pl	ro	nl
en	-	81	83	80	81	81	79	72	80
fr	32	-	21	20	21	22	20	18	22
de	30	18	-	17	18	18	17	17	18
it	37	21	21	-	21	21	21	19	20
es	22	14	14	14	-	14	13	12	13
pt	15	10	10	10	10	-	9	9	9
pl	28	18	18	17	18	18	-	16	18
ro	24	12	12	12	12	12	12	-	12
nl	7	5	5	4	5	4	4	4	-

The original Europarl-ST publication showed experiments between 4 language pairs (English, German, French and Spanish). Italian and Portuguese were also included in the original release, but no experiments were carried out for those two languages. The development of additional alignment systems allowed us to expand the amount of languages covered. As a result, a new v1.1 version of the corpus was released, which adds Polish, Romanian and Dutch to the list of supported languages. An overview of the latest version of the corpus in shown in Table 3.1.

Table 3.2: BLEU score on the Europarl-ST set. These results show the effect of using different segmenter models for processing the transcriptions of an ASR system.

Segmenter	En-De	En-Es	En-Fr	Es-En	Fr-En	De-En
Baseline (VAD)	26.5	35.5	29.3	33.8	29.9	25.8
Text(RNN)	27.6	37.0	29.4	34.7	31.6	28.1
Audio w/o RNN	28.4	37.2	30.0	34.4	32.1	28.3
Audio w/ RNN	28.4	37.3	30.1	33.9	32.1	28.2
Text(XLM-Roberta)	29.1	38.6	31.5	36.4	33.3	29.1
Oracle	31.6	41.3	33.6	38.1	35.3	31.3

Paper 2 of this thesis focused on the second scientific question. With regards to dealing with unpunctuated ASR output, our experiments shown how the simple technique of pre-processing the MT source training data using the same recipe as the LM training data provides impressive results when compared with the do-nothing baseline. Alternatively, an specific punctuation step could be applied, but adding yet another model to the pipeline would significantly increase complexity. The majority of casing and punctuation research is focused on the offline scenario, so it would have also represented an additional challenge to adapt it to the online scenario. As a result, we selected the simpler yet effective approach of pre-processing the MT data so that it resembles ASR transcriptions.

Under the the Direct Segmentation framework, segmentation can be understood as a classification problem on which the decision to split is based on the current word of the stream, the past history and a small future window. In the original publication, an RNN followed by a series of feed-forward layers is used to obtain the probabilities of the split decision, but this is not a requirement. Any model that works over sequences can be used as a drop-in replacement for the RNN.

In recent years, pre-trained LM have become the state-of-the-art technique for many text tasks, so replacing the RNN with one of these models could further improve the results. In order to test this hypothesis, an XLM-Roberta Base model (Conneau et al. 2020)[1], fine-tuned on the Europarl-ST train set, was used to segment the text, which was then translated with the same translation model used in the original publication. Table 3.2 shows the results on the Europarl-ST test set when segmenting the ASR output with different segmenters.

[1] xlm-roberta-base

It can be observed how the XLM-Roberta text model obtains improvements of 1-2 BLEU over its RNN counterpart, and significantly reduces the gap with respect to the oracle results. This highlights the flexibility of the streaming Direct Segmentation approach. In the future, any additional breakthroughs in text classification could also be easily integrated under this framework.

Moreover, there have been parallel developments in segmentation using audio information. In particular, the SHAS technique has shown promising results (Tsiamas et al. 2022) when applied to the offline case. These findings support our hypothesis that audio information is key for achieving good segmentation quality.

On Paper 3, the previously developed techniques were integrated into a pipeline that uses a streaming ASR system, and all system hyperparameters where jointly optimized. The most relevant result of this paper is the fact that joint optimization of cascade model hyperparameters is required in order to achieve the best possible results. Figure 3.1 better illustrates this by highlighting the combinations of segmenter window d and MT wait k that belong to the Pareto frontier, that is, points for which an increase in quality implies additional latency, and vice versa. This shows how, depending on the required goal, it might be better to allocate more latency to either the segmenter or the MT model. If the hyperparameters had been selected for each system in isolation (for example by first selecting a segmenter model and then adjusting the MT policy), sub-optimal results would have been obtained.

Additionally, for the first time on this thesis, simultaneous MT system were considered. In the previous contribution, real-time translation was tackled by calling an offline MT system every time a new segment was detected by the segmenter. If the segments are small enough, this is perceived by the user as real-time translation, but significantly better results can be achieved by using an actual real-time model trained for this task.

Paper 4 introduces a latency evaluation framework for the streaming MT scenario. Up to this point, latency had been measured using either simultaneous metrics or word-level latency in seconds, but only at the segment-level. Previous attempts at computing simultaneous metrics (AP, AL, and DAL) for long streams had failed because the assumptions made in standard simultaneous MT metrics (constant oracle writing rate) do not hold when applied to long text streams. The proposed re-segmentation approach fixes this issue by computing local oracles which are consistent with the actual behaviour of the model. There are significant advantages of the proposed approach which make it the ideal candidate for streaming evaluation:

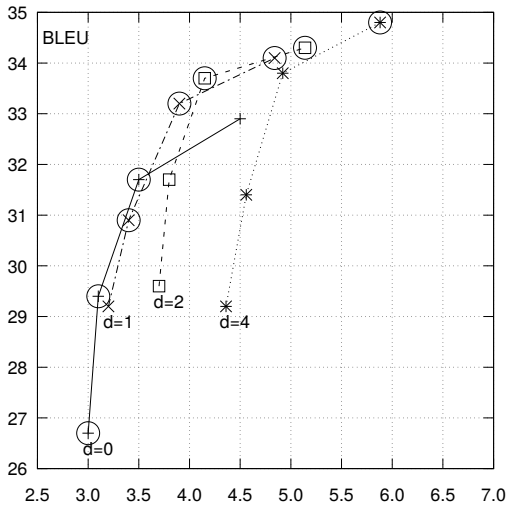

Figure 3.1: BLEU vs average word-level latency for Es-En with future window length $d = \{0, 1, 2, 4\}$ of the segmentation model on Europarl-ST dev set. Points on each curve from left to right represent increasing values of $k = \{1, 2, 4, 8\}$ in the wait-k MT system. The points belonging to the Pareto frontier are highlighted with a circle.

- The interpretation of the metrics is unchanged from their sentence-level counterparts, so it is easy to understand how the system is behaving.
- Minimum distance re-segmentation is a well understood technique that is also used for quality evaluation.
- The approach can be used to compare systems that use different segmentations.
- The metrics are computation-agnostic, so they can be used to compare systems running on different hardware setups.

A similar approach, called Translation Lag (TL) (Arivazhagan et al. 2020) uses the re-segmentation technique for computing latency using timestamps measured in seconds. A corresponding source word is computed for each output word, using the same monotonic alignment provided by the target-to-source length ratio γ of the re-computed segment. The cost assigned to each target word is the difference (in seconds) between the source word being spoken, and the target word being produced by the MT system. The final result is the average of the costs over all target words. Both approaches, that is, stream-adapted AP/AL/DAL and TL, were developed independently from each other. Using the stream-adapted metrics has the advantage of being computation-agnostic, so different approaches can be fairly compared across different hardware setups. When looking at system deployment in an specific hardware configuration, TL can be used in order to guarantee that the model is actually ready to translate in real-time. Thus, both approaches complement each other. Further research could share new insights on the effects of the differences between the two implementations, such as the final result being an average of sentence-level averages (stream-adapted AP/AL/DAL) or an average over all target word costs (TL).

All the previous developments come together in Paper 5, which uses the streaming segmenter developed in Papers 2 and 3, as well as the streaming metrics of Paper 4, in order to build and evaluate a streaming-specific MT model. Unlike a standard sentence-level MT model, the proposed streaming model uses a sliding history window that moves over the translation stream in order to provide the model with additional contextual information that would be ignored otherwise. The final result is a streaming MT system that achieves a similar quality level than the previous state-of-the-art approach but at a fraction of the latency.

When this thesis was started, cascaded ST systems were the clear winners when compared with end-to-end ST systems. During these past few years, significant advances in end-to-end ST have shrunk this gap significantly, but it

remains unclear when or if they will overtake end-to-end ST systems. On the one hand, the simplicity and ease of maintainability of having a single system is attractive when compared with a cascade composed of multiple systems with multiple points of failure. On the other hand, the fact that ASR and MT are mature tasks with widespread adoption means that for many use cases a cascaded ST system can be implemented in a fraction of the time that it takes to train a ST system from scratch, by using pre-existing ASR and MT models. Changes in how systems are trained, such as the raise of pre-trained models and massively crawled unsupervised data might also shake things up. Current ST systems also make heavy use of data augmentation, multi-task training regimes and specific components, so the lines between cascaded and end-to-end systems become very blurry at times. Given that it is impossible to predict which new techniques will be developed to improve ST performance, it seems safe to say that not only small differences in translation quality, but significant engineering and organizational considerations will be the deciding factor when choosing between end-to-end and cascaded systems for many organizations in the short and medium term.

This thesis has been developed with the financial support of the FPU scholarship program of the Government of Spain (FPU18/04135), which allowed the author to focus its time on training and research activities. Likewise, before the start of this thesis, the author worked for a year under EU's Horizon 2020 project X5gon (761758), during which the key issues to be addressed by this thesis were identified. The results obtained during this thesis were integrated into the aforementioned X5gon project, as well as Government of Spain's Multisub (RTI2018-094879-B-I00) and Erasmus+ EXPERT (no. 20-226-093604-SCH) research projects, and the CERN-UPV technology transfer contract. The author gratefully acknowledges the computer resources at Artemisa, funded by the European Union ERDF and Comunitat Valenciana as well as the technical support provided by the Instituto de Física Corpuscular, IFIC (CSIC-UV).

Some of the papers presented in this thesis have been carried out in collaboration with other researchers from the MLLP group. The conceptual design, implementation and evaluation of the main contributions of this thesis have all been implemented by the author of this thesis. The supervisors of this thesis, professors Alfons Juan and Jorge Civera provided invaluable guidance with regards to the scientific problems that should be addressed by this thesis. Researchers of the MLLP ASR team collaborated on the papers that use the output of ASR systems instead of the reference transcriptions. Specifically, recent advances in streaming ASR technology (Jorge Cano 2022) allowed this

work to use realistic, high-quality real-time ASR transcriptions as input to the streaming MT systems developed during this thesis.

Likewise, as a result of multiple research collaborations, some additional scientific contributions have also been co-authored with other MLLP research members that have not been included on this thesis. Some of the developments carried out during this thesis, most notably the streaming MT systems, where then used as input for some of the publications.

- 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

- LSTM-Based One-Pass Decoder for Low-Latency Streaming; Jorge, Javier; Giménez, Adrià; **Iranzo-Sánchez, Javier**; Silvestre-Cerdà, Joan Albert; Civera, Jorge, Sanchís; Albert, Juan; Alfons; ICASSP 2020 , Conference Core A

- 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 UPV systems for the IWSLT 2022 Simultaneous Speech Translation and Speech-to-Speech Translation tasks; **Iranzo-Sánchez, Javier**; Jorge, Javier; Pérez-González-de-Martos, Alejandro; Giménez Pastor, Adrià ; V. Garcés Díaz-Munío; Gonçal; Baquero-Arnal, Pau; Silvestre-Cerdà, Joan-Albert; Civera, Jorge; Sanchis, Albert; Juan, Alfons; IWSLT 2022, Workshop

References

Arivazhagan, N. et al. (2020). "Re-Translation Strategies for Long Form, Simultaneous, Spoken Language Translation". In: *Proc. of ICASSP*, pp. 7919–7923 (cit. on p. 171).

Conneau, Alexis et al. (2020). "Unsupervised Cross-lingual Representation Learning at Scale". In: *Proc. of ACL*, pp. 8440–8451 (cit. on p. 168).

Jorge Cano, Javier (2022). "Streaming Automatic Speech Recognition with Hybrid Architectures and Deep Neural Network Models". PhD thesis. Universitat Politècnica de València (cit. on p. 172).

Tsiamas, Ioannis et al. (2022). "SHAS: Approaching optimal Segmentation for End-to-End Speech Translation". In: *Proc. Interspeech 2022*, pp. 106–110 (cit. on p. 169).

Chapter 4

Conclusions and future work

The findings reported on this thesis are the result of almost 4 years of work devoted to addressing the challenges that need to be solved for the development of a realistic, streaming ST system that can work in a real production environment. To summarize, the main contributions of this thesis are:

- A multilingual ST dataset that enables the development and evaluation of streaming ST systems
- A novel streaming-ready segmenter that can use both audio and text features to segment ASR transcriptions into chunks to be translated by the MT model
- An evaluation framework for adapting simultaneous MT metrics to the streaming scenario
- A streaming-specific MT system that uses contextual information from the streaming process in order to improve translation quality

These very productive 4 years have laid the foundation for further development of streaming ST systems, providing answers to the data-availability and evaluation problems that had plagued the ST community for a significant number of years. When this thesis started, neural ST, and specially streaming neural ST, was still on its infancy, but now it's reaching a mature state with signifi-

cant results and applications. The author hopes that this research project has helped to move the field forward.

With regards to future work, there are many challenges that need to be addressed for streaming ST. Although the segmenter introduced in this thesis is a powerful and elegant solution for segmentation, it nevertheless adds an additional model into the cascade pipeline. Furthermore, as the results of this thesis has shown, segmentation errors can have significant impacts on translation quality. An interesting research direction would be to move towards a segmentation-free MT model, that can directly process and translate an unbounded text stream. This is no easy feat to achieve, but it would allow for more control over the translation policy and improve translation quality by eliminating cascading errors.

Current automatic evaluation metrics, be it either computation-agnostic stream metrics (AP/AL/DAL) or word-level real latency (in seconds), are computed as sentence-level or word-level averages over an entire evaluation set, but this ignores the effects of long-tail catastrophic delays that can significantly alter user perception. This is a problem that affects both simultaneous and streaming ST. Our proposed automatic streaming ST evaluation addresses some of these problems, but further analysis should be conducted to measure how systems and metrics behave when faced with these issues.

The results obtained by pre-trained models have dramatically improved during these past few years, and they have become the state-of-the-art approach across many tasks. There exist some preliminary works on evaluating their performance for MT (Zhu et al. 2023; Bawden and Yvon 2023), but they all focus on the offline task. Further research needs to be carried out in order to assess their performance for the simultaneous and streaming case, and whether any specific technique needs to be developed to adapt them to this scenario. Computational efficiency is crucial for the streaming task, something which might tip the scales towards more efficient solutions at the expense of some translation quality.

Last but not least, the vast majority of work in ST uses only acoustic or textual information during the translation process. Multimodal ST (also known as audio-visual ST) adds a third source of information: a video feed. This field has started to gather recent attention with the release of the MuAViC dataset (Anwar et al. 2023). The experiments carried out in this thesis show that adding contextual information to the streaming ST system is particularly helpful because it can help alleviate the fact that the model has only access to a partial prefix. Likewise, having access to a video feed of the speaker could

be another important source to bridge the gap with the offline task, specially for use cases such as lecture translation, where the text present in the slides would allow the model to anticipate the missing content.

References

Anwar, Mohamed et al. (2023). "MuAViC: A Multilingual Audio-Visual Corpus for Robust Speech Recognition and Robust Speech-to-Text Translation". In: *arXiv:2303.00628* (cit. on p. 176).

Bawden, Rachel and François Yvon (2023). "Investigating the Translation Performance of a Large Multilingual Language Model: the Case of BLOOM". In: *arXiv:2303.01911* (cit. on p. 176).

Zhu, Wenhao et al. (2023). "Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis". In: *arXiv:2304.04675* (cit. on p. 176).