Neural Networks 142 (2021) 303–315

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2021 Special Issue

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

Machine Learning and Language Processing Group, Valencian Research Institute for Artificial Intelligence, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain

ARTICLE INFO

Article history:
Available online 17 May 2021

Keywords:
Streaming Cascade Speech Translation
Segmentation Model

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.

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, Jaitly, Le, & Vinyals, 2016; Irie, Zeyer, Schlüter, & Ney, 2019; Jorge et al., 2020; Park, et al., 2019), but also other closely connected fields, such as Machine Translation (MT) (Bahdanau, Cho, & Bengio, 2015; Sennrich, Haddow, & Birch, 2016a, 2016b; 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 (Berard, Besacier, Kocabiyikoglu, & Pietquin, 2018; Gangi, Negri, Cattoni, Dessi, & Turchi, 2019; Jia, et al., 2019; Weiss, Chorowski, Jaitly, Wu, & Chen, 2017). 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, & 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

[*] Corresponding author.
E-mail address: jorcisai@vrain.upv.es (J. Civera).

https://doi.org/10.1016/j.neunet.2021.05.013
0893-6080/© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

decoupled models, if sufficient data would be available. All in all, cascade systems still outperform end-to-end systems in standard setups (Bahar, et al., 2020; Niehues, et al., 2019; Pino, et al., 2019).

Streaming cascade-based ST systems just started to become available (Bahar, et al., 2020) thanks to recent advances in streaming hybrid and end-to-end ASR systems stating competitive results compared to offline systems (Jorge, et al., 2019, 2020; Miao, Cheng, Gao, Zhang, & Yan, 2020; Moritz, Hori, & Le, 2020; Zeyer, Bahar, Irie, Schlüter, & Ney, 2019; Zeyer, Schlüter, & Ney, 2016; Zhang, Lu, et al., 2020), and also due to the significant progress in simultaneous MT (Arivazhagan, et al., 2019, 2020; Ma, et al., 2019; Niehues, Pham, Ha, Sperber, & Waibel, 2018; Zheng, Zheng, Ma, & Huang, 2019). However, the segmentation of the ASR output is still essential to deal with the simultaneous translation of long audio streams (Cho, Niehues, Kilgour, & Waibel, 2015; Fügen, Waibel, & Kolss, 2007; Gu, Neubig, Cho, & Li, 2017; Iranzo-Sánchez, et al., 2020; Oda, Neubig, Sakti, Toda, & Nakamura, 2014; Rangarajan Sridhar, Chen, Bangalore, Ljolje, & Chengalvarayan, 2013). 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 (Dai, et al., 2019; Popel & Bojar, 2018). 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[1] 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, 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, 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 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 optimized 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.

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

This paper is organized as follows. The neural-based stream-adapted ASR and MT components of the ST system are reviewed in Sections 2 and 3 , respectively. Next, Section 4 describes in full detail the DS model seamlessly integrated between the ASR and MT components to allow streaming ST decoding. Then, in Section 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 6.

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.

2.1. Acoustic model

Over the last decade, deep Feed-Forward Networks (FFNs) (Hinton, et al., 2012), Convolutional Neural Networks (CNNs) (Bozheniuk, Zeyer, Schlüter, & Ney, 2020; Sainath, et al., 2015) and Recurrent Neural Networks (RNNs) (Schuster & Paliwal, 1997) have contributed to improve acoustic modeling with respect to the historical approach based on Gaussian Mixture Model (GMM) (Bourlard & Wellekens, 1990; Russell & Moore, 1985). Indeed, RNNs based on the Long–Short Term Memory (LSTM) unit (Hochreiter & Schmidhuber, 1997) have been successfully applied in ASR (Graves, Jaitly, & Mohamed, 2013; Zeyer, Doetsch, Voigtlaender, Schlüter, & Ney, 2017). More precisely, the so-called Bidirectional LSTM (BLSTM) architecture has been widely applied and studied for AM in ASR (Zeyer 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 Zeyer et al. (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, 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, et al., 2020).

2.2. Language model

Similarly to AMs, LMs have significantly evolved from the count-based *n*-gram model (Chen & Goodman, 1999; Shannon, 1948) to continuous neural LMs based on LSTM RNN (Bengio, Ducharme, Vincent, & Janvin, 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, Zhang, Cai, & Liu, 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.

2.3. Decoding

In the hybrid approach, decoding is performed with the conventional beam-search Viterbi algorithm (Viterbi, 1967; 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 & Riley, 1999, 2001; Povey, et al., 2011), and those grounded on History Conditioned Search (HCS) (Ney & 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 (Chen, Liu, Ragni, Wang, & Gales, 2017; Sundermeyer, Tüske, Schlüter, & Ney, 2014; 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, Chen, Ramabhadran, and Sethy (2014), Lee, Park, Kim, and Lee (2018) and Singh, Oualil, and Klakow (2017), but few of them have reached the technology readiness level for a production environment under streaming conditions. In Jorge, 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 previous 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 & 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. 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 (Arivazhagan, et al., 2019; Elbayad, Besacier, & Verbeek, 2020; Ma, et al., 2019; Ma, Pino, Cross, Puzon, & Gu, 2020; Raffel, Luong, Liu, Weiss, & Eck, 2017). 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 (Ma et al., 2020) and the efficient multi-path wait-*k* approach (Elbayad et al., 2020), since both have achieved competitive results on reference tasks.

On the one hand, two attention mechanisms are discussed in 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 (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 hyper-parameter 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 (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 et al. (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.

4. Direct segmentation model

Our streaming cascade-based ST system integrates the ASR and MT components described above in Sections 2 and 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 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 2.2, 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, 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

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

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

$$\hat{y}_1^J = \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_1^{j-1}, w_1^{j+d}, \check{\mathbf{x}}_1^{j+d}). \tag{2}$$

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

$$\hat{y}_1^J = \arg\max_{y_1^J} \prod_{j=1}^{J} p(y_j \mid y_{j-n}^{j-1}, w_{j-n}^{j+d}, \check{\mathbf{x}}_j^{j+d}). \tag{3}$$

The DS model estimates the probabilistic term in Eq. (3) as schematically depicted from bottom to top in Fig. 1. 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 $w_{j-n}'^{j+d}$ 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 (Cho, et al., 2014), represented by function $f_1(\;)$. So, the resulting text state vectors are defined as

$$\mathbf{s}_j^{j+d} = f_1(\mathbf{h}_{j-n}^{j+d}). \tag{4}$$

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

$$p(y_j \mid y_{j-n}^{j-1}, w_{j-n}^{j+d}, \check{\mathbf{x}}_j^{j+d}) \approx f_2([\mathbf{s}_j^{j+d}; \check{\mathbf{x}}_j^{j+d}]). \tag{5}$$

The incorporation of the acoustic word-based vectors into the segmentation model has been shown to outperform a version of

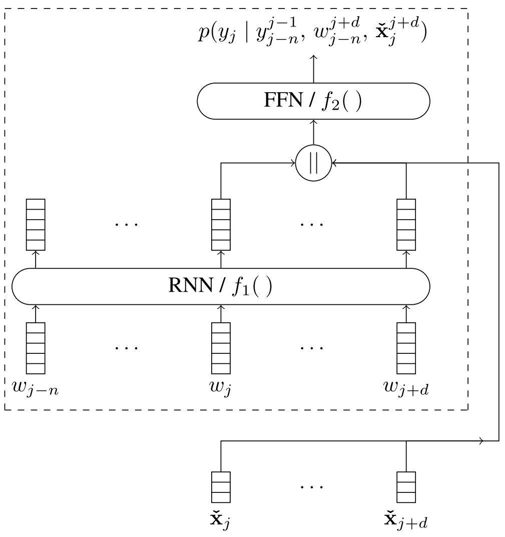
$p(y_j \mid y_{j-n}^{j-1}, w_{j-n}^{j+d}, \tilde{\mathbf{x}}_j^{j+d})$
FFN / $f_2(\,)$
$||$
RNN / $f_1(\,)$
w_{j-n}
w_j
w_{j+d}
$\tilde{\mathbf{x}}_j$
$\tilde{\mathbf{x}}_{j+d}$

Fig. 1. Architectural overview of the DS model. At the bottom, input acoustic word-based vectors $\tilde{\mathbf{x}}_j^{j+d}$ 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 $p(y_j \mid y_{j-n}^{j-1}, w_{j-n}^{j+d}, \tilde{\mathbf{x}}_j^{j+d})$.

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, et al., 2020).

At training time, the components inside the dashed boundary in Fig. 1 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. (3), but no significant differences in performance were observed between them (Iranzo-Sánchez, 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.

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.

5.1. 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, 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 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 & 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 1. For the purpose of 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 s, and 27 to 28 words for Spanish and English, and 32 to 33 words for French.

5.2. Automatic speech recognition

The AMs integrated in our ASR system follow the hybrid approach introduced in Section 2.1. 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 25 ms shifted at 10 ms intervals into 16 Mel-frequency cepstral coefficients (MFCC) plus deltas and accelerations (Zolnay, Schluter, & 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) (del 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 & Dave, 2012). This network is trained using TensorFlow (Abadi et al., 2015) with cross-entropy loss during 16 epochs. Dropout (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 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 Zeyer et al. (2017). Table 2 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 h, which included the dev set of Europarl-ST, was used to tune model hyperparameters.

Table 3 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 & 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

Table 1
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					Test				
Lang Pairs	Vids	Chks	Hrs	Kwords Src	Kwords Trg	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	206	1816	5.3	50	50
Es–Fr	439	4673	13.7	129	149	121	1115	3.2	30	35	124	1082	3.2	31	36

Table 2
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

(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 (Chen, Liu, Qian, Gales, & Woodland, 2016) for 6 epochs. BPTT was set to consider the 6 previous words. Training criterion was based on the Noise Contrastive Estimation (NCE) (Mnih & 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 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 4 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 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 et al., 2020; Zhang 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 2.1 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.

Fig. 2 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, et al., 2020), an $n_{lookahead}$ value of 0.6 s is a reasonable baseline delay, since subsequent components

Table 3
Statistics of Spanish text resources used for language modeling. S = Sentences, RW = Running words, V = Vocabulary. Units are thousands (K).

Corpus	S(K)	RW(K)	V(K)
Internal: TV, entertainment	4799	59 235	307
Internal: education	87	1526	35
Internal: politics	1361	35 170	126
Opensubtitles (*OpenSubtitles*, 2020)	212 635	1 146 861	1576
UFAL (*UFAL Medical Corpus*, 2020)	92 873	910 728	2179
Wikipedia (*Wikipedia*, 2020)	32 686	586 068	3373
UN (Callison-Burch, Koehn, Monz, et al., 2012)	11 196	343 594	381
News Crawl (*News Crawl corpus (WMT workshop) 2015*, 2015)	7532	198 545	648
eldiario.es (*Eldiario.es*, 2020)	1665	47 542	247
El Periódico (*ElPeriodico.com*, 2020)	2677	46 637	291
Common Crawl (*CommonCrawl 2014*, 2014)	1719	41 792	486
News Commentary (*News Crawl corpus (WMT workshop) 2015*, 2015)	207	5448	83
TOTAL	369 434	3 423 146	5785

Table 4
PPLs, interpolation weights and WERs for EuroParl dev and test sets.

Model	PPL-dev	PPL-test	Weights	WER-dev	WER-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

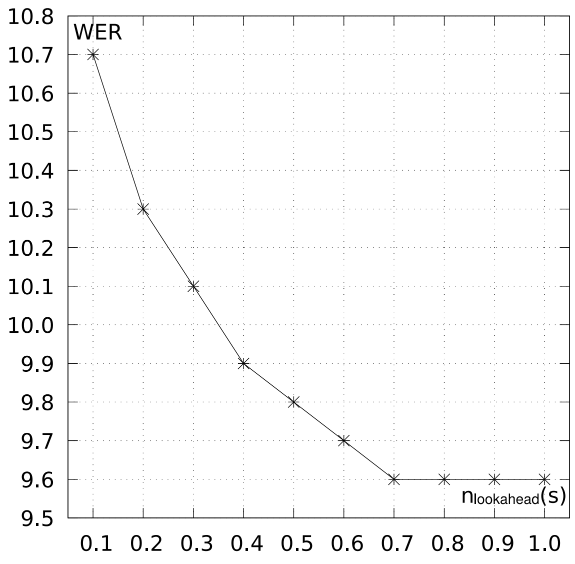

Fig. 2. WER vs $n_{\text{lookahead}}$ in seconds on the EuroParl-ST dev set.

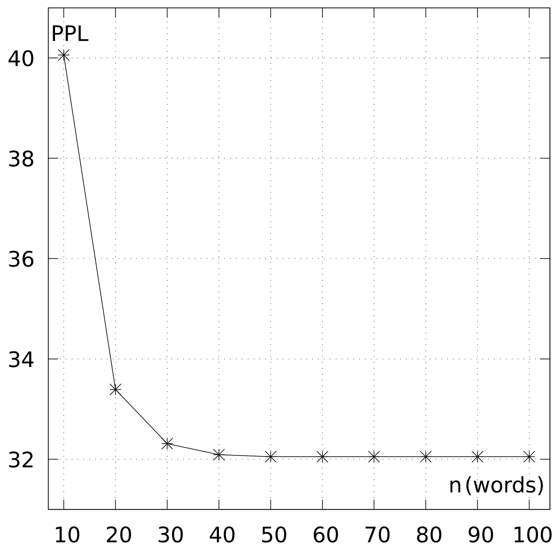

Fig. 3. Perplexity as a function of the TLM history size measured in n words on the EuroParl-ST dev set.

of the cascade ST system will introduce additional delays. Hence, we fixed this value for the following experiments.

As mentioned in Section 2.2, the computational cost of the TLM requires to limit its history size in order to perform streaming decoding. For this reason, Fig. 3 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 Fig. 3, increasing the history size consistently decreases perplexity, reaching a minimum value with a history

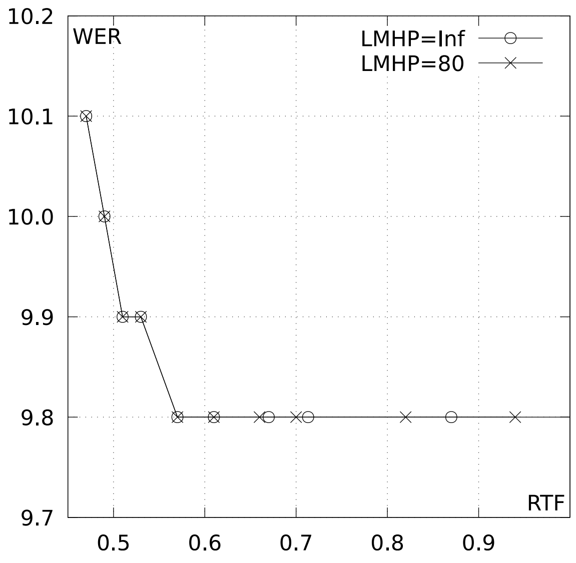

Fig. 4. WER vs. RTF as a function of the beam width without limiting the value of LMHP (LMHP=Inf) and considering LMHP equal to 80 on the Europarl-ST dev set.

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 parameter 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. Fig. 4 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, 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 s, 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 test sets, respectively. These figures are good enough to provide high-quality transcriptions to ease the downstream MT process.

5.3. 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

Table 5
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*, 2014)	1.8	–
DGT (Tiedemann, 2012)	–	4.8
EU Bookshop (Tiedemann, 2012)	5.2	4.9
EU Bulletin *EU Bulletin* (2020)	1.0	–
JRC-Acquis (Tiedemann, 2012)	–	1.6
United Nations (Callison-Burch et al., 2012)	11.2	25.8
Wikipedia (*Wikipedia*, 2020)	1.8	–

toolkit (Ott, et al., 2019). The initial models are general out-of-domain systems trained with the data shown in Table 5. After training finishes, domain adaptation by finetuning (Luong & 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 decoding time. The accuracy of MT systems is evaluated in terms of Bilingual Evaluation Understudy (BLEU) (Papineni, Roukos, Ward, & Zhu, 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) (Cho & Esipova, 2016), Average Lagging (AL) (Ma, et al., 2019) and Differentiable Average Lagging (DAL) (Cherry & 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,

$$\text{AP} = \frac{1}{|\mathbf{w}| \cdot |\mathbf{e}|} \sum_{i=1}^{|\mathbf{e}|} g(i). \tag{6}$$

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) (Ma, et al., 2019), that is,

$$\text{AL} = \frac{1}{\tau} \sum_{i=1}^{\tau} g(i) - \frac{i-1}{\gamma} \tag{7}$$

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

$$\tau = \operatorname*{arg\,min}_{i:g(i)=|\mathbf{w}|} g(i) \tag{8}$$

Both AP and AL, specially the former, present some issues (Cherry & 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

$$g'(i) = \begin{cases} g(i) & i = 1 \\ \max\left(g(i), g'(i-1) + \frac{1}{\gamma}\right) & i > 1 \end{cases} \tag{9}$$

Table 6
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

So, the DAL is defined as

$$\text{DAL} = \frac{1}{|\mathbf{e}|} \sum_{i=1}^{|\mathbf{e}|} g'(i) - \frac{i-1}{\gamma} \tag{10}$$

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 6 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 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 6, 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.

5.4. Speech translation

Once the ASR system has been adjusted for streaming conditions in Section 5.2 and the simultaneous MT system was selected in Section 5.3, we move on to evaluate the complete ST pipeline, including the segmentation model which allows for a seamless connection between the ASR and MT systems under a streaming setup.

The architectural overview of the segmentation model provided in Section 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, 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 5.3, 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 behavior 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 realized 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 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 7
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

Table 7 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.

As observed in Table 7, 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 ($\simeq$ 0.01s). 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. Fig. 5 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

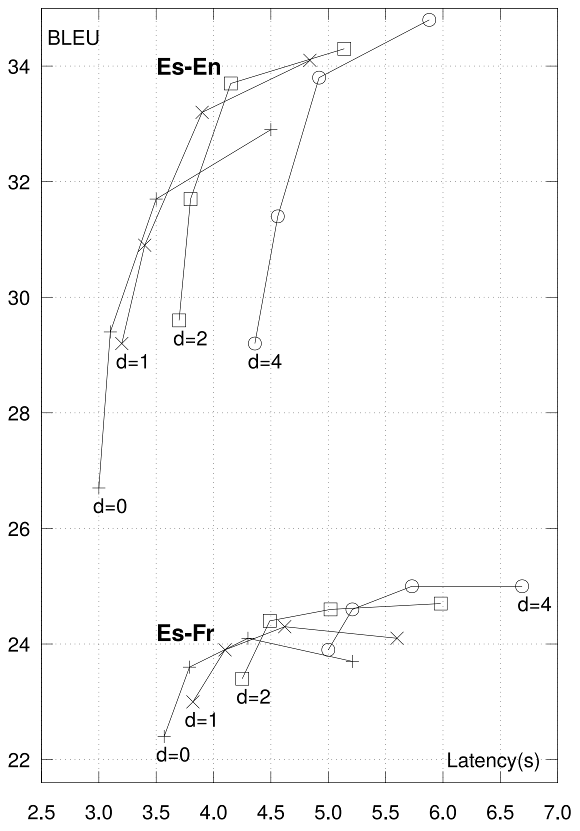

Fig. 5. BLEU vs average word-level latency for Es–En (top) and Es–Fr (bottom) with future window length $d = \{0, 1, 2, 4\}$ of the segmentation model on Europarl-ST dev set. Points on each curve from left to right represent increasing values of $k = \{1, 2, 4, 8\}$ in the wait-k MT system.

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.

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 s is a reasonable expected value (Barik, 1973; Lederer, 1978; Lee, 2002). Therefore, we select a combination of d and k values that maximizes quality but does not exceed a latency of 4 s. 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 8 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 8
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

Table 9
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

Table 9 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 hyperparameter tuning on the dev set, the average latency of the complete ST systems is 4 s 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, Giménez, Andrés-Ferrer, Civera, & Juan, 2012), a monolingual MT segmenter (Cho, Niehues, & 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 10 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

Table 10
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

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

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 simultaneous 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.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

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, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
Aggarwal, R., & Dave, M. (2012). Filterbank optimization for robust ASR using GA and PSO. *International Journal of Speech Technology*, *15*, 191–201.
del Agua, M. A., Giménez, A., Serrano, N., Andrés-Ferrer, J., Civera, J., Sanchis, A., et al. (2014). The translectures-UPV toolkit. In J. L. Navarro Mesa, A. Ortega, A. Teixeira, E. Hernández Pérez, P. Quintana Morales, A. Ravelo García, I. Guerra Moreno, D. T. Toledano (Eds.), *Advances in Speech and Language Technologies for Iberian Languages* (pp. 269–278). Springer International Publishing.
Arısoy, E., Chen, S. F., Ramabhadran, B., & Sethy, A. (2014). Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(1), 184–192.
Arivazhagan, N., Cherry, C., Macherey, W., Chiu, C.-C., Yavuz, S., Pang, R., et al. (2019). Monotonic infinite lookback attention for simultaneous machine translation. In *Proc. of ACL* (pp. 1313–1323).
Arivazhagan, N., Cherry, C., Te, I., Macherey, W., Baljekar, P., & Foster, G. (2020). Re-translation strategies for long form, simultaneous, spoken language translation. In *Proc. of ICASSP* (pp. 7919–7923).
Bahar, P., Bieschke, T., & Ney, H. (2019). A comparative study on end-to-end speech to text translation. In *Proc. of IEEE ASRU* (pp. 792–799). IEEE Signal Processing Society.
Bahar, P., Wilken, P., Alkhouli, T., Guta, A., Golik, P., Matusov, E., et al. (2020). Start-before-end and end-to-end: Neural speech translation by AppTek and RWTH Aachen university. In *Proc. of IWSLT*.
Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio, & Y. LeCun (Eds.), *Proc. of ICLR*.
Baquero-Arnal, P., Jorge, J., Giménez, A., Silvestre-Cerdà, J. A., Iranzo-Sánchez, J., Sanchis, A., et al. (2020). Improved hybrid streaming ASR with transformer language models. In *Proc. of interspeech* (pp. 2127–2131).
Barik, H. (1973). Simultaneous interpretation: Temporal and quantitative data. *Language and Speech*, *16*, 237–270.
Barrault, L., Biesialska, M., Bojar, O., Costa-jussà, M. R., Federmann, C., Graham, Y., et al. (2020). Findings of the 2020 conference on machine translation (WMT20). In *Proc. of WMT* (pp. 1–54). ACL.
Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.
Berard, A., Besacier, L., Kocabiyikoglu, A. C., & Pietquin, O. (2018). End-to-end automatic speech translation of audiobooks. In *Proc. of ICASSP* (pp. 6224–6228). IEEE.
Bourlard, H., & Wellekens, C. J. (1990). Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(12), 1167–1178.
Bozheniuk, V., Zeyer, A., Schlüter, R., & Ney, H. (2020). A comprehensive study of residual CNNs for acoustic modeling in ASR. In *Proc. of ICASSP*. IEEE.
Callison-Burch, C., Koehn, P., Monz, C., et al. (2012). Findings of the 2012 workshop on statistical machine translation. In *Proc. of WMT* (pp. 10–51). ACL.
Chan, W., Jaitly, N., Le, Q. V., & Vinyals, O. (2016). Listen, Attend and Spell: A neural network for large vocabulary conversational speech recognition. In *Proc. of ICASSP* (pp. 4960–4964).
Chen, S. F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, *13*(4), 359–394.
Chen, X., Liu, X., Qian, Y., Gales, M. J. F., & Woodland, P. C. (2016). CUED-RNNLM — An open-source toolkit for efficient training and evaluation of recurrent neural network language models. In *Proc. of ICASSP* (pp. 6000–6004). IEEE.
Chen, X., Liu, X., Ragni, A., Wang, Y., & Gales, M. J. (2017). Future word contexts in neural network language models. In *Proc of ASRU* (pp. 97–103). IEEE Signal Processing Society.
Cherry, C., & Foster, G. (2019). Thinking slow about latency evaluation for simultaneous machine translation. arXiv preprint arXiv:1906.00048.
Cho, K., & Esipova, M. (2016). Can neural machine translation do simultaneous translation? arXiv preprint arXiv:1606.02012.
Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP* (pp. 1724–1734). ACL.
Cho, E., Niehues, J., Kilgour, K., & Waibel, A. (2015). Punctuation insertion for real-time spoken language translation. In *Proc. of IWSLT*. ISCA.

Cho, E., Niehues, J., & Waibel, A. (2017). NMT-based segmentation and punctuation insertion for real-time spoken language translation. In *Proc. of interspeech* (pp. 2645–2649). ISCA.
CommonCrawl 2014. (2014). http://commoncrawl.org/.
Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of ACL* (pp. 2978–2988).
Elbayad, M., Besacier, L., & Verbeek, J. (2020). Efficient wait-k models for simultaneous machine translation. In *Proc. of interspeech* (pp. 1461–1465).
Eldiario.es. (2020). https://www.eldiario.es/.
ElPeriodico.com. (2020). https://www.elperiodico.com/.
EU Bulletin. (2020). https://ec.europa.eu/archives/bulletin/en/welcome.htm.
European Parliament, & DG Translation (2019). Live speech to text and machine translation tool for 24 Languages. URL https://etendering.ted.europa.eu/cft/cft-display.html?cftId=5249.
Fügen, C., Waibel, A., & Kolss, M. (2007). Simultaneous translation of lectures and speeches. *Machine Translation*, *21*(4), 209–252.
Gangi, M. A. D., Negri, M., Cattoni, R., Dessi, R., & Turchi, M. (2019). Enhancing transformer for end-to-end speech-to-text translation. In *Proc. of MT summit XVII* (pp. 21–31). EAMT.
Graves, A., Jaitly, N., & Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional LSTM. In *Proc. of ASRU* (pp. 273–278). IEEE, IEEE Signal Processing Society.
Gu, J., Neubig, G., Cho, K., & Li, V. O. (2017). Learning to translate in real-time with neural machine translation. In *Proc. of EACL* (pp. 1053–1062). ACL.
Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.
Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.
Iranzo-Sánchez, J., Giménez, A., Silvestre-Cerdà, J. A., Baquero, P., Civera, J., & Juan, A. (2020). Direct segmentation models for streaming speech translation. In *Proc. of EMNLP* (pp. 2599–2611). ACL.
Iranzo-Sánchez, J., Silvestre-Cerdà, J. A., Jorge, J., Roselló, N., Giménez, A., Sanchis, A., et al. (2020). Europarl-ST: A multilingual corpus for speech translation of parliamentary debates. In *Proc. of ICASSP* (pp. 8229–8233). IEEE.
Irie, K., Zeyer, A., Schlüter, R., & Ney, H. (2019). Language modeling with deep transformers. In *Proc. of interspeech* (pp. 3905–3909).
Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., et al. (2019). Direct speech-to-speech translation with a sequence-to-sequence model. In *Proc. of interspeech* (pp. 1123–1127). ISCA.
Jorge, J., Giménez, A., Iranzo-Sánchez, J., Civera, J., Sanchis, A., & Juan, A. (2019). Real-time one-pass decoder for speech recognition using LSTM language models. In *Proc. of interspeech* (pp. 3820–3824).
Jorge, J., Giménez, A., Iranzo-Sánchez, J., Silvestre-Cerda, J. A., Civera, J., Sanchis, A., et al. (2020). LSTM-based one-pass decoder for low-latency streaming. In *Proc. of ICASSP* (pp. 7814–7818). IEEE.
Junczys-Dowmunt, M. (2019). Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proc. of WMT* (pp. 225–233).
Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling. *1*, In *Proc .of ICASSP* (pp. 181–184). IEEE.
Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit* (pp. 79–86). AAMT.
Lederer, M. (1978). Simultaneous interpretation — Units of meaning and other features. In D. Gerver, & H. W. Sinaiko (Eds.), *Language interpretation and communication* (pp. 323–332). Springer US.
Lee, T.-H. (2002). Ear voice span in english into Korean simultaneous interpretation. *Meta*, *47*(4), 596–606.
Lee, K., Park, C., Kim, N., & Lee, J. (2018). Accelerating recurrent neural network language model based online speech recognition system. arXiv preprint arXiv:1801.09866.
Lleida, E., Ortega, A., Miguel, A., Bazán-Gil, V., Pérez, C., Gómez, M., et al. (2019). Albayzin 2018 evaluation: The IberSpeech-RTVE challenge on speech technologies for spanish broadcast media. *Applied Sciences*.
Luong, M.-T., & Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domain. In *Proc. of IWSLT*.
Ma, M., Huang, L., Xiong, H., Zheng, R., Liu, K., Zheng, B., et al. (2019). STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proc. of ACL* (pp. 3025–3036).
Ma, X., Pino, J. M., Cross, J., Puzon, L., & Gu, J. (2020). Monotonic multihead attention. In *Proc. ICLR 2020*. OpenReview.net.
Miao, H., Cheng, G., Gao, C., Zhang, P., & Yan, Y. (2020). Transformer-based online CTC/attention end-to-end speech recognition architecture. In *Proc. of ICASSP* (pp. 6084–6088).
Mnih, A., & Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. arXiv preprint arXiv:1206.6426.
Mohri, M., & Riley, M. (1999). Integrated context-dependent networks in very large vocabulary speech recognition. In *Proc. of ECSCT*.
Mohri, M., & Riley, M. (2001). A weight pushing algorithm for large vocabulary speech recognition. In *Proc. of ECSCT*.
Moritz, N., Hori, T., & Le, J. (2020). Streaming automatic speech recognition with the transformer model. In *Proc. of ICASSP* (pp. 6074–6078).
News Crawl corpus (WMT workshop) 2015. (2015). http://www.statmt.org/wmt15/translation-task.html.
Ney, H. (1984). The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *32*(2), 263–271.
Ney, H., & Ortmanns, S. (2000). Progress in dynamic programming search for LVCSR. *Proc. IEEE*, *88*(8), 1224–1240.
Niehues, J., Cattoni, R., Stüker, S., Negri, M., Turchi, M., Ha, T., et al. (2019). The IWSLT 2019 evaluation campaign. In *Proc. of IWSLT*. ISCA.
Niehues, J., Pham, N.-Q., Ha, T.-L., Sperber, M., & Waibel, A. H. (2018). Low-latency neural speech translation. In *Proc. of interspeech* (pp. 1293–1297). ISCA.
Nolden, D. (2017). *Progress in decoding for large vocabulary continuous speech recognition* (Ph.D. thesis), RWTH Aachen University, Germany.
Oda, Y., Neubig, G., Sakti, S., Toda, T., & Nakamura, S. (2014). Optimizing segmentation strategies for simultaneous speech translation. In *Proc. of ACL* (pp. 551–556).
OpenSubtitles. (2020). http://www.opensubtitles.org/.
Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., et al. (2019). Fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL-HLT: Demonstrations*. ACL.
Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL* (pp. 311–318).
Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., et al. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proc. of Interspeech* (pp. 2613–2617).
Pino, J., Puzon, L., Gu, J., Ma, X., McCarthy, A. D., & Gopinath, D. (2019). Harnessing indirect training data for end-to-end automatic speech translation: Tricks of the trade. In *Proc. of IWSLT*. ISCA.
Popel, M., & Bojar, O. (2018). Training tips for the Transformer model. *The Prague Bulletin of Mathematical Linguistics*, *110*(1), 43–70.
Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., et al. (2011). The kaldi speech recognition toolkit. In *Proc. of ASRU*. IEEE Signal Processing Society.
Raffel, C., Luong, M.-T., Liu, P. J., Weiss, R. J., & Eck, D. (2017). Online and linear-time attention by enforcing monotonic alignments. *70*, In *Proc. of ICML* (pp. 2837–2846). PMLR.
Rangarajan Sridhar, V. K., Chen, J., Bangalore, S., Ljolje, A., & Chengalvarayan, R. (2013). Segmentation strategies for streaming speech translation. In *Proc. of NAACL-HLT* (pp. 230–238). ACL.
Russell, M., & Moore, R. (1985). Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition. In *Proc. of ICASSP (vol. 10)* (pp. 5–8).
Sainath, T. N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A.-r., Dahl, G., et al. (2015). Deep convolutional neural networks for large-scale speech tasks. *Neural Networks*, *64*, 39–48.
Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.
Schwenk, H. (2007). Continuous space language models. *Computer Speech and Language*, *21*(3), 492–518.
Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proc. of ACL* (pp. 1715–1725).
Sennrich, R., Haddow, B., & Birch, A. (2016). Improving neura machine translation models with monolingual data. In *Proc. of ACL* (pp. 86–96).
Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423.
Shi, Y., Zhang, W.-Q., Cai, M., & Liu, J. (2014). Efficient one-pass decoding with NNLM for speech recognition. *IEEE Signal Processing Letters*, *21*(4), 377–381.
Silvestre-Cerdà, J. A., Giménez, A., Andrés-Ferrer, J., Civera, J., & Juan, A. (2012). Albayzin evaluation: The PRHLT-UPV audio segmentation system. In *Proc. of IberSPEECH 2012* (pp. 596–600).
Singh, M., Oualil, Y., & Klakow, D. (2017). Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition. In *Proc. of interspeech* (pp. 2720–2724).
Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In *Proc. of interspeech*. ISCA.
Sundermeyer, M., Tüske, Z., Schlüter, R., & Ney, H. (2014). Lattice decoding and rescoring with long-span neural network language models. In *Proc. of ISCA*.
Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In *Proc. of LREC* (pp. 2214–2218). ELRA.
UFAL Medical Corpus. (2020). http://ufal.mff.cuni.cz/ufal_medical_corpus.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Proc. of NIPS* (pp. 5998–6008).

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269.

Weiss, R. J., Chorowski, J., Jaitly, N., Wu, Y., & Chen, Z. (2017). Sequence-to-sequence models can directly translate foreign speech. In *Proc. of interspeech* (pp. 2625–2629). ISCA.

Wikipedia. (2020). https://www.wikipedia.org/.

Xu, H., Chen, T., Gao, D., Wang, Y., Li, K., Goel, N., et al. (2018). A pruned RNNLM lattice-rescoring algorithm for automatic speech recognition. In *Proc. of ICASSP* (pp. 5929–5933).

Yagi, S. (2000). Studying style in simultaneous interpretation. *Meta: Journal des traducteurs/Meta: Translators' Journal*, *45*(3), 520–547.

Zeyer, A., Bahar, P., Irie, K., Schlüter, R., & Ney, H. (2019). A comparison of transformer and LSTM encoder decoder models for ASR. In *Proc. of ASRU* (pp. 8–15). IEEE Signal Processing Society.

Zeyer, A., Doetsch, P., Voigtlaender, P., Schlüter, R., & Ney, H. (2017). A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition. In *Proc. of ICASSP* (pp. 2462–2466). IEEE.

Zeyer, A., Schlüter, R., & Ney, H. (2016). Towards online-recognition with deep bidirectional LSTM acoustic models. In *Proc. of interspeech* (pp. 3424–3428). ISCA.

Zhang, Q., Lu, H., et al. (2020). Transformer transducer: a streamable speech recognition model with transformer encoders and RNN-T loss. In *Proc. of ICASSP* (pp. 7829–7833).

Zheng, B., Zheng, R., Ma, M., & Huang, L. (2019). Simpler and faster learning of adaptive policies for simultaneous translation. In *Proc. of EMNLP-IJCNLP* (pp. 1349–1354). ACL.

Zhou, W., Michel, W., Irie, K., Kitza, M., Schlüter, R., & Ney, H. (2020). The RWTH ASR system for ted-lium release 2: Improving hybrid HMM with SpecAugment. In *Proc. of ICASSP* (pp. 7839–7843).

Zolnay, A., Schluter, R., & Ney, H. (2005). Acoustic feature combination for robust speech recognition. In *Proc. of ICASSP 2005 (vol. 1)* (pp. I–457). IEEE.