Don't Discard Fixed-Window Audio Segmentation in Speech-to-Text Translation

Chantal Amrhein[1] and **Barry Haddow**[2]
[1]Department of Computational Linguistics, University of Zurich
[2]School of Informatics, University of Edinburgh
amrhein@cl.uzh.ch, bhaddow@ed.ac.uk

Abstract

For real-life applications, it is crucial that end-to-end spoken language translation models perform well on continuous audio, without relying on human-supplied segmentation. For *online* spoken language translation, where models need to start translating before the full utterance is spoken, most previous work has ignored the segmentation problem. In this paper, we compare various methods for improving models' robustness towards segmentation errors and different segmentation strategies in both offline and online settings and report results on translation quality, flicker and delay. Our findings on five different language pairs show that a simple fixed-window audio segmentation can perform surprisingly well given the right conditions.[1]

1 Introduction

End-to-end spoken language translation (SLT) has seen considerable advances in recent years. To apply these findings to real online and offline SLT settings, we need to be able to process continuous audio input. However, most previous work on end-to-end SLT makes use of human-annotated, sentence-like gold segments both at training and test time which are not available in real-life settings. Unfortunately, SLT models that were trained on such gold segments often suffer a noticeable quality loss when applied to artificially split audio segments (Zhang et al., 2021; Tsiamas et al., 2022b). This also highlights that a good segmentation is more important for SLT than for automatic speech recognition (ASR) because we need to split the audio into "translatable units". For a cascade system, a segmenter/punctuator can be inserted between the ASR and machine translation (MT) model (Cho et al., 2017) in order to create suitable segments for the MT model. However for end-to-end SLT systems, it is still not clear how to best translate continuous input.

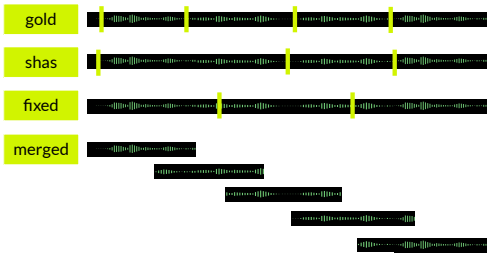

Figure 1: Visualisation of the different audio segmentation methods studied in this paper.

Solving this problem is very much an active research field that has mainly been tackled from two sides: (1) improving SLT models to be more robust towards segmentation errors (Gaido et al., 2020; Li et al., 2021; Zhang et al., 2021) and (2) developing strategies to split streaming audio into segments that resemble the training data more closely (Gaido et al., 2021; Tsiamas et al., 2022b). Both types of approaches were successfully used in recent years for the IWSLT offline SLT shared task (Ansari et al., 2020; Anastasopoulos et al., 2021, 2022) to translate audio without gold segmentations. However, they have not yet been tested systematically in the online SLT setup where translation starts before the full utterance is spoken. Recent editions of the IWSLT simultaneous speech translation shared task focused more on evaluation using the gold segmentation rather than unsegmented audio (Anastasopoulos et al., 2021, 2022). Segmenting streaming audio is especially interesting in online SLT because aside from effects on translation quality, different segmentations can also influence the delay (or latency) of the generated translation.

In this paper, we aim to fill this gap and focus on the end-to-end online SLT setup. We suspect that there is an interplay between more robust models and better segmentation strategies

[1]We publicly release our code and model outputs here: https://github.com/ZurichNLP/window_audio_segmentation

Proceedings of the Seventh Conference on Machine Translation (WMT), pages 203 - 219

and that an isolated comparison may not be informative enough. Consequently, we explore different combinations of these two approaches for two different SLT models and present results in five language pairs. Figure 1 shows the four segmentation methods we study in this work (see also Section 3.3). Our experiments follow the popular retranslation approach (Niehues et al., 2016, 2018; Arivazhagan et al., 2020a,b) where a partial segment is retranslated every time new audio becomes available. Retranslation has the advantage of being a simple approach to online SLT, which can use a standard MT inference engine. As a side-effect, the previous translation can change in later retranslations and the resulting "flicker" (i.e. sudden translation changes in the output of previous time steps) is also considered in our evaluation of different strategies.

Our main contributions are:

- We explore various combinations of segmentation strategies and robustness-finetuning approaches for translating unsegmented audio in an online SLT setup.
- We find that the advantage of dedicated audio segmentation models over a fixed-window approach becomes much smaller if the translation model is context-aware, and merging translations of overlapping windows can perform comparatively to the gold segmentation.
- We discuss issues with the evaluation of delay in an existing evaluation toolkit for retranslation when different segmentations are used and show how these can be mitigated.

2 Related Work

In recent years, the IWSLT shared task organisers have stopped providing gold segmented test sets for the offline speech translation task which has lead to increased research focus on audio segmentation (Ansari et al., 2020; Anastasopoulos et al., 2021, 2022). One obvious strategy to segment audio is to create fixed windows of the same duration, but previous research has mostly relied on more elaborate methods. Typically, methods with voice activity detection (VAD) (Sohn et al., 1999) were employed to identify natural breaks in the speech signal. However, VAD models do not guarantee breaks that align with complete utterances and can produce segments that are too long or too short which is why hybrid approaches that also consider the length of the predicted utterance can be helpful (Potapczyk and Przybysz, 2020; Gaido et al., 2021; Shanbhogue et al., 2022). Most recently, Tsiamas et al. (2022b) finetune a wav2vec 2.0 model (Baevski et al., 2020) to predict gold segmentation-like utterance boundaries, an approach which outperforms several alternative segmentation methods and was widely adopted in the 2022 IWSLT offline SLT shared task (Tsiamas et al., 2022a; Pham et al., 2022; Gaido et al., 2022).

Apart from improving automatic audio segmentation methods, previous research has also focused on making SLT models more robust toward segmentation errors. Gaido et al. (2020) and Zhang et al. (2021) both explore context-aware end-to-end SLT models and show that context can help to better translate VAD-segmented utterances. Similarly, training on artificially truncated data can be beneficial to segmentation robustness in cascaded setups (Li et al., 2021) but also in end-to-end models (Gaido et al., 2020). While this approach can introduce misalignments between source audio and target text, such misalignments in the training data are not necessarily harmful to SLT models as Ouyang et al. (2022) recently showed in an evaluation of the MuST-C dataset (Di Gangi et al., 2019).

Both of these approaches – improving automatic segmentation and making models more robust toward segmentation errors – can be combined. For example, Papi et al. (2021) show that continued finetuning on artificial segmentation can help narrow the gap between hybrid segmentation approaches and manual segmentation. However, a combination of both methods is not always equally beneficial. Gaido et al. (2022) repeat Papi et al. (2021)'s analysis with the segmentation model proposed by Tsiamas et al. (2022b) and show that for this segmentation strategy, continued finetuning on resegmented data does not lead to an improvement in translation quality.

In our work, we aim to extend these efforts and test various combinations of segmentation and model finetuning strategies. We are especially interested in fixed-window segmentations which have largely been ignored in SLT research but are attractive from a practical point of view because they do not require an additional model to perform segmentation. To the best of our knowledge, we are the first to perform such an extensive segmentation-

	train		test	
	# talks	# segments	# talks	# segments
en-de	2,043	229,703	27	2,641
es-en	378	36,263	15	996
fr-en	250	30,171	11	1,041
it-en	221	24,576	11	979
pt-en	279	30,855	11	1,022
multi	1,128	121,865	48	4038

Table 1: Overview of dataset statistics. The last row shows the total numbers for the multilingual model on es-en, fr-en, it-en and pt-en combined.

focused analysis for online SLT, considering delay, flicker and translation quality for the evaluation.

3 Experiment Setup

3.1 Data

We run experiments with TED talk data in five different language pairs where the task is to translate a TED talk as an incoming stream without having any gold sentence segmentation.

For English-to-German, we use the data from the MuST-C corpus (Di Gangi et al., 2019) version 1.0[2]. This dataset is built from TED talk audio with human-annotated transcriptions and translations. For testing, we use the "tst-COMMON" test set. For Spanish-, French-, Italian- and Portuguese-to-English, we use the data from the mTEDx corpus (Salesky et al., 2021)[3]. This dataset is also based on TED talks and provides human annotated transcriptions and translations of the audio files. For testing, we use the "iwslt2021" test set from the IWSLT 2021 multilingual speech translation shared task (Anastasopoulos et al., 2021). The dataset statistics can be seen in Table 1.

3.2 Spoken Language Translation Models

We base all our experiments on the joint speech- and text-to-text model (Tang et al., 2021a,b,c) released by Meta AI. For the English-German experiments, we use the model provided by Tang et al. (2021b)[4] and for the other language pairs, we use the multilingual model provided by Tang et al. (2021a)[5]. We refer to these models as the **original** models. These models are trained on full segments that mostly comprise one sentence:

> **And like with all powerful technology, this brings huge benefits, but also some risks.**

To investigate the effects of different segmentation strategies combined with segmentation-robust models, we finetune three different variants based on each model. In each case, the finetuning data is augmented with artificially segmented data, but no segments cross the boundaries between the individual TED talks.

- **prefix:** This model is finetuned on a 50-50 mix of original segments and synthetically created prefixes (i.e. sentences where the end is arbitrarily chopped off). Finetuning on prefixes should help for translating artificially segmented audio where the segment stops in the middle of an utterance. We create prefixes of the original segments by randomly sampling a new duration for an audio segment and using the length ratio to extract the corresponding target text. An example for a prefixed version of the original segment can be seen here:

  > **And like with all**

- **context:** This model is finetuned on a mix of original segments and synthetically created longer segments. Context was already shown to help with segmentation errors by Zhang et al. (2021). This model should be able to translate segments that consist of multiple utterances. For each segment in the original training set, we randomly either use the original segment (50% of the time) or an extended segment created by prepending the previous segment (25% of the time) or the 2 previous segments (also 25% of the time). We then add context-prefixed segments for each of these (possibly-extended) segments, by truncating the last concatenated segment. An example for a context-prefixed version of the original segment can be seen here:

  > We work every day to generate those kinds of technologies, safe and useful. **And like with all powerful technology, this brings huge benefits,**

- **windows:** This model is finetuned on a 50-50 mix of original segments and windows of random duration. We split the audio into windows by starting at the beginning of the audio

[2] https://ict.fbk.eu/must-c/

[3] http://www.openslr.org/100

[4] https://github.com/facebookresearch/fairseq/blob/main/examples/speech_text_joint_to_text/docs/ende-mustc.md

[5] https://github.com/facebookresearch/fairseq/blob/main/examples/speech_text_joint_to_text/docs/iwslt2021.md

and then sampling the duration of the first window. The end of this window then becomes the start of the next window and we repeat this process until we reach the end of a TED talk. For every such window, we extract the corresponding target text from the time-aligned gold segment(s) via length ratios. This mirrors the conditions at inference time with a fixed-window segmentation where a segment can start and end anywhere in an utterance and can also comprise multiple utterances. The segment durations are sampled uniformly between 10 and 30 seconds. Note that this model will see the qualitatively poorest data out of all finetuned models because both the end of the segment and the beginning depend on length ratios which can introduce alignment errors. An example for a window version of the original segment can be seen here:

> or death diagnosis without the help of artificial intelligence. We work every day to generate those kinds of technologies, safe and useful. **And like with all powerful technology, this brings huge benefits, but also some risks.** I don't know how this debate ends, but what I'm sure of, is that the game

All models are trained from the original checkpoint for an additional 20k steps and the last two checkpoints are averaged if more than one is saved. We do this finetuning by continuing training with the config file of the original model. For the English→German MuST-C model, we train on the audio as well as the corresponding phoneme sequences based on the transcript, however, we do not use additional parallel text data during finetuning. For the multilingual mTEDx model, we only train on data for the selected language pairs and only on audio (no phoneme sequences) because this model was already finetuned on the spoken language translation task. The validation sets only contain gold segments and all models stop training due to the step limit before early stopping is triggered.

3.3 Segmentation Strategies

We consider four different inference-time segmentation strategies in our experiments, visualised in Figure 1:

- **gold:** These are human annotated segmentation boundaries that are released as part of the MuST-C and mTEDx data. This segmentation can be viewed as an oracle segmentation even though it may not necessarily be the best segmentation for all models. Using the gold segmentation in practice is unrealistic, especially in the online setting where there would be no time for a human to segment the audio before translation.

- **SHAS:** This segmentation method was recently proposed by Tsiamas et al. (2022b). The authors finetune a pretrained wav2vec 2.0 model (Baevski et al., 2020) on the gold segmentations and train it to predict probabilities for segmentation boundaries. SHAS can be used both in offline and online setups using different algorithms to determine the segmentation boundaries based on the model's probabilities. Since we perform our experiments in an online setup, we use the pSTREAM algorithm to identify segments with SHAS. We set the maximum segment length to 18 seconds which the authors reported as best-performing.

- **fixed:** This is a simple approach that splits the audio stream into independent fixed windows of a given duration. In our experiments, we use durations of 26 seconds, which performed best in experiments by Tsiamas et al. (2022b).

- **merged:** Similarly to above, we consider fixed-size windows for this segmentation strategy but here we construct overlapping windows. We use a duration of 15 seconds[6] and shift the window with a stride of 2 seconds at a time. The translations of these overlapping windows are merged before the next window is translated (see Section 3.5).

3.4 Retranslation

We employ a retranslation strategy (Niehues et al., 2016, 2018; Arivazhagan et al., 2020a,b) for our end-to-end SLT experiments. This means that we retranslate the incoming audio at fixed time intervals. In our experiments, we retranslate every 2 seconds to be consistent with the 2-second stride from the merging windows approach. Because of such retranslations of the full audio segment — from the start of the segment up to the current time step — the SLT model may correct translation mistakes from earlier time steps. This means that the

[6] We found empirically that this works better than a duration of 26 seconds as for fixed-windows, with both increased translation quality and reduced flicker (see Appendix B).

final translation of a complete segment reaches the quality of offline translation. However, if these updated partial translations are presented to users and there are changes to previously translated text, this may be hard to follow. Therefore, it is important to not only evaluate the quality of the translations and the delay but also how often previously translated words are changed which is termed "flicker". Typically, when delay improves there will be more flicker because translating sooner means a higher chance of errors that need to be corrected in the next retranslation.

3.5 Window Merging Algorithm

One reason why a fixed-window segmentation might underperform compared to other segmentations is that utterances are likely to be split up into two or multiple segments which can introduce ambiguities and result in disfluent translations. However, this problem can be reduced if the windows are overlapping which is technically very easy to do. With a retranslation approach, we can simply shift the whole window by X seconds to obtain overlapping translations.

To merge the resulting translations, we employ a merging algorithm that was previously proposed for a cascaded SLT setup (Sen et al., 2022). Their merging window algorithm also works for end-to-end SLT because it is not dependent on a transcript of the source audio. The algorithm identifies the longest common substring (LCS) between the growing translation of the output stream and the translation of the current window. The current output is formed by everything to the left of the LCS coming from the output at the previous time step, followed by the LCS and then everything to the right of the LCS from the current translation output. In this way, the translation of the input stream is continuously extended.

The merging is controlled by a threshold that defines the minimum required length of the LCS. At every time step, this threshold is computed by:

$$threshold = |T_t| * \tau$$

Where T_t is the current window translation length and τ is a ratio hyperparameter. If the LCS is shorter than this minimum length, instead of merging the current translation with the output stream, the window is backtracked to the left and a longer window is translated. We backtrack 0.1 seconds at a time for a maximum of three backtracks. Only when a sufficiently long LCS is found or the maximum number of backtracks is reached, do we perform the merging operation. In our experiments, we set the ratio τ to 0.4 which performed best in the cascaded setup (Sen et al., 2022). If there are multiple LCS (common substrings with the same length), we merge at the last-occurring one.

3.6 Evaluation

For evaluation, we use SLTev[7] (Ansari et al., 2021), a toolkit that can evaluate translation quality, delay and flicker in a retranslation SLT setup. We explain below how the evaluation is adapted for unsegmented input. Since we assume our input is segmented at the talk level, we evaluate at the talk level too.

For **translation quality**, SLTev internally resegments the translations and aligns the new segments to the reference segments such that the word error rate is minimised (Matusov et al., 2005). It is not guaranteed that the new segments follow the sentence boundaries and are perfectly aligned but, as long as the introduced alignment errors are similar for different segmentations, they can be compared.

For **flicker**, we cannot use the sentence-level measure in SLTev because this is computed as an average over all segment-level flicker scores, and with different segmentations, this measure is not comparable. However, the document-level measure is evaluated independent of the segmentation and this works well for our purpose.

For **delay**, we do not use the official implementation in SLTev because of the way it assigns timestamps to repeated tokens. To explain the problem, consider the following example:

P	13.18	O
P	14.18	O horror,
P	15.18	O horror, terror, horror
C	16.18	O horror, horror, horror.

where we retranslate the newly available audio every second and consequently get three partial translations (P) and one final, complete translation (C). In SLTev, every token is assigned the time stamp of its type's first occurrence. This results in the following time stamp assignments with the original implementation.

O	horror	,	horror	,	horror	.
13.18	14.18	14.18	14.18	14.18	14.18	16.18

[7] https://github.com/ELITR/SLTev

All occurrences of "horror" and "," are assigned the timestamp 14.18 even though most of them are not yet generated by that time. If we translate longer segments that may be comprised of multiple sentences, encountering tokens that were already seen before becomes more and more likely. All of those would be assigned the timestamp of the first occurrence which favours longer segments (which we take to the extreme with our merged windows output stream). To solve this issue, we adapt the delay computation and store the individual timestamps for all repeated tokens. For this, we also need to be aware that previous content can change with each retranslation (e.g. terror → horror). We solve this following Arivazhagan et al. (2020b)'s notation of content delay and only assign timestamps once the previous context has finalised:

O	horror	,	horror	,	horror	.
13.18	14.18	14.18	16.18	16.18	16.18	16.18

With these new timestamps, all possible segmentations will receive the same delay if the translated text is identical and longer segments are no longer favoured in the SLTev delay calculation. However, since we wait until the context has finalised before we assign the time stamps, the new delay measure is now also affected by flicker.

4 Results

4.1 Translation Quality

We compare the different SLT models on different segmentations of the test sets and show the resulting translation quality of the complete segments in terms of BLEU in Table 2. Note that we would reach the same translation quality in an offline setting because the final retranslation is a translation of the full window, and in common with previous work, translation quality of online SLT is only measured on the final retranslation. We also evaluate with COMET (Rei et al., 2020) and report even better results with the merging windows approach but also find that COMET might be less reliable in a streaming SLT setup due to resegmentation errors (see Section D.1).

Does SHAS perform best with the original model (first column) as in previous work? When the SLT model is just trained on gold data, SHAS proves to be the best-performing segmentation out of all automatic segmentations which is in line with results by Tsiamas et al. (2022b) and Gaido et al. (2022). As in previous studies, we also find that the original model shows a considerable drop in BLEU when moving from the gold segmentation to automatically split segments.

		original	prefix	context	window
en-de	gold	25.4	25.5	25.2	25.5
	SHAS	24.5	23.9	24.9	24.8
	fixed	22.4	21.1	23.6	23.1
	merged	24.8	23.8	**25.3**	22.8
es-en	gold	41.6	41.3	41.1	41.4
	SHAS	40.2	40.3	40.7	41.0
	fixed	35.0	36.9	39.6	38.4
	merged	38.9	39.9	**42.0**	39.7
fr-en	gold	37.2	36.2	35.6	35.6
	SHAS	**36.2**	36.1	35.8	36.1
	fixed	31.0	32.0	34.5	32.9
	merged	34.6	35.2	35.8	31.9
it-en	gold	27.0	28.7	28.8	29.0
	SHAS	26.4	28.0	28.7	29.0
	fixed	22.5	25.6	27.5	26.3
	merged	25.3	27.4	**29.2**	27.6
pt-en	gold	30.6	29.5	28.7	29.1
	SHAS	**29.5**	28.9	29.2	28.6
	fixed	23.6	24.0	26.9	26.2
	merged	26.6	27.4	28.1	24.3

Table 2: BLEU scores with different SLT models (columns) and different audio segmentation methods (rows). Best result for *automatic* segmentation scenario marked in bold and green.

Is SHAS still the best-performing segmentation with the finetuned models? Finetuning with alternative segmentations can offer strong improvements for SHAS (+2.3) on it-en, with small improvements on es-en and en-de, but lower BLEU on pt-en and fr-en. Similarly, Gaido et al. (2022) found that SHAS did not benefit from finetuning on resegmented data. However, for the two segmentation approaches based on fixed windows, finetuning greatly reduces the gap to the gold segmentation.

This is especially noticeable when we finetune on context and prefixes (third column). This confirms the finding by Zhang et al. (2021) that context-

aware models can better translate artificially segmented audio. When merging overlapping windows, we consistently see an improvement over the segmentation with non-overlapping fixed windows. In three language pairs, this method outperforms SHAS and in two the context-finetuned model even improves over the gold segmentation.

Do training conditions need to match the segmentation at inference time? Apart from the context-aware finetuned model, we also finetuned a model on fixed windows of random duration (last column). This matches the fixed-window audio input at inference time better because a segment can start anywhere in an utterance, unlike the context-based model where every training segment started at the beginning of an utterance. Surprisingly, we find that the model finetuned on windows of random duration generally performs worse with the merging window strategy than the context-based model. This suggests that the training data for this model contains more misalignments between speech and translation because we extract both the start and the end of the segment via length ratios. This causes more flicker (see next Section) which makes it harder to merge the translations at each time step correctly. We leave extended experiments where the alignments between speech and translation are computed via ASR or the SLT output of the windows of random durations (as opposed to a simple length ratio) to future work.

4.2 Flicker

As mentioned in the Introduction, translation quality is not the only important evaluation metric in an online SLT scenario. When using a retranslation approach, we also need to consider the flicker that is caused by the model updating its translations at every time step. We compute the flicker as described in Section 3.6. The flicker scores for the Spanish-to-English test set can be seen in Figure 2, the same figures for the other language pairs are in Appendix D. For the results shown here, we use an output mask of 0. We show in Appendix C that our findings also hold with larger output masks. We show scores with and without biased beam search.

Biased beam search (Arivazhagan et al., 2020a) is a modification to regular beam search that biases the probability distribution at the current time step towards a token in a given prefix translation at the same timestep. This can be used to stabilise retranslation – the translation of the current prefix is biased towards the translation of the previous prefix, suppressing flicker. In our experiments, we use the translation of the previous step as the prefix with a beta parameter of 0.25 and mask the 5 last tokens such that changes towards the end of the sentence are still possible[8]. Biased beam search cannot be applied directly to the merged window approach, since it depends on an alignment between the translation of the current prefix and that of the previous prefix. When translating using sliding windows, the current and previous prefixes have different start points, so their translations cannot be easily aligned. We experimented with a way to reduce flicker by merging on the last common substring rather than the longest but this causes considerable translation quality loss (see Appendix B).

Does the segmentation strategy matter for flicker? From Figure 2, we can see that there are big differences between the different segmentation strategies. Fixed windows have the highest flicker because there we translate the longest windows. If something at the beginning of the window translation is changed, this will increase the flicker score considerably. With biased beam search, the flicker can be dramatically reduced. Merging overlapping windows has a lower flicker than fixed windows without biased beam search, both because the duration of the windows is shorter and because the merging algorithm prohibits changes to the left of the longest common substring[9]. This segmentation method even has lower flicker than SHAS when no biased beam search is applied. With biased beam search, SHAS performs mostly similar to the gold segmentation which has the lowest flicker overall.

Does model finetuning help reduce flicker? Prefix finetuning helps reduce flicker both with and without biased beam search because the models see incomplete sentences at training time and are less likely to hallucinate to finish the sentence. Context finetuning helps even more and we saw in the outputs that this model has less of a tendency to connect multiple sentences into a longer sentence which can reduce flicker. The model finetuned on windows shows an even higher flicker than the original model for most segmentation strategies even though it was designed to be able to translate seg-

[8] We do not show translation quality scores with biased beam search because on average there is only a difference of -0.006 BLEU.

[9] Reducing the window length to 15 seconds for the fixed window segmentation reaches a flicker that is only slightly higher than for merged windows but the translation quality suffers considerably.

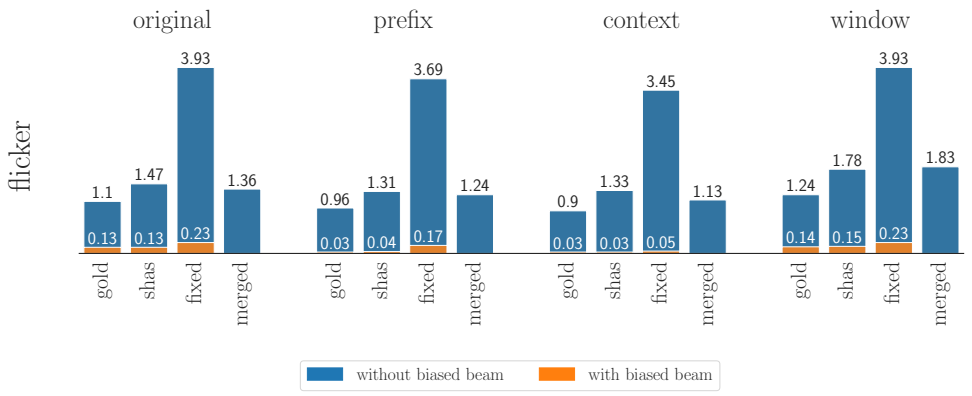

Figure 2: Flicker values for the different segmentation strategies and SLT models on the Spanish-to-English test set. The results are grouped by training strategy and each bar corresponds to a different segmentation strategy. We do not apply biased beam search to the merged segmentation.

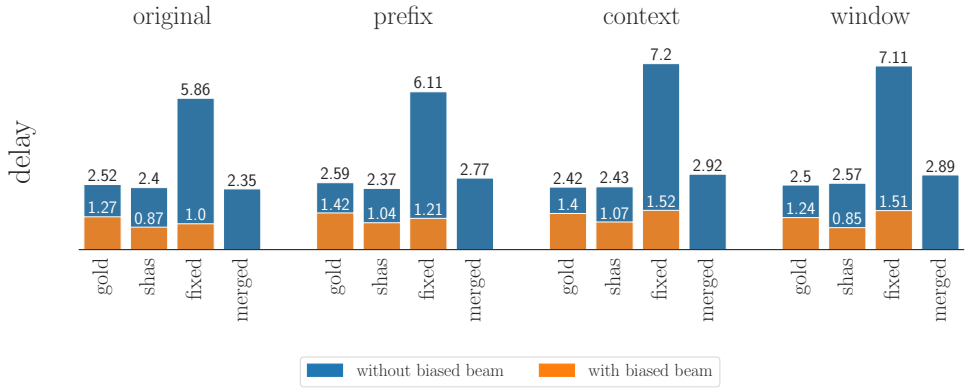

Figure 3: Delay values for the different segmentation strategies and SLT models on the Spanish-to-English test set. The results are grouped by training strategy and each bar corresponds to a different segmentation strategy.

ments that can start and end anywhere in a sentence. As discussed in the previous section, we think this increased flicker is an artefact of the automatically generated training data which can be erroneous.

4.3 Delay

The final evaluation metric we consider is delay. The results can be seen in Figure 3. Again, we show results with and without biased beam search for the gold, SHAS and fixed-window segmentation.

Does the segmentation strategy matter for delay? Because our definition of delay is affected by flicker as well (see Section 3.6), the fixed segmentation without biased beam search not only has the highest flicker but also the highest delay. In our results, we can see that the high delay is caused by the flicker because when we reduce flicker with biased beam search the fixed segmentation has comparable delay to the gold and SHAS segmentations. The merging windows approach has comparable delay to the gold and SHAS segmentations without biased beam search. Since we cannot apply biased beam search reliably to the merging windows approach without hurting translation quality, the flicker cannot be reduced and therefore, the merging windows approach has higher delay than the other segmentation methods with biased beam search. If delay could be defined independently of flicker in a way that still works for comparing different segmentations, the merging windows approach would likely have similar delay also compared to the outputs with biased beam search.

Does model finetuning help reduce delay? The results are a bit mixed. For example, the context model reduces delay for the gold segmentation but increases it slightly for SHAS and more for the fixed segmentation and the merging windows approach. In general, the choice of the model does not seem to be as important for delay as for translation quality and to a lesser extent flicker. It is possible that apparent effects only occur because our definition of delay is affected by flicker.

5 Discussion

Based on our results in Section 4, we believe that fixed-window segmentation should not be disregarded in future SLT research on unsegmented audio. Given the right setup with a context-aware model and a merging window algorithm, this segmentation can outperform current state-of-the-art automatic segmentation models and in some cases even the gold segmentation in terms of translation quality. Moreover, in an online SLT setup, a fixed-window approach brings the additional benefit that no dedicated segmentation model needs to be loaded at inference time and run every time new audio becomes available.

While there is currently no solution to bring flicker down to biased beam search levels without hurting quality (see Appendix B) or increasing delay (see Appendix C), this should not be a reason to disregard fixed-window segmentation as it opens exciting opportunities for future research.

6 Conclusion

In this paper, we explored several combinations of segmentation-robust finetuning and different automatic segmentation strategies in an online SLT setup. We focus on a retranslation-based approach to SLT and we run experiments on five different language pairs based on two different SLT models. Considering the evaluation of translation quality, flicker and delay, we discuss several issues that arise when comparing different segmentations and propose a fix to an existing toolkit for evaluating delay. Our results show that a simple fixed-window segmentation can perform surprisingly well if an algorithm is used for merging overlapping windows and a context-aware SLT model is used. In terms of translation quality, this segmentation performs comparably to SHAS — the current state-of-the-art segmentation method — and in some cases even outperforms the gold segmentation, showing potential for future application to offline SLT. In terms of flicker and delay, the results of the merging windows approach are comparable to the other segmentations if biased beam search is not enabled but future work is needed to reduce flicker in the merging windows approach to similar levels as biased beam search for other strategies without hurting translation quality.

Ethical Considerations

In our work, we only use publicly available model checkpoints, toolkits and datasets and do not collect any additional data. Our experiments also do not involve human annotators.

Limitations

While we aim to evaluate on a number of language pairs and with different automatic metrics, there are still some open questions that we could not answer in this work. First, we did not perform a human evaluation and, therefore, it remains unclear how distracting the different flicker and delay values with different setups would be for a user. However, previous work by Macháček and Bojar (2020) shows that character erasure - a metric related to flicker - correlates with usability scores in a human evaluation which suggests that this would also be true for flicker. Second, the current implementation of SHAS can be used to simulate an online setting but it still expects the full audio as input. Consequently, we could not empirically compare how long translation takes with different segmentation methods in a real online setup. Third, our experiments are limited to SLT using a retranslation strategy. We leave further experiments with simultaneous SLT models that use a policy to decide at each time step whether to wait for further input or to translate for the future.

Acknowledgements

We thank Sukanta Sen and Ioannis Tsiamas who shared their code with us for the window merging algorithm and the pSTREAM implementation of SHAS, respectively. We are also grateful to Noëmi Aepli and Amit Moryossef and the anonymous reviewers for their valuable feedback. This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements (project ELITR; no. 825460 and project GoURMET; no. 825299), and from the Swiss National Science Foundation (project MUTAMUR; no. 176727). All experiments made use of the infrastructure services provided by S3IT, the Service and Support for Science IT team at the University of Zurich.

References

Antonios Anastasopoulos, Loïc Barrault, Luisa Bentivogli, Marcely Zanon Boito, Ondřej Bojar, Roldano

Cattoni, Anna Currey, Georgiana Dinu, Kevin Duh, Maha Elbayad, Clara Emmanuel, Yannick Estève, Marcello Federico, Christian Federmann, Souhir Gahbiche, Hongyu Gong, Roman Grundkiewicz, Barry Haddow, Benjamin Hsu, Dávid Javorský, Věra Kloudová, Surafel Lakew, Xutai Ma, Prashant Mathur, Paul McNamee, Kenton Murray, Maria Nădejde, Satoshi Nakamura, Matteo Negri, Jan Niehues, Xing Niu, John Ortega, Juan Pino, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Yogesh Virkar, Alexander Waibel, Changhan Wang, and Shinji Watanabe. 2022. Findings of the IWSLT 2022 evaluation campaign. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 98–157, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alexander Waibel, Changhan Wang, and Matthew Wiesner. 2021. FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 1–29, Bangkok, Thailand (online). Association for Computational Linguistics.

Ebrahim Ansari, Amittai Axelrod, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, Alexander Waibel, and Changhan Wang. 2020. FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34, Online. Association for Computational Linguistics.

Ebrahim Ansari, Ondřej Bojar, Barry Haddow, and Mohammad Mahmoudi. 2021. SLTEV: Comprehensive evaluation of spoken language translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 71–79, Online. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Te I, Wolfgang Macherey, Pallavi N. Baljekar, and George F. Foster. 2020a. Re-translation strategies for long form, simultaneous, spoken language translation. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7919–7923.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020b. Re-translation versus streaming for simultaneous translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 220–227, Online. Association for Computational Linguistics.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Eunah Cho, Jan Niehues, and Alex Waibel. 2017. NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation. *Proceedings of Interspeech*, pages 2645–2649.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

Marco Gaido, Mattia A. Di Gangi, Matteo Negri, Mauro Cettolo, and Marco Turchi. 2020. Contextualized Translation of Automatically Segmented Speech. In *Proc. of Interspeech 2020*, pages 1471–1475.

Marco Gaido, Matteo Negri, Mauro Cettolo, and Marco Turchi. 2021. Beyond voice activity detection: Hybrid audio segmentation for direct speech translation. In *Proceedings of the Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021)*, pages 55–62, Trento, Italy. Association for Computational Linguistics.

Marco Gaido, Sara Papi, Dennis Fucci, Giuseppe Fiameni, Matteo Negri, and Marco Turchi. 2022. Efficient yet competitive speech translation: FBK@IWSLT2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 177–189, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Daniel Li, Te I, Naveen Arivazhagan, Colin Cherry, and Dirk Padfield. 2021. Sentence boundary augmentation for neural machine translation robustness. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Dominik Macháček and Ondřej Bojar. 2020. Presenting simultaneous translation in limited space. In *Proceedings of the 20th Conference Information Technologies - Applications and Theory (ITAT 2020)*.

Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. 2005. Evaluating machine translation output with automatic sentence segmentation. In *Proceedings of the Second International Workshop on Spoken Language Translation*, Pittsburgh, Pennsylvania, USA.

Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, and Alexander H. Waibel. 2016. Dynamic transcription for low-latency speech translation. In *Proceedings of Interspeech*.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-Latency Neural Speech Translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Hyderabad, India.

Siqi Ouyang, Rong Ye, and Lei Li. 2022. On the impact of noises in crowd-sourced data for speech translation. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 92–97, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2021. Dealing with training and test segmentation mismatch: FBK@IWSLT2021. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 84–91, Bangkok, Thailand (online). Association for Computational Linguistics.

Ngoc-Quan Pham, Tuan Nam Nguyen, Thai-Binh Nguyen, Danni Liu, Carlos Mullov, Jan Niehues, and Alexander Waibel. 2022. Effective combination of pretrained models - KIT@IWSLT2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 190–197, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Tomasz Potapczyk and Pawel Przybysz. 2020. SR-POL's system for the IWSLT 2020 end-to-end speech translation task. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 89–94, Online. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W. Oard, and Matt Post. 2021. Multilingual tedx corpus for speech recognition and translation. In *Proceedings of Interspeech*.

Sukanta Sen, Ondřej Bojar, and Barry Haddow. 2022. Simultaneous translation for unsegmented input: A sliding window approach. arXiv preprint 2210.09754.

Akshaya Shanbhogue, Ran Xue, Ching-Yun Chang, and Sarah Campbell. 2022. Amazon Alexa AI's system for IWSLT 2022 offline speech translation shared task. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 169–176, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Jongseo Sohn, Nam Soo Kim, and Wonyong Sung. 1999. A statistical model-based voice activity detection. *IEEE Signal Processing Letters*, 6(1):1–3.

Yun Tang, Hongyu Gong, Xian Li, Changhan Wang, Juan Pino, Holger Schwenk, and Naman Goyal. 2021a. FST: the FAIR speech translation system for the IWSLT21 multilingual shared task. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 131–137, Bangkok, Thailand (online). Association for Computational Linguistics.

Yun Tang, Juan Pino, Xian Li, Changhan Wang, and Dmitriy Genzel. 2021b. Improving speech translation by understanding and learning from the auxiliary text translation task. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4252–4261, Online. Association for Computational Linguistics.

Yun Tang, Juan Miguel Pino, Changhan Wang, Xutai Ma, and Dmitriy Genzel. 2021c. A general multi-task learning framework to leverage text data for speech to text tasks. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6209–6213.

Ioannis Tsiamas, Gerard I. Gállego, Carlos Escolano, José Fonollosa, and Marta R. Costa-jussà. 2022a. Pretrained speech encoders and efficient fine-tuning methods for speech translation: UPC at IWSLT 2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 265–276, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Ioannis Tsiamas, Gerard I. Gállego, José A. R. Fonollosa, and Marta R. Costa-jussà. 2022b. Shas: Approaching optimal segmentation for end-to-end speech translation. In *Proceedings of Interspeech*.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. 2021. Beyond sentence-level end-to-end speech translation: Context helps. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2566–2578, Online. Association for Computational Linguistics.

Appendix

A Further Finetuning Specifications

We finetune all models and translate with a single NVIDIA Tesla V100 GPU. For the multilingual mTEDx model, the additional parameter `load-speech-only` needs to be added to the official training script[10]. We use the `restore-file` parameter to specify the checkpoints of the original models from which continued training should be initialised.

We will release all code (training scripts, translation scripts and evaluation modifications), the finetuned model checkpoints and the outputs upon publication.

B Experiments with Last Common Subsequence

As a possible way of reducing flicker for the merging windows approach, we try merging on the last common subsequence (longer than two tokens) instead of the longest common subsequence. In this way, we can maximise the finalised part of the growing output translation and reduce flicker. Figure 4 shows how the flicker increases for both merging strategies when the window size increases. With the original implementation that merges on the longest common subsequence, the flicker increases dramatically when the window size is increased. For the modified merging algorithm that merges on the last subsequence (longer than two tokens) the flicker increases only moderately with increased window size and is in general much lower.

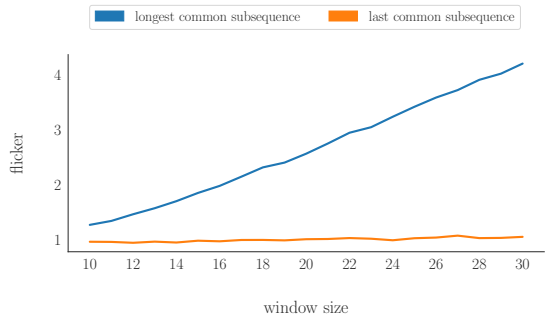

Figure 4: Flicker values with the original model on the English-to-German test set for different window sizes when merging on the longest common subsequence (blue) and the last common subsequence (orange).

[10] https://github.com/facebookresearch/fairseq/blob/main/examples/speech_text_joint_to_text/docs/iwslt2021.md

Based on these results, one might choose to merge on the last sequence, however, this change also affects the translation quality. Figure 5 shows the BLEU scores of both merging methods with different window sizes. Unfortunately, merging on the last common subsequence performs continuously worse than merging on the longest common subsequence. If quality is the main focus, this merging method is not advisable. These results also show that a window size of 15 performs best for the merging windows approach.

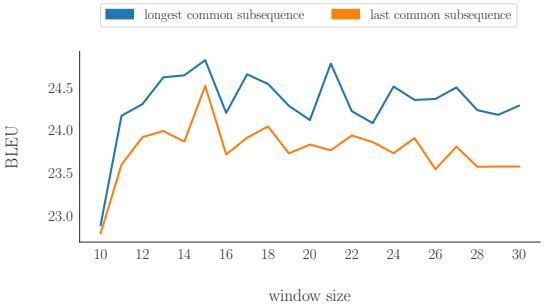

Figure 5: BLEU scores with the original model on the English-to-German test set for different window sizes when merging on the longest common subsequence (blue) and the last common subsequence (orange).

C Results with Output Mask

We also evaluate the four different segmentation methods when an output mask is applied. This means at every time step the output is truncated from the right. The number of tokens that are removed is defined by the mask size, i.e. a mask of size 0 means no tokens are removed and a mask of size 7 means seven tokens are removed. We compute these results for Spanish-to-English without biased beam search and the context-aware model which showed the lowest flicker in general.

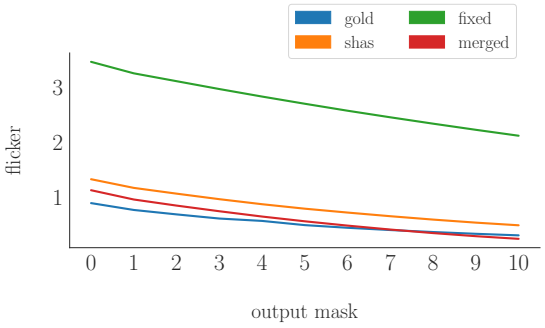

Figure 6: Flicker with different output masks on the Spanish-to-English test set. Results for all four segmentation methods with the context-finetuned model.

Figure 6 shows the flicker at different output mask sizes. First of all, it can be noticed that the fixed window segmentation has a continuously higher flicker than all other segmentation methods and that the flicker is still rather large even with a mask of size 10. This suggests that most flicker in the fixed-window segmentation does not occur towards the end of the segments.

The merging windows approach consistently has lower flicker than SHAS and with larger mask sizes even lower flicker than the gold segmentation. With a mask of size 10, the flicker is at 0.25 which is comparable to the flicker of the original model with fixed window segmentation where biased beam search is enabled.

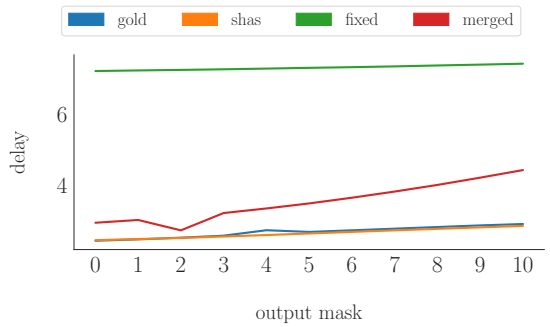

Figure 7: Delay with different output masks on the Spanish-to-English test set. Results for all four segmentation methods with the context-finetuned model.

		original	prefix	context	window
en-de	gold	-0.0589	-0.0801	-0.0659	-0.0696
	SHAS	-0.1762	-0.1934	-0.0835	-0.1418
	fixed	-0.3080	-0.3655	-0.1846	-0.1671
	merged	-0.1821	-0.2169	**-0.0683**	-0.1133
es-en	gold	0.3175	0.2864	0.2776	0.2981
	SHAS	0.2145	0.2291	0.2784	0.2638
	fixed	-0.0339	0.0448	0.2637	0.2633
	merged	0.2658	0.2736	**0.3962**	0.3642
fr-en	gold	0.1702	0.1380	0.1123	0.1078
	SHAS	0.1147	0.1421	0.1742	0.134
	fixed	-0.1696	-0.1115	0.0777	0.0316
	merged	0.0978	0.1066	**0.2170**	0.1109
it-en	gold	0.0566	0.0583	0.0704	0.0886
	SHAS	-0.012	0.0215	0.0915	0.0709
	fixed	-0.305	-0.2072	0.0142	-0.0408
	merged	-0.0536	-0.0066	**0.1255**	0.0619
pt-en	gold	0.0662	0.0234	-0.0130	-0.0048
	SHAS	-0.0108	-0.0104	-0.0085	-0.0085
	fixed	-0.2939	-0.2853	-0.0854	-0.0937
	merged	-0.0581	-0.0784	**0.0276**	-0.0554

Table 3: COMET scores with different SLT models (columns) and different audio segmentation methods (rows). Best result for *automatic* segmentation scenario marked in bold and green.

However, this reduced flicker comes at the cost of a higher delay because the masked tokens will not be available at the time they are actually produced. This flicker-delay trade-off is well-known. Figure 7 shows the increase in delay with larger output masks. For the merging windows approach, we see that the delay increases more than for SHAS and the gold segmentation. Since our definition of the delay measure is affected by flicker, these results are hard to interpret. Nevertheless, using an output mask is a way to reduce flicker for the merging windows approach without reducing translation quality but we need to accept a higher delay.

D Additional Results

D.1 Translation Quality with COMET

For completeness, we present performance results measured with COMET (Rei et al., 2020) in Table 3. This is evaluated outside of SLTev but we use the same resegmentation tool (Matusov et al., 2005) to align the translations with the reference segments. The results show similar patterns as with BLEU and the context model paired with the merging window approach performs best among the automatic segmentation approaches on all language pairs. This approach even outperforms the gold segmentation on three language pairs. Note however that evaluating resegmented text with COMET may have some undesirable side-effects because the translated text is not always split at correct segmentation boundaries, e.g. the first token of a segment often is glued to the end of the previous segment.

We tested this with 200 gold segments for en-de and manually corrected the resegmentation of the original model output. While the BLEU score does not change much with these corrections (24.70 vs. 24.73), the COMET score jumps from -0.1128 to 0.0467 which is a larger improvement than some differences in Table 3. Since it is unclear if such resegmentation errors occur equally often in all our experiment setups, we only include the results with BLEU in the main body of the paper. We hypothesise that COMET has only seen well-formed sentences at training time and consequently is less reliable on such resegmented data. In the future,

document-level neural evaluation metrics could be better suited for evaluating translations of unsegmented or automatically segmented audio in SLT.

D.2 Flicker Results for Other Language Pairs

We present the same plots as in Section 4.2 for English-to-German in Figure 8, French-to-English in Figure 9, Italian-to-English in Figure 10 and Portuguese-to-English in Figure 11. The results follow the same patterns as the results for Spanish-English discussed in Section 4.2:

- Fixed windows without biased beam search have the highest flicker.
- For the language pairs into English, the merging windows approach has lower flicker than SHAS if no biased beam search is used.
- Finetuning on context reduces flicker.

D.3 Delay Results for Other Language Pairs

We present the same plots as in Section 4.3 for English-to-German in Figure 12, French-to-English in Figure 13, Italian-to-English in Figure 14 and Portuguese-to-English in Figure 15. The results follow the same patterns as the results for Spanish-English discussed in Section 4.3:

- Fixed windows without biased beam search have the highest delay.
- The merging windows approach has comparable delay to SHAS if no biased beam search is used.
- Finetuning has less of an effect on delay than on flicker.

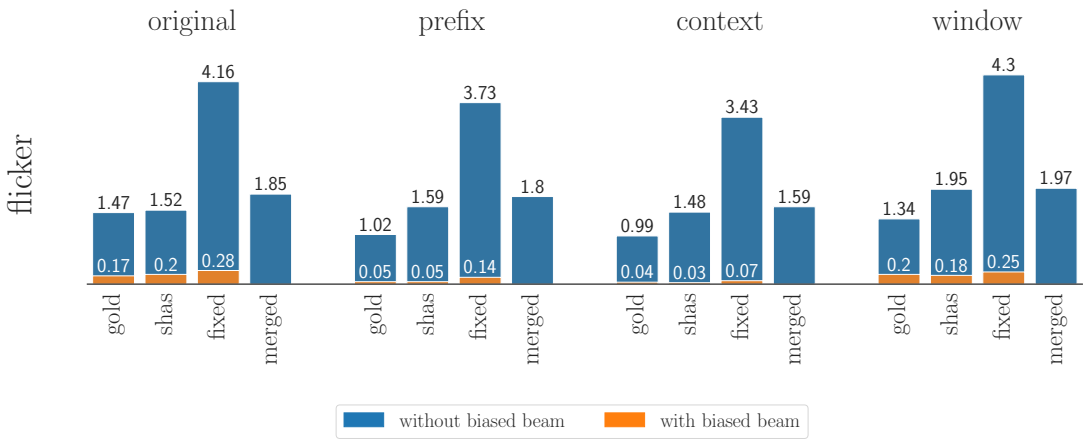

Figure 8: Flicker values for the different segmentation strategies and SLT models on the English-to-German test set.

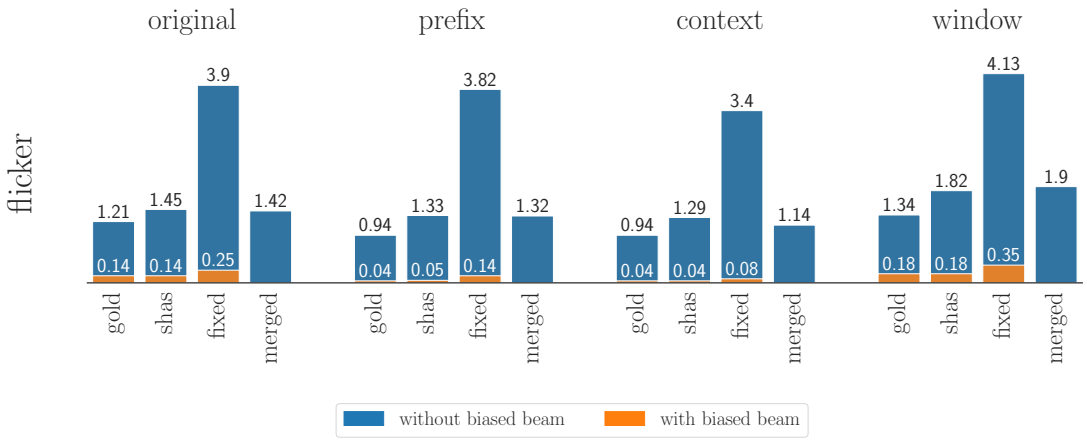

Figure 9: Flicker values for the different segmentation strategies and SLT models on the French-to-English test set.

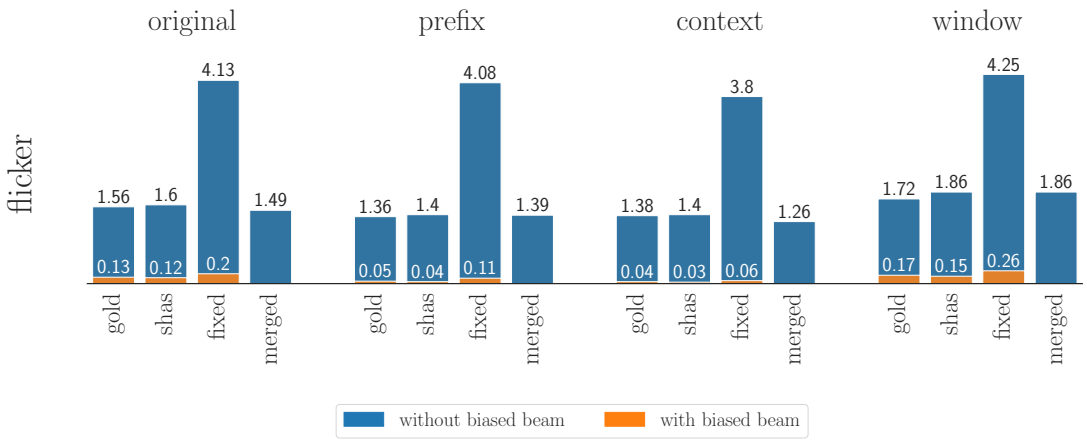

Figure 10: Flicker values for the different segmentation strategies and SLT models on the Italian-to-English test set.

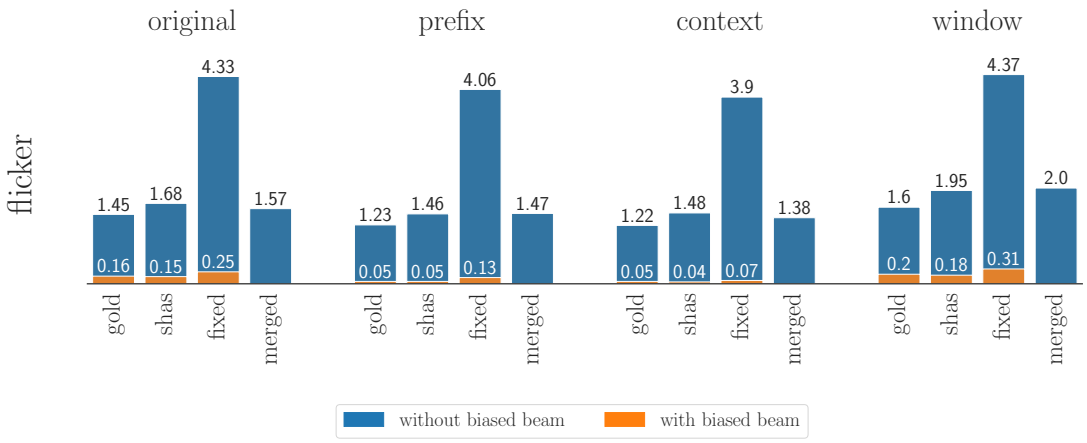

Figure 11: Flicker values for the different segmentation strategies and SLT models on the Portuguese-to-English test set.

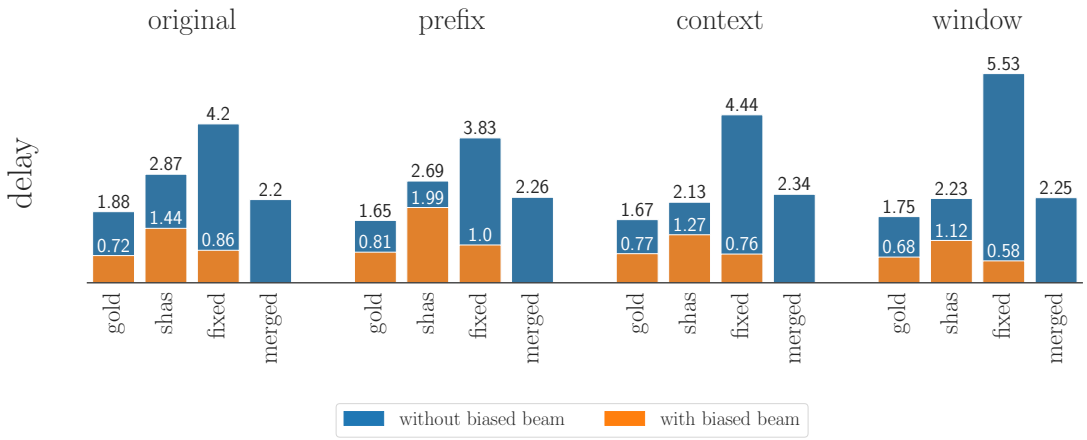

Figure 12: Delay values for the different segmentation strategies and SLT models on the English-to-German test set.

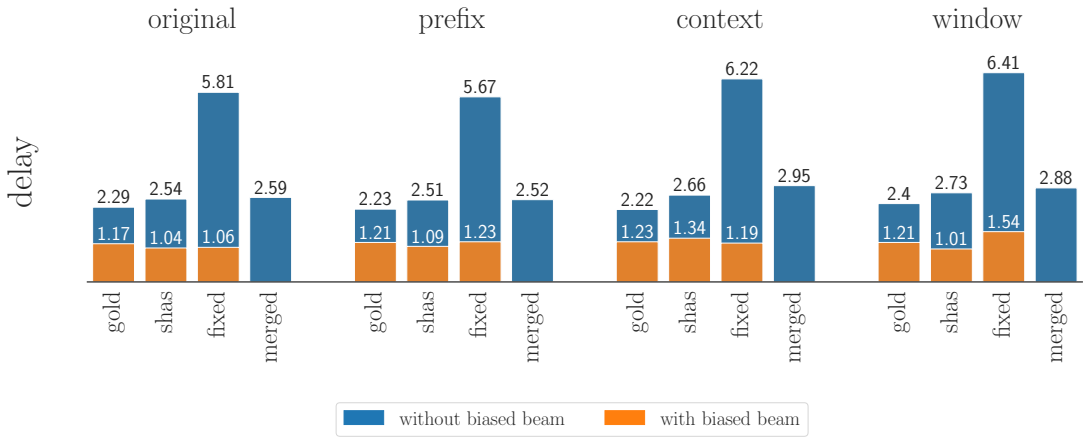

Figure 13: Delay values for the different segmentation strategies and SLT models on the French-to-English test set.

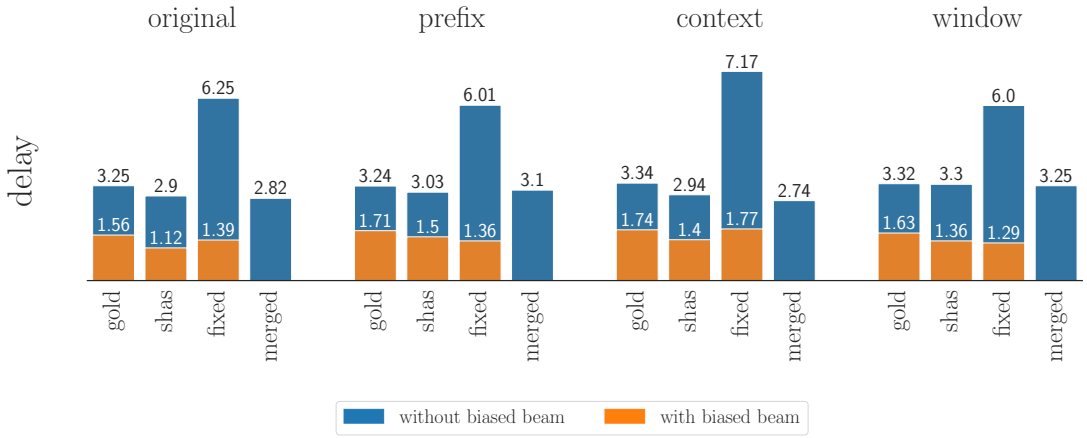

Figure 14: Delay values for the different segmentation strategies and SLT models on the Italian-to-English test set.

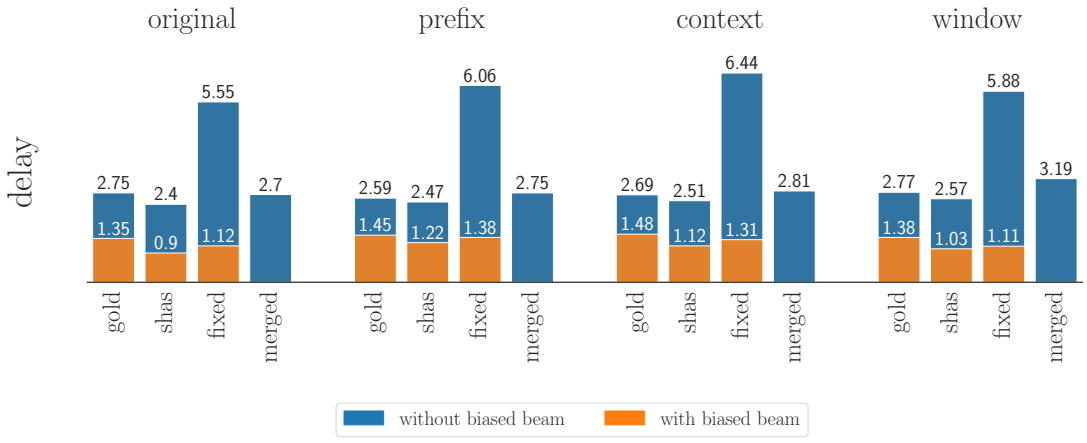

Figure 15: Delay values for the different segmentation strategies and SLT models on the Portuguese-to-English test set.