SLTEV: Comprehensive Evaluation of Spoken Language Translation

Ebrahim Ansari
Charles University MFF ÚFAL
and IASBS

Ondřej Bojar
Charles University
MFF ÚFAL

Barry Haddow
University of Edinburgh

Mohammad Mahmoudi
IASBS

surname@ufal.mff.cuni.cz except bhaddow@ed.ac.uk

Abstract

Automatic evaluation of Machine Translation (MT) quality has been investigated over several decades. Spoken Language Translation (SLT), especially when simultaneous, needs to consider additional criteria and does not have a standard evaluation procedure and a widely used toolkit. To fill the gap, we introduce SLTEV, an open-source tool for assessing SLT in a comprehensive way. SLTEV reports the quality, latency, and stability of an SLT candidate output based on the time-stamped transcript and reference translation into a target language. For quality, we rely on sacreBLEU which provides MT evaluation measures such as chrF or BLEU. For latency, we propose two new scoring techniques. For stability, we extend the previously defined measures with a normalized Flicker in our work. We also propose a new averaging of older measures.

A preliminary version of SLTEV was used in the IWSLT 2020 SHARED TASK. Moreover, a growing collection of test datasets directly accessible by SLTEV are provided for system evaluation comparable across papers.

1 Introduction

Spoken Language Translation (SLT), i.e. translation of human speech across languages, is an application at least as important as Machine Translation (MT). Many approaches have been examined so far, ranging from translation of transcript chunks (Fügen et al., 2008; Bangalore et al., 2012) to fully end-to-end, speech-to-speech neural systems, (Jia et al., 2019). In recent years, simultaneous translation systems aim at behavior similar to human interpreters, digesting and producing an infinite sequence of words. Some systems (Grissom II et al., 2014; Gu et al., 2017; Arivazhagan et al., 2019b; Press and Smith, 2018; Xiong et al., 2019; Ma et al., 2019; Zheng et al., 2019) do not consider any revision of their outputs and can be evaluated in two main criteria: quality and latency, allowing users to trade a bigger delay (including waiting for more input text) for a more accurate translation. Simultaneous translation systems aimed at automatic subtitling (Niehues et al., 2016; Müller et al., 2016; Niehues et al., 2018; Arivazhagan et al., 2019a) may revise their outputs, demanding a new evaluation measure: the stability, i.e. the amount of revision. Trading these qualities for one another is again possible: It is obvious that if a system creates the translations with a longer Delay or revises them more (higher Flicker), the quality of the final translation (i.e., the output text) can be better. Given the existence of three evaluation criteria and a multitude of possible definitions for each of them, the need for some robust and standard metrics to evaluate SLT is inevitable.

Recently, the MT community tackled a similar problem (i.e., the inconsistency in the reporting of BLEU scores) by introducing a tool named sacreBLEU (Post, 2018) with a canonical implementation of the widely user metric. In this work, we propose SLTEV,[1] an open-source tool to calculate the quality of SLT systems based on three different criteria: translation quality, latency, and stability, in a standardized way. Furthermore, we complement SLTEV with a growing collection of freely-available test sets for Automatic Speech Recognition (ASR), MT and SLT for a number of languages, so that these technologies can be evaluated in comparable settings, similarly to what the WMT news test sets (Barrault et al., 2020) offer in MT.

2 Related Work

SLTEV is designed to be versatile enough to score automatic SLT as well as transcribed human interpretation. Shimizu et al. (2014) are probably the first to score human interpretation with automatic

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

P	0	50	Good	P	60	0	50	Gut
P	0	65	Good mor	P	80	0	65	Guten Morgen!
P	0	119	Good morning how	P	130	0	119	Guten wie morgen
P	0	195	Good morning. How are you?	P	201	0	195	Guten Morgen! Wie geht es dir?
C	0	102	Good morning.	C	201	0	102	Guten Morgen!
P	102	218	How are you? I	P	220	102	218	Wie geht es dir? Ich
C	102	195	How are you?	C	220	102	195	Wie geht es dir?
P	195	239	I am	P	245	195	239	Ich bin
...	...	...	...	...	...	...	...	...
(a) Time-stamped transcript				(b) SLT candidate output				

Figure 1: Example of SLTEV file formats. All timestamps in centiseconds.

measures but they segment the output manually and assess only the quality using BLEU (Papineni et al., 2002), WER (Matusov et al., 2005), TER (Snover et al., 2006), and RIBES (Isozaki et al., 2010).

Most SLT evaluations require sentence segmentation of the candidate to match the reference one. Using *mwerSegmenter* (Matusov et al., 2005), they re-segment the candidate automatically, minimizing WER against the reference. We complement this approach with time-based segmentation.

Niehues et al. (2016) introduced the retranslation approach to simultaneous SLT, and define latency based on the time between a word expected and actually displayed, considering only the final version of the word, not early revisions. They did not provide any evaluation of stability. However, in follow-up work (Niehues et al., 2018) they assess the level of SLT stability (i.e., the number of corrections) by measuring the overlap between consecutive updates. As soon as a word is changed, all the following words are counted as updated, suggesting than any word change forces the user to reread all the rest.

Gu et al. (2017) consider two versions of delay when assessing their reinforcement learning based simultaneous SLT model: Average Proportion (of waiting compared to producing words) and Consecutive Wait (the silence duration so far), and prescribe a target value for each of them to steer the learning, also balancing it with quality estimated by smoothed BLEU (Lin and Och, 2004). Since their model does not allow corrections, they do not require a measure of stability.

The delay measures of (Gu et al., 2017) were criticised by Ma et al. (2019) when they introduced their wait-k model. They defined a measure Average Lag (AL), which measures how far, in words, the translation is behind an ideal wait-k model. Since wait-k does not allow corrections, they do not need a stability measure. AL was improved by Cherry and Foster (2019), with Differentiable Average Lag (DAL), which not only is differentiable, but fixes some undesirable behaviour of AL around sentence boundaries. However DAL, like AL, is defined in terms of word count and on segmented text (although it could be extended to fix these shortcomings).

Arivazhagan et al. (2019a) extended the retranslation model of Niehues et al. (2016), and so needed a measure of stability. For evaluation, they check the output of ASR and the output of the MT system as recorded over time in their simple logging system. They assess the quality, latency, and stability (i.e., Flicker). The quality is estimated using BLEU after *mwerSegmenter* re-segmentation. For the assessment of latency (translation lag) and stability (the number of erased tokens in temporary translations per final target token, "Normalized Erasure" in the paper), they do not use any segmentation at all and instead calculate the scores for ten-minute long audio chunks.

The closest to our work is *SIMULEVAL* (Ma et al., 2020), a client-server toolkit measuring the latency of SLT including any network effects between the evaluated system (client) and the mock user (SIMULEVAL server). SIMULEVAL offers a nice visualization interface but the required client-server approach may be unsuitable for research prototypes solving SLT only partially. Most importantly, updates of output (Flicker) are not supported and no test set for reproducible scoring is provided.

3 Input Formats

SLTEV can evaluate separate ASR and MT systems as well as cascaded and end-to-end SLT systems. We focus on SLT here. Three input files are used for SLT evaluation: a time-stamped golden transcript in the source language (Section 3.1), a reference translation (or translations in multi-reference format; Section 3.2) and candidate output (Section 3.3) in the target language. The intermediate ASR output can be provided as the fourth input file to calculate the accuracy of the ASR system if it was part of the cascade (Section 3.3).

3.1 File Format of Time-Stamped Transcript

Time-stamped transcript files (golden transcript and ASR output, both in the source language) are line-oriented text files. The lines contain gradually growing "partial" (P) segments, until the segment is "completed" (C);[2] see Figure 1 (a). All lines are equipped with timestamps measured in centiseconds from the start of the sound file: the *start time* and *end time* of the given segment.

Partial segments add one or more words at once, sub-word updates are possible but SLTEV is not ready for them. We expect that a well-aligned transcript file (such as the golden reference) should have the exact same number of completed (C) segments as the reference translation file.

3.2 File Format of the Reference Translation

Each line of the reference translation file shows the translation of the corresponding complete (C) line in the time-stamped transcript file. The number of lines in the reference translation thus equals the number of C lines in the time-stamped transcript.

3.3 File Format of ASR, MT or SLT Output

An ideal SLT system will report all the details as in Figure 1 (b): partial (P) vs. complete (C) flag, the *display time* when the segment was produced and the *start time* and *end time* indicating the time span supposedly containing the given message. Partial segments allow the user to provide finer timing information and to provide revisions of outputs,[3] trading lower Delay for higher Flicker.

For the output of ASR, the segment is in the source language. For the output of MT as the second step or the output of end-to-end SLT, the segment is in the target language but the start and end timestamps should reflect the span in the *original* sound, i.e., when the source was uttered.

Again, "P"artial candidates allow for revising the output so far, and the "C"omplete candidates are required. The concatenation of all the (C) segments corresponds to the whole document but their number and segmentation may differ from the reference one. If the ASR, MT or SLT outputs lack some of the timestamp information, zeros should be used for format consistency. SLTEV then calculates limited results based on the provided information.

[2]We use the term "segment" for generality but typically, completed segments correspond to sentences. Usually, a sentence ends with a punctuation mark.

[3]Revisions do not occur in golden transcripts but we want the same format to suit both golden and candidate outputs.

4 Proposed Metrics

In this section, all evaluation metrics and strategies introduced in our evaluation framework are described. Our evaluations are based on three criteria: latency, stability, and the quality of MT outputs.

4.1 Delay to Assess Latency

In our point of view, *latency* should reflect the delay with which the recipient receives the message from the sender. Words are reasonable smallest units that the message can be broken into but there is not a 1-to-1 correspondence between source and target words. Ideally, we would know the alignment between the source words and the words in the candidate translation, and we would have exact timing information for both.

Defining latency as the sum of how long we had to wait for a target word given the time of the corresponding source word[4] would render the values dependent on the language pair in question. When translating, e.g., from English into German, all verbs in subordinate clauses would increase the value of latency because they simply have to appear at the end of the German clause, so the recipient receives them much later than their English source verb was uttered. We thus focus on the extra delay beyond what the language pair implies.

We propose two approaches to delay calculation. Both measure the difference between the time that a target word was displayed and an estimate of when it should have been displayed but differ in estimating the expected display time:[5] The first one is proportional while the second one uses automatic word alignment between the source and reference translations to account for word order differences across languages, see Sections 4.2.1 and 4.2.2 below.

Both approaches produce a two-dimensional matrix $T(i, j)$ storing the expected time of the jth word of the reference sentence i.

[4]In this calculation, we only include reference words which also appear in the SLT output, because they are the only ones that contribute meaningfully to the delay. A reference word which never appears in the SLT output has an infinite delay, and a word appearing in the output but not in the reference is, well, unexpected, so no delay makes sense. We acknowledge that this design decision brings the risk of gaming Delay by producing words different from the reference; this would however lead to a clear loss in Quality.

[5]If the reference translation was also time-stamped at the word level, this estimate would be easier to make but we do not assume that. We are however experimenting with reference *interpretation*, where a human interpreter produces translation in time. This exploration is left for future.

Given T, our Delay is calculated by summing differences between the expected word emission time in T and the reported emission time in the segments of the SLT candidate output. If the SLT system predicts the word earlier than its expected time, its delay is set to zero, not negative.

4.2 Segmentation Strategies

Note that delay calculation operates on the individual *segments* of the transcript. We use two segmentation strategies to re-segment the candidate to match the reference as described below. In both cases, only completed (C) segments are re-segmented, but partial (P) segments are used to estimate timings of individual words.

Time-Based Segmentation: Using the *starting time* and *ending time* of each segment in the reference transcript, the corresponding words in the SLT output are selected (i.e., words with their time between the *starting* and *ending time*). To compensate for minor timing errors, we expand this span by one word in each direction in the SLT output. All the words from the starting to the ending one are taken as the candidate segment, see Figure 3.

Word-Based Segmentation: We use the 1-1 correspondence between segments in the golden source transcript and reference translation. We apply *mwerSegmenter* to re-segment candidate translation (the concatenation of "C" segments) to match exactly the segmentation of the reference translation and then work with source–candidate segment pairs. Again, minor *mwerSegmenter* errors are compensated by expanding candidate segments by one word at each end, see Figure 4.

4.2.1 Proportional Delay Calculation

We need to attribute an "expected" time to each word in the reference and then compare it with the time the word was displayed in SLT output.

For proportional delay calculation, we first estimate the timing of each source word based on partial (P) segments[6] in the golden transcript and then attribute these times to words in the reference translation, proportionally along the sequence of words.[7] This is an oversimplification because word alignment is not monotonic and also because the reference translation was created in written form with access to the full source, so even the first word of the reference may well be influenced by some late source words.

Formally, we are populating table $T(i, j)$ with expected times of jth word in the ith segment of the reference translation. First we estimate the times of *source* words in the ith complete segment of the golden transcript based on *starting* and *ending time* of the (partial) source segment where the source word first appeared. For example, when three new words are added in a partial segment ending at t_2 compared to the previous partial segment which ended at t_1, we need to divide the time interval (t_1, t_2) among these three words. We estimate that the first word appeared at $t_1 + (t_2 - t_1)/3$, the second one at $t_1 + 2 * (t_2 - t_1)/3$ and the last one at t_2.

This source word timing is transferred to the target word timing proportionally. With l_i being the length in words of source segment i and m_i the corresponding reference length, we denote $P = j * l_i/m_i$ as a shortcut for the fractional index of the *source* word which corresponds to the j word in the reference segment i. We then define:

$$T(i,j) = t_{\lfloor P \rfloor} + \left((t_{\lceil P \rceil} - t_{\lfloor P \rfloor}) * (P - \lfloor P \rfloor) \right) \quad (1)$$

t_x is the expected time of the xth word of the *source* sentence i and $\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$ round to the nearest integer.

To see how the proportional delay calculation works in practice, consider the example in Figure 2. We first need to estimate the times for each source word. Since the first source partial segment consists of 3 words, we estimate the times of "We","would" and "like" by dividing the 760–827 interval into three equal parts, whereas the other source words are assigned to the end timestamps of their time intervals, since they appear in individual increments. The estimated source times are therefore:

We	would	like	to	introduce	our	company
782	805	827	846	919	961	1062

In order to perform the proportional delay calculation, we note that the source-reference length ratio is 7/6, so that the value P in Equation 1 is equal to $7j/6$ for the j^{th} reference word. Substituting into Equation 1 gives the following expected times for the reference words:

Wir	würden	gern	unser	Unternehmen	vorstellen
786	812	836	895	954	1062

Comparing the expected times with the actual times in Figure 2b, we can see that the total delay

[6] Partial segments provide more accurate word-level timing but we can and do resort to equidistant division of the complete segment time span if golden transcript lacks partial segments.

[7] Word lengths could be used as an additional refinement.

P	760	827	We would like
P	760	847	We would like to
P	760	919	We would like to introduce
P	760	961	We would like to introduce our
C	760	1062	We would like to introduce our company.

(a) Time-stamped golden source transcript

P	800	720	760	Wir
P	870	720	860	Wir möchten
P	910	720	905	Wir möchten vorstellen
C	1200	720	1110	Wir möchten unser Unternehmen vorstellen.

(b) SLT output

Wir würden gern unser Unternehmen vorstellen

(c) Reference

Figure 2: Example for proportional delay calculation.

is given by:

$$(800-786)+(1200-895)+(1200-954)+0 = 565$$

In this sum, "würden" and "gern" are not included at all because they do not appear in the hypothesis. "unser" and "Unternehmen" both appeared at the same time 1200 and "vorstellen" has zero delay, since it arrives at 910, *before* its expected time.

4.2.2 Delay Calculation using Alignments

The SLT system should not be expected to produce any word earlier than the reference produced it, e.g. due to grammatical constraints of the target language. (If it does, we do not penalize it. Giving a bonus for such an earlier appearance is yet to be considered.)

We use the word alignment between source words (which are time-stamped in the golden transcript) and reference words to attribute timing information to reference words, see "Table T" in the middle of Figures 3 and 4.

We set the expected time of each reference word as the maximum of the timestamp of the last source word aligned to this reference word (the reference translator "had to wait" for the respective source piece of information) and the expected time of the preceding reference word (the translator "had to postpone" any words he or she already knew until the missing one became available to respect target language grammatical order). With this definition, any SLT system is allowed to "wait" for the source or "postpone" its output without penalization same as the reference translator did. E.g. the word "vorstellen" (introduce) is expected at 1062 in the proportional delay calculation (upper Tables T). Based on alignment only, it would be expected at 919 (struck out in the figures), because that is the time when the aligned "introduce" appeared but we max it out to 1062 because the preceding "Unternehmen" (company) was available only at 1062.

For "unser", SLTEV selects the expected time as the maximum between 895 (its expectation time under proportional delay) and 961 (the time that its aligned source word "our" appeared) . In other words, SLTEV gives more time to the SLT system to display the "unser" because its aligned word is output a bit later than the proportional expectation of "unser". Under the alignment-based delay, we do not expect that the word will be output earlier than its alignment indicates.

Technically, we rely on automatic word alignments by MGIZA (Gao and Vogel, 2008) which is a multi-threaded version of GIZA++ (Och and Ney, 2003), aligning the completed segments of the golden source transcript and the reference translation. The effect of alignment errors on the reliability of the evaluation is yet to be explored.

4.2.3 Multi-Reference Delay Calculation

With multiple references, we create a separate table T for each and calculate the delay of each segment individually, taking the minimum across all references. The final delay is the sum of these minima. We use this strategy for both delay calculation methods and both segmentation strategies introduced above.

4.3 Flicker to Assess Stability

For systems that revise their outputs, (in)stability of the output is important because it could distract the user. Following Niehues et al. (2018), "flicker" commonly reflects the number of words after the first difference between two consecutive output updates. We report two variants of Flicker:

Average revision count per segment:

The revision count RC for each completed ("C") segment k is calculated as: $RC_k = \sum_{i=2}^{n_k} (|s_{i-1}| - |\text{LCP}(s_{i-1}, s_i)|)$, where s_i is the ith partial segment preceding the current complete segment k and n_k is the number of partial segments between complete segments $k-1$ and k. LCP gets the longest common prefix. If segment k has no

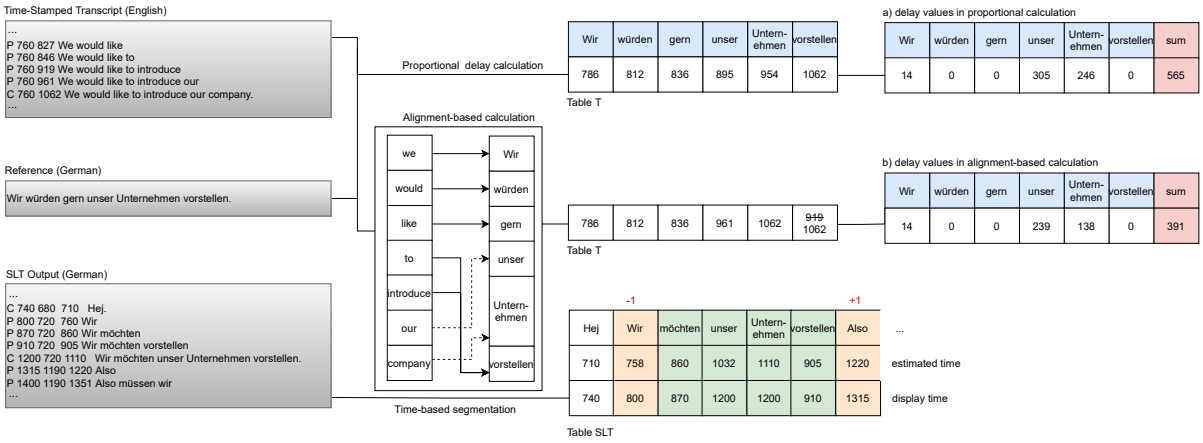

Figure 3: Time-based segmentation for proportional (a) and alignment-based (b) delay calculation. Using time-stamped transcript and reference translation, Table T is pre-computed. Then using timings in SLT output, word-level timestamps are estimated ("Table SLT"). The a) and b) value of delay is the sum of differences between expected word times in Table T and display times in Table SLT.

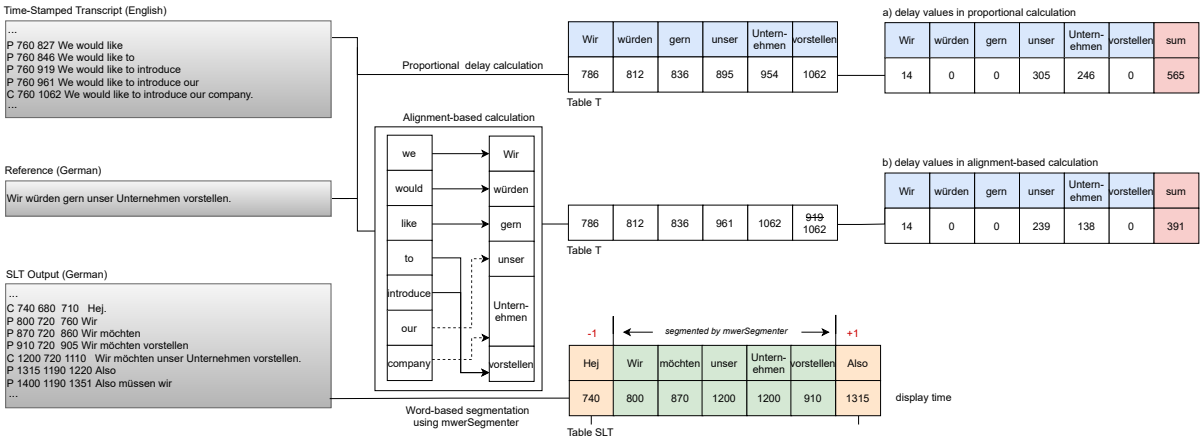

Figure 4: Word-based segmentation (*mwerSegmenter*) for proportional (a) and alignment-based (b) delay calculation. The main difference from Figure 3 is in finding the span of words that form the candidate segment, i.e. contribute to "Table SLT". Table SLT now needs to contain only display time of words.

preceding partial segments, RC_k is zero. Average revision count is calculated as: $\frac{1}{K}\sum_{k=1}^{K} RC_k$, where K is the total number of complete segments.

A disadvantage of this strategy is that if the system makes a little change (1-2 chars) at the start of the sentence, it gets heavily penalised.

Normalized revision count:

Similar to Arivazhagan et al. (2019a), normalized revision is the total revision count ($\sum_{k=1}^{K} RC_k$) divided by the output length (sum of lengths of completed segments).

4.4 sacreBLEU to Assess Quality

Early versions of SLTEV used NLTK (Bird et al., 2009) implementation of BLEU but it behaved badly on empty segments and used a less common tokenization scheme. We fully switched to sacreBLEU, calculating three variants of the score: (1) disregarding segmentation, we concatenate all completed segments and evaluate them against the concatenated reference as if it was a single segment, (2) force the candidate to reference segmentation using *mwerSegmenter* and calculate standard BLEU, (3) time-span quality. The third option divides the whole document into chunks of a fixed duration (e.g. 30 seconds) and treats all words in that span as a single segment. These single-segment BLEUs are reported, providing an estimate of translation quality over time, and also averaged for a summary.

If multiple references are available, we pass them to sacreBLEU which follows standard multi-reference BLEU and chrF calculations. In word-based segmentation, we use the first reference as the basis for *mwerSegmenter* re-segmentation.

5 A Growing Test Set

To allow for continued and comparable evaluation of SLT by the research community, we create and keep extending a publicly available dataset which contains source audio, time-stamped golden transcripts and reference translations for different types of inputs called `elitr-testset`.[8] The dataset currently focuses on European languages, as needed by the ELITR project (Bojar et al., 2020, 2021), but it is designed to be easily extensible in both languages and domains. With the help of commit IDs, full reproducibility of evaluations is ensured, even as the dataset will be growing.

In simple words, the `elitr-testset` is an assorted collection of documents, with inputs and expected outputs for ASR, MT and/or SLT systems. We expect our users to pick a relevant subset of these documents depending on their application needs and evaluate on this subset. For comparability, we standardize some of these selections by introducing the concept of "indices".

Each index is simply a file list of documents and it is also versioned in the `elitr-testset`. For example, we provide indices of documents which are good for purposes like: (1) SLT of English into Czech/German in the auditing domain, (2) English ASR in the computational linguistics domain, and (3) Czech/German ASR, regardless of the domain.

Another feature of `elitr-testset` is a collection of automatic checks that verify formal integrity of the documents (e.g. character encoding, line ends, number of lines) before every commit.

All datasets included in `elitr-testset` are free for public use but some indices include confidential files.

SLTEV can be used as a stand-alone tool to evaluate ASR, MT or SLT using source, candidate and reference files you provide, or it can be used very conveniently with `elitr-testset`. Running `SLTev -g index-name` will provide you with input files that your system should process and a second run of SLTEV will report your system's scores for the given index.

5.1 Practical Check

A preliminary version of SLTEV evaluated the submitted systems participating in the "Non-Native Speech Translation" shared task of IWSLT 2020 (Ansari et al., 2020). We ran simplified configurations of SLTEV (i.e., without calculating Delay and Flicker) for systems that did not provide enough information in their output.

Five teams from three institutions took part in the IWSLT 2020 SHARED TASK which was designed for English-Czech and English-German language pairs. The main test sets used in the shared task (and now included in `elitr-testset`) were:

Antrecorp: 37 files each of which is an up to 90-second mock business presentation given by high school students in very noisy conditions. None of the speakers is a native speaker of English and their English contains many lexical, grammatical and pronunciation errors as well as disfluencies due to the spontaneous nature of the speech.

KhanAcademy: six files each of which is an educational video. The speaker is not a native speaker of English but his accent is generally rather good.

SAO: six files illustrating interpretation needs of the Supreme Audit Office of the Czech Republic. The speakers' nationality affects their accent. The Dutch file is a recording of a remote conference call with further distorted sound quality.

6 Conclusion

In this paper, we introduced SLTEV, a framework for comprehensive and fine-grained evaluation of the output of simultaneous SLT systems, i.e., systems for live speech translation, and their components (ASR, MT). In contrast to text translation systems, simultaneous SLT systems cannot be judged just based on translation quality. For example, if the system waits for the whole sentence to be analyzed and processed, the translation quality will likely be better but the high latency may not be acceptable to the end user. SLTEV evaluates quality, latency, and stability (the number of corrections the system makes). In order to tackle the problem of the output segmentation, we proposed a new time-based segmentation method, in addition to the classical re-segmentation strategy of *mwerSegmenter*.

We complement the release of SLTEV with `elitr-testset`, a publicly available dataset of source speech and reference translations, so that truly comparable evaluations are available for the research community. SLTEV directly accesses this growing collection for easy and comparable scoring of your systems in various domains. We used a preliminary version of SLTEV to evaluate systems in one of the IWSLT 2020 shared tasks.

[8] `https://github.com/ELITR/elitr-testset/`

Acknowledgments


This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 825460 (ELITR) and the grant 19-26934X (NEUREM3) of the Czech Science Foundation.

The authors are grateful to Rishu Kumar, Dominik Macháček, Sangeet Sagar, Matúš Žilinec, and other members of the ELITR project for their valuable technical support on SLTEV and their help in improving it.

References

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.

Naveen Arivazhagan, Colin Cherry, Te I, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2019a. Re-translation strategies for long form, simultaneous, spoken language translation.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019b. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 437–445, Montréal, Canada. Association for Computational Linguistics.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (wmt20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

Ondřej Bojar, Dominik Macháček, Sangeet Sagar, Otakar Smrž, Jonáš Kratochvíl, Ebrahim Ansari, Dario Franceschini, Chiara Canton, Ivan Simonini, Thai-Son Nguyen, Felix Schneider, Sebastian Stücker, Alex Waibel, Barry Haddow, Rico Sennrich, and Philip Williams. 2020. ELITR: European live translator. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 463–464, Lisboa, Portugal. European Association for Machine Translation.

Ondřej Bojar, Dominik Macháček, Sangeet Sagar, Otakar Smrž, Jonáš Kratochvíl, Peter Polák, Ebrahim Ansari, Mohammad Mahmoudi, Rishu Kumar, Dario Franceschini, Chiara Canton, Ivan Simonini, Thai-Son Nguyen, Felix Schneider, Sebastian Stüker, Alex Waibel, Barry Haddow, Rico Sennrich, and Philip Williams. 2021. ELITR Multilingual Live Subtitling: Demo and Strategy. In *Proceedings of the System Demonstrations of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, Kyiv, Ukraine. Association for Computational Linguistics.

Colin Cherry and George Foster. 2019. Thinking slow about latency evaluation for simultaneous machine translation.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2008. Simultaneous translation of lectures and speeches. *Springer Netherlands, Machine Translation, MTSN 2008, Springer, Netherland*, 21(4).

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. *Association for Computational Linguistic*, 8(1):49—-57.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352, Doha, Qatar. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language

pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952, Cambridge, MA. Association for Computational Linguistics.

Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu. 2019. Direct speech-to-speech translation with a sequence-to-sequence model.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.

Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. 2005. Evaluating machine translation output with automatic sentence segmentation. In *International Workshop on Spoken Language Translation*, pages 148–154, Pittsburgh, PA, USA.

Markus Müller, Thai Son Nguyen, Jan Niehues, Eunah Cho, Bastian Krüger, Thanh-Le Ha, Kevin Kilgour, Matthias Sperber, Mohammed Mediani, Sebastian Stüker, and Alex Waibel. 2016. Lecture translator - speech translation framework for simultaneous lecture translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 82–86, San Diego, California. Association for Computational Linguistics.

J. Niehues, T. S. Nguyen, E. Cho, T.-L. Ha, K. Kilgour, M. Müller, M. Sperber, S. Stüker, and A. Waibel. 2016. Dynamic transcription for low-latency speech translation. In *17th Annual Conference of the International Speech Communication Association, INTERSPEECH 2016; Hyatt Regency San FranciscoSan Francisco; United States; 8 September 2016 through 16 September 2016*, volume 08-12-September-2016 of *Proceedings of the Annual Conference of the International Speech Communication Association. Ed. : N. Morgan*, pages 2513–2517. International Speech and Communication Association, Baixas.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Interspeech 2018*, Hyderabad, India.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Ofir Press and Noah A. Smith. 2018. You may not need attention.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Collection of a simultaneous translation corpus for comparative analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 670–673, Reykjavik, Iceland. European Language Resources Association (ELRA).

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Hao Xiong, Ruiqing Zhang, Chuanqiang Zhang, Zhongjun Hea, Hua Wu, and Haifeng Wang. 2019. Dutongchuan: Context-aware translation model for simultaneous interpreting.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.