This document is part of the Research and Innovation Action "European Live Translator (ELITR)". This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 825460.


# Deliverable D1.6

# Year 3 Test Sets

Rishu Kumar (CUNI), Vojtěch Srdečný (CUNI), Ebrahim Ansari (CUNI), Mohammad Mahmoudi (CUNI), Ondřej Bojar (CUNI), Peter Polák (CUNI), Felix Schneider (KIT)

Dissemination Level: Public

Final (Version 1.0), 30th June, 2021

| | |
|---|---|
| Grant agreement no. | 825460 |
| Project acronym | ELITR |
| Project full title | European Live Translator |
| Type of action | Research and Innovation Action |
| Coordinator | Doc. RNDr. Ondřej Bojar, PhD. (CUNI) |
| Start date, duration | 1st January, 2019, 36 months |
| Dissemination level | Public |
| Contractual date of delivery | 30th June, 2021 |
| Actual date of delivery | 30th June, 2021 |
| Deliverable number | D1.6 |
| Deliverable title | Year 3 Test Sets |
| Type | Demonstrator |
| Status and version | Final (Version 1.0) |
| Number of pages | 29 |
| Contributing partners | CUNI |
| WP leader | UEDIN |
| Author(s) | Rishu Kumar (CUNI), Vojtěch Srdečný (CUNI), Ebrahim Ansari (CUNI), Mohammad Mahmoudi (CUNI), Ondřej Bojar (CUNI), Peter Polák (CUNI), Felix Schneider (KIT) |
| EC project officer | Luis Eduardo Martinez Lafuente |
| The partners in ELITR are: | ▪ Univerzita Karlova (CUNI), Czech Republic<br>▪ University of Edinburgh (UEDIN), United Kingdom<br>▪ Karlsruher Institut für Technologie (KIT), Germany<br>▪ PerVoice SPA (PV), Italy<br>▪ alfatraining Bildungszentrum GmbH (AV), Germany |
| Partially-participating party | ▪ Nejvyšší kontrolní úřad (SAO), Czech Republic |

For copies of reports, updates on project activities and other ELITR-related information, contact:

Doc. RNDr. Ondřej Bojar, PhD., ÚFAL MFF UK
Malostranské náměstí 25
118 00 Praha, Czech Republic
bojar@ufal.mff.cuni.cz
Phone: +420 951 554 276
Fax: +420 257 223 293

Copies of reports and other material can also be accessed via the project's homepage:

`http://www.elitr.eu/`

# Contents

# 1 Executive Summary

This deliverable reports on the status of test data that we collect and curate for the speech recognition and translation technologies, as well as for the automatic meeting summarization prototypes we are developing within WP1 Data of the ELITR project.

The data collected and described here serve directly in the research work packages WP2 ASR, WP3 SLT, WP4 MT and WP5 Minuting, but they are also used in tests of the integrated solutions (WP7 Integration) and offered for public use by fellow researchers in the field.

In Section 2, we describe new additions to `elitr-testset`. Section 3 contains a brief description of `SLTev`, a tool we developed to automatically evaluate ASR, SLT and MT systems against `elitr-testset` or custom files. `SLTev` presentation at EACL (Ansari et al., 2021) received the "outstanding demo" badge.

In Section 4, we describe how our system evaluation is done, either by evaluating individual components or evaluating complete processing pipelines, i. e. multiple components at once. Finally, Section 5 describes the dataset created for Automatic Minuting and the Automatic Minuting shared task presented on Interspeech 2021. Section 6 concludes the report.

Overall, WP1 Data is proceeding according to the plan.

# 2 ELITR Test Set

In this section, we report on the new additions to `elitr-testset` over the last year. The design and composition of `elitr-testset` was described in the Deliverable D1.5 (Final Training Data, Separating Confidential and Public Version).

Since Deliverable D1.5, we have utilized our systems for transcription and translation at multiple venues and based on the preparation for the venue, we describe the addition of documents in `elitr-testset` as follows:

- MALACH:[1] This dataset consists of manually revised YouTube transcripts of publicly available videos of previous MALACH Keynote presentations and talks, namely by Jakub Mlynář, Paul Rukesha and Martin Šmok. All are non-native speakers speaking English; the technical details of the recording are not known. For links of the individual keynote or talk, please refer to the README.md file in the subset of the public repository of `elitr-testset`.[2]
- THEAITRE: This dataset consists of the Czech script of the play named **AI: WHEN A ROBOT WRITES A PLAY** and the manual transcription of the Czech discussion that followed the play, as well as the interpretation into English.
- Antrecorp-2021: Following our good experience with obtaining data from high-school students, non-native speakers presenting their mock companies in English (Macháček et al., 2019), we repeated the collection in 2021 again. The new set of files consists of 11 manually corrected transcripts and sound files of the virtual Fair of Student Firms 2021 grouped by the name of participating student firms. The recordings were obtained on the hardware of the participating students and the quality and conditions thus vary.
- ESIC Dev Set: a validation subset of ESIC – Europarl Simultaneous Interpreting Corpus. In `elitr-testset`, we include 5 hours of English speeches from European Parliament plenary sessions, with simultaneous interpreting into Czech and German, as released with the paper Macháček et al. (2021). There are source audios, manual transcripts of both the source and interpeters' speech with word-level timestamps, and independently created revised translations in this `elitr-testset` copy. The ESIC release, as described in Macháček et al. (2021), contains 5 additional hours of test set. Given the ease of access to `elitr-testset`, we preferred to include only ESIC dev set, so that ESIC test set remains less used.

In Deliverable D1.5, we introduced the concept of indices to explicitly capture a curated subset of related documents from `elitr-testset`. Whenever a new document is added to `elitr-testset`, it is also included in relevant indices so that it is possible to automatically use the documents for evaluation purposes.

Three recent additions (MALACH, THEAITRE, ESIC) also introduce new domains, so we create new index files for them. We also create new index files for Antrecorp-2021 files as previous dataset for Antrecorp also include Reference files for MT, whereas the new addition is solely to be used for ASR evaluation.

## 2.1 Summary

We define `elitr-testset` as an ever-growing collection of documents.

In this section, we provide statistics of the current documents included in `elitr-testset` for ASR, MT and SLT evaluation. These tables along with the tables reported in Section 6 of D1.5, give an overview of the growth of `elitr-testset`.

[1] `https://ufal.mff.cuni.cz/malach/en`
[2] `https://github.com/ELITR/elitr-testset/tree/master/documents/malach`

| Index | Words | Lang | Duration |
|---|---|---|---|
| auto-asr-czech-auditing | 5579 | cs | 1:17:06 |
| auto-czech-asr | 72408 | cs | 10:04:13 |
| auto-2021-antrecorp | 2204 | en | 0:14:32 |
| auto-asr-english-auditing | 19949 | en | 3:06:39 |
| auto-asr-esic | 46892 | en | 5:07:17 |
| auto-iwslt2020-antrecorp | 6634 | en | 1:28:09 |
| auto-iwslt2020-consecutive | 3207 | en | 1:05:54 |
| auto-iwslt2020-devset | 196325 | en | 1:46:41 |
| auto-iwslt2020-khanacademy | 4470 | en | 0:45:46 |
| auto-iwslt2020-wgvat | 21792 | en | 4:21:23 |
| auto-langtools-workshop | 5145 | en | 0:17:49 |

Table 1: This table is extracted from auto-generated summaries of individual index files for `elitr-testset` commit-id **554090** on June 24, 2021. The prefix "auto-" in index names means that these indices were created automatically by considering all documents from a particular source.

Table 1 briefly summarizes the part of `elitr-testset` which can be used for evaluating ASR quality for English and Czech language. It covers a wide array of domains and includes speech from non-native speakers for English. Table 2 lists the part of the dataset which is used for MT system evaluation. It includes new files from a previously left-out files for language pairs such as English↔Irish. We also add some low-resource pairs such as English↔Montenegrin. We disregard the effects of translationese and use these index files bi-directionally, i.e. target language and source language files can be swapped when evaluating the reverse direction of MT. Please note that source language and target language may have different line counts, because some documents have multiple reference translation files present. A part of the dataset used for evaluating MT is not for public release, because it contains confidential contents. Please refer to D1.5 for a precise description of how we distinguish between the documents for ASR, MT and SLT.

| Index | Lang 1 | Line Count | Word Count | Lang 2 | Line Count | Word Count |
|---|---|---|---|---|---|---|
| auto-mt-en2ar | en | 1167 | 7174 | ar | 1166 | 6577 |
| auto-mt-en2be | en | 5000 | 76077 | be | 5000 | 61058 |
| auto-mt-en2bg | en | 5000 | 78602 | bg | 5000 | 70565 |
| auto-mt-en2bs | en | 1346 | 34924 | bs | 1346 | 30563 |
| auto-mt-en2ca | en | 5842 | 86170 | ca | 5842 | 92542 |
| auto-mt-en2cnr | en | 350 | 10203 | cnr | 350 | 8614 |
| auto-mt-en2cs | en | 17651 | 352058 | cs | 18285 | 292271 |
| auto-mt-en2da | en | 6655 | 127364 | da | 6655 | 127056 |
| auto-mt-en2de | en | 17609 | 355233 | de | 15406 | 268355 |
| auto-mt-en2es | en | 9884 | 178732 | es | 9884 | 190343 |
| auto-mt-en2fi | en | 5000 | 74933 | fi | 5000 | 56436 |
| auto-mt-en2fr | en | 5000 | 89541 | fr | 5000 | 96549 |
| auto-mt-en2hr | en | 6044 | 111019 | hr | 6044 | 96795 |
| auto-mt-en2hu | en | 9000 | 155546 | hu | 9000 | 123768 |
| auto-mt-en2ir | en | 1045 | 21425 | ir | 1045 | 22582 |
| auto-mt-en2it | en | 5000 | 80004 | it | 5000 | 78645 |
| auto-mt-en2lt | en | 5000 | 83961 | lt | 5000 | 58132 |
| auto-mt-en2lv | en | 5000 | 80868 | lv | 5000 | 62740 |
| auto-mt-en2mk | en | 5000 | 74533 | mk | 5000 | 68557 |
| auto-mt-en2nl | en | 10014 | 201665 | nl | 10014 | 205139 |
| auto-mt-en2no | en | 5000 | 83400 | no | 5000 | 82039 |
| auto-mt-en2pl | en | 9042 | 151937 | pl | 9042 | 130613 |
| auto-mt-en2pt | en | 5000 | 77563 | pt | 5000 | 74094 |
| auto-mt-en2ro | en | 9000 | 174948 | ro | 9000 | 181361 |
| auto-mt-en2ru | en | 9173 | 199606 | ru | 9173 | 180596 |
| auto-mt-en2sk | en | 5042 | 76598 | sk | 5042 | 62200 |
| auto-mt-en2sl | en | 5000 | 76693 | sl | 5000 | 69096 |
| auto-mt-en2sr | en | 6187 | 117775 | sr | 6187 | 106128 |
| auto-mt-en2sv | en | 5000 | 77744 | sv | 5000 | 76025 |
| auto-mt-en2uk | en | 5000 | 81313 | uk | 5000 | 66873 |

Table 2: Automated summary of the **auto-mt-*** index files in `elitr-testset` at commit-id 912586.

# 3 SLTev

In this section, `SLTev` (Ansari et al., 2021), our tool for comprehensive evaluation of spoken language translation, and its usage with `elitr-testset` are summarized.

`SLTev` was presented to the research community at EACL 2021, receiving the "outstanding demo paper' recognition. The full paper is available in Appendix A.

## 3.1 `SLTev`: The tool

`SLTev` is an open-source tool for evaluating spoken language translation (SLT) in a comprehensive way. `SLTev` can also evaluate the intermediate steps alone: the output of automatic speech recognition (ASR) and machine translation (MT).

Three input files are used for SLT evaluation: a time-stamped golden transcript in the source language (OStt; original speech transcribed with timestamps), a reference translation and candidate output in the target language (hypothesis).

`SLTev` reports the quality, latency, and stability of an SLT candidate output and word error rate (WER) of an ASR output. `SLTev` can be installed from source[3] or PyPI[4]. Moreover, `SLTev` works with `elitr-testset` and can automatically use the growing collection of input files of `elitr-testset`.

[3] https://github.com/ELITR/SLTev
[4] pip3 install `SLTev`

## 3.2 The communication between SLTev and elitr-testset

Each index in elitr-testset has been created for a specific domain and purpose, containing the list of all relevant files in the documents/ directory in elitr-testset. SLTev can generate a simple (flat) directory with all files belonging to an index, so that the user does not have to navigate the directory structure, using SLTev with the -g[5] parameter. When SLTev is called this way for the first time, it clones elitr-testset repository to the local system and copies the desired files to the output directory, see "Generate input files" in Figure 1. In subsequent calls, the local clone of the repository is used. After input files are generated, SLTev can evaluate user hypothesis files with generated input files by its modules.

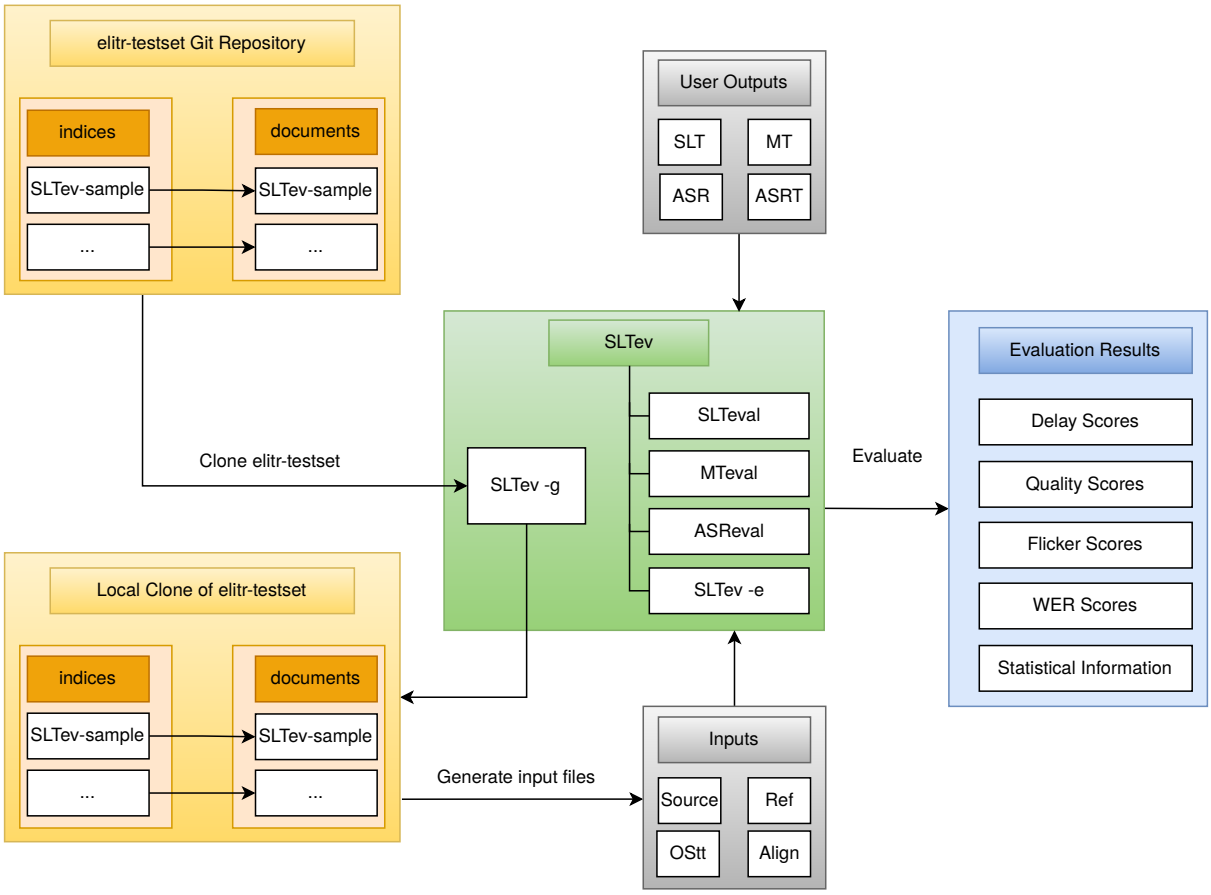


Figure 1: By running SLTev with -g parameter, elitr-testset repository is cloned and files are generated into the output directory (yellow boxes). The gray boxes show user outputs and inputs types and blue box shows different types of scores.

## 3.3 Score calculation

SLTev has four types of modules for evaluating user outputs based on the input files. The first one is SLTeval that evaluate SLT files,[6] the second one is MTeval which designed for evaluating MT files, the third module is ASReval that evaluate ASR and ASRT[7] files and the last one is SLTev itself with called as "SLTev -e <HYPOTHESIS-FILES>" which evaluates the four types of user outputs in a folder together.

MT and SLT files are in the target language and ASR and ASRT files are in the source language. Here we summarize the input and user output types of each module, as well as the scores returned in that case:

[5] SLTev -g <INDEX-NAME> –outdir <OUTPUT-DIR>
[6] Evaluating SLT files is the main goal of SLTev.
[7] Timestamped ASR hypothesis, with partial outputs.

- SLTeval
  - User output type(s): SLT
  - Input type(s): OStt, Reference, Align[8] (optional)
  - Assumptions: SLT and Reference may have a different number of segments (i.e. lines), `SLTev` resegments the SLT output
  - Score types: Delay scores, Quality scores, Flicker scores
- MTeval
  - User output type(s): MT
  - Input type(s): Reference
  - Assumptions: MT and Reference must have the same number of segments, `SLTev` honors the segmentation
  - Score types: Quality scores
- ASReval
  - User output type(s): ASR, ASRT
  - Input type(s): OSt, OStt
  - Assumptions: ASR (or ASRT) and OSt may have a different number of segments, `SLTev` resegments the ASR/ASRT output
  - Score types: WER scores, Quality scores
- `SLTev -e`
  - User output type(s): SLT, MT, ASR, ASRT
  - Input type(s): OSt, OStt, Reference, Align
  - Score types: Delay scores, Quality scores, Flicker scores, WER scores

# 4 Evaluation of ELITR Systems

We use `SLTev` (Section 3) to evaluate our systems both at the level of individual components (Section 4.2) as well as at the level of complete pipelines (Section 4.3). The evaluations are performed in `elitr-testset-evaluations` (Section 4.1), a repository containing results of individual evaluations.

## 4.1 `elitr-testset-evaluations`

`elitr-testset-evaluations` is the repository used for storing data used for individual evaluations and the obtained results. It is intended to be used for multiple kinds of evaluations, such as individual components and pipelines. This allows for easy chaining of evaluations: for example, when the outputs of an ASR system are to be translated with a MT system, the outputs are already stored in the repository, so they can be directly reused. On numerous occasions, the stored outputs were handy in various discussions etc.: we just exchanged links to sample output files and our colleagues could immediately analyze or build upon the outputs.

As `elitr-testset` grew, participants updated their ASR and MT systems with more training data, or optimized the existing systems. This resulted in systems performing differently over time. By storing the inputs and outputs of the evaluations in `elitr-testset-evaluations`, we can evaluate specific versions of any system in connection with other systems to see how the recent updates to the system fairs in comparison to the previous version of the same system.

[8] Word alignment between reference and source words

For the purpose of an evaluation, the concept of *campaign* is introduced in this subsection. A campaign is a name for the evaluation of `elitr-testset` or part thereof, using specific components of the ELITR systems. It is advised to include the date in campaign name to have a clear indication of the version of the systems used. This strategy will provide a clean way of comparing performance of individual components during different time in individual component development cycle. `elitr-testset-evaluations` is currently kept as a private repository on GitHub and can be made accessible for review upon request. The evaluations are saved in the evaluations directory.

## 4.2 Evaluating Individual Components

We illustrate the structure of the `elitr-testset-evaluations` directory with an example below, where we evaluate an individual component.

Let's consider the Czech audio file *europarliament.cs.OS.mp3*, for which an automatic transcript was generated in the campaign called "20210530-Czech-ASR-comparison". The transcript for the audio file will be present at *evaluations/20210530-Czech-ASR-comparison/transcript/cs-CZ-kaldi_general*
*/czech-asr/europarliament/europarliament.cs.cs.asr*

We briefly summarize the hierarchical directory structure as follows:

- evaluations: The main folder where all campaign results are stored.
- 20210530-Czech-ASR-comparison: Name of the campaign.
- transcript: It denotes that the file inside this directory will be transcripts of the audio. Other options are "raw_logs", for ASR result with timestamps and "MT" for machine translation of the ASR output.
- cs-CZ-kaldi_general: Fingerprint of the ASR worker used for the evaluation
- czech-asr/europarliament/: This is the directory structure of *europarliament.cs.OS.mp3* inside the documents folder of `elitr-testset`. For simplicity, this structure has been replicated in the `elitr-testset-evaluations` as well.
- europarliament.cs.cs.asr: Name of the transcript file.

The evaluation process in `elitr-testset-evaluations` is fairly simple and automated for user's ease. Depending on your requirement for getting ASR or MT, we provide simple shell-script files which takes index file, fingerprint of worker and campaign name as input and can produce raw logs and transcripts for ASR evaluation and translated files for MT evaluation. The ASR output can then be further processed with another shell script which takes ASR outputs (either raw logs or transcript) as input and can generate the corresponding SLT files for them. These scripts are located in the "scripts" directory of the `elitr-testset-evaluations` GitHub repository and the outputs from these scripts will be saved in evaluations directory with the hierarchy described above. While running a campaign, the ASR and MT evaluation scripts also create a file which stores the current commit id of `elitr-testset` and `SLTev` as text files. This is included to help reproduce any result or error, as `elitr-testset` is ever changing and `SLTev` is still in active development.

To illustrate such an evaluation campaign in practice, we compare two ASR systems for Czech within ELITR QuartzNet-based (Kriman et al., 2020) fully-neural off-line one with the Czech ASR demo by the University of Western Bohemia available as a LINDAT service.[9] For results marked with $\star$ in Table 3, the audio files were split in multiple segments with individual duration between 20-30 minutes, as we noticed that LINDAT API was closing the connection around the 30 minutes mark. Each audio segment was split on occurance of a gap in speech

[9] `https://lindat.mff.cuni.cz/services/uwebasr/`

| File_name | LINDAT | ELITR | CUNI-E2E |
|---|---|---|---|
| belgian | 0.152 | 0.302 | 0.199 |
| comp_linguistics | 0.196⋆ | 0.252 | 0.186 |
| europarliament | 0.327⋆ | 0.332 | 0.349 |
| polish | 0.181 | 0.279 | 0.186 |
| romanian | 0.247 | 0.228 | 0.283 |
| rozhlas | 0.184⋆ | 0.154 | 0.415 |
| snemovna | 0.253⋆ | 0.264 | 0.127 |
| spanish | 0.153 | 0.217 | 0.187 |
| wgvat | 0.108 | 0.220 | 0.208 |
| Average ± Std. Dev | 0.200 ± 0.066 | 0.250 ± 0.053 | 0.238 ± 0.092 |

Table 3: A WER score comparison between ASR Evaluation results for Czech ASR from LINDAT, ELITR system and an E2E ASR system developed at CUNI (**lower is better**). Each line corresponds to a single file in `elitr-testset`. Scores were obtained with `SLTev`, using the LPC variant, i.e. Lowercased words, Punctuation removed and Concatenating all sentences into one. For more details on other methods, please refer to the README file in SLTev GitHub repository as described in Section 3.

for 3-4 seconds. These files were then individually transcribed using LINDAT and the output of each segment was concatenated to get a final output. As documented in Table 3, we see that in some cases, LINDAT performs better or at-par with our Kaldi-based ASR system (denoted "ELITR" in the table). Similarly, the QuartzNet-based solution (labelled "CUNI-E2E") performs reasonably well across all the files as it has an advantage of being an end-to-End neural network for ASR generation.

Further insight was obtained in manual inspection of selected files from the `elitr-testset`:

- `belgian, europarliament, polish, romanian, spanish` The recordings are actually recordings of interpretations (simultaneous in the `europarliament` case, consecutive otherwise). There are multiple speakers (interpreters) with various level of fluency. The recordings are characterized by disfluent speech with background noise (original speech).
- `comp_linguistics` The recording is from a Czech lecture on Computational Linguistics (CL). It contains a large portion of English terms from CL. The speech is mainly fluent (it is a lecture) with minor background noise.
- `rozhlas` The recording is from the Czech Radio. It contains different domains, multiple speaker and even sound tracks (e.g., the initial music of a news program). ELITR system performs the best as the Czech Radio recordings (different from those in the test set) are part of the training data.
- `snemovna` The recording is from the Czech Parliament. The speeches are mainly fluent with low background noise. CUNI-E2E performs best, as the training data of CUNI-E2E is the corpus of Czech Parliament Plenary Hearings (Kratochvíl et al., 2020); again distinct from this test set.
- `wgvat` The recordings come from an auditor workshop which was held in English. The tested sound files are thus again recordings of interpreters, but in this case the interpreters are still students (not yet professionals) of interpretation.

The use of language model re-scoring in LINDAT and ELITR can be recognized in the transcripts. CUNI-E2E occasionally makes mistakes easy to recover because it does not use LM re-scoring. On the other hand, it is capable of some limited code-switching and proper nouns recognition. It would be interesting to prepare in-domain LMs for particular files to see how much improvement they can bring.

| File_name | en_asr_WER | en2de_rb_sacreBLEU | en2cs_rb_sacreBLEU |
|---|---|---|---|
| 003_007_EN_Swinburne | 0.167 | 3.4 | 2.4 |
| 003_017_EN_Allister | 0.117 | 5.2 | 6.2 |
| 003_021_EN_Nicholson | 0.264 | 7.4 | 5.6 |
| 003_070_EN_McGuinness | 0.088 | 5.6 | 14.0 |
| 006_005_EN_Malmström | 0.163 | 11.5 | 9.7 |
| 006_006_EN_Hannan | 0.168 | 3.5 | 6.5 |
| 006_016_EN_Batten | 0.116 | 19.9 | 4.9 |
| 007_002_EN_Malmström | 0.137 | 7.6 | 5.3 |
| 007_004_EN_Hannan | 0.241 | 5.1 | 6.6 |
| 008_014_EN_Dodds | 0.103 | 4.3 | 8.7 |
| 008_054_EN_Karim | 0.159 | 6.3 | 10.9 |
| 009_010_EN_Barroso | 0.175 | 10.0 | 10.8 |
| 011_008_EN_Andreasen | 0.147 | 13.0 | 19.2 |
| 012_005_EN_Andreasen | 0.127 | 11.6 | 7.2 |
| 012_008_EN_Ojuland | 0.199 | 7.6 | 11.5 |
| 012_024_EN_Arlacchi | 0.381 | 8.4 | 6.5 |
| 012_038_EN_Savi | 0.263 | 7.0 | 6.3 |
| 013_014_EN_Bloom | 0.192 | 4.0 | 3.9 |
| 013_052_EN_Mitchell | 0.264 | 3.8 | 3.0 |
| 014_007_EN_Wallis | 0.204 | 4.7 | 3.3 |
| 015_012_EN_Kasoulides | 0.179 | 14.3 | 3.8 |
| 015_013_EN_Deva | 0.119 | 19.8 | 16.0 |
| 015_014_EN_Corazza-Bildt | 0.187 | 9.2 | 8.3 |
| 015_018_EN_Gallagher | 0.927 | 2.4 | 3.0 |
| 016_005_EN_Nicholson | 0.206 | 4.6 | 2.5 |
| 016_017_EN_Hannan | 0.159 | 6.5 | 6.0 |
| 016_032_EN_Vassiliou | 0.145 | 12.1 | 14.3 |
| 017_038_EN_McGuinness | 0.141 | 8.3 | 7.0 |
| Average ± Std. Dev | 0.205 ± 0.154 | 8.2 ± 4.6 | 7.7 ± 4.3 |

Table 4: ASR and MT scores obtained from individual pipeline components and evaluated against manually verified golden transcripts with MTeval (Section 3.3). For this evaluation, the source file for each row was obtained from the Rainbow MT system (EN→41 language) from UEDIN, and reference file was the transcription of the Interpreter's speech in German and Czech of the original English audio.

The recordings within `elitr-testset` are difficult for automatic speech systems because the domains of the speeches are very specific. A huge challenge for ASR systems is also a vast amount of code-switching and many proper nouns, especially foreign ones. Furthermore, 6 out of 9 files are simultaneous interpretations. They are inherently disfluent with many pauses where the interpreter is trying to comprehend the speaker. The recordings of simultaneous interpretations contain significant levels of background noise (original speech). Overall, `elitr-testset` covers most problematic parts of ASR systems: robustness towards background noise, difluent speech, non-native speech and various domains. This makes it a perfect resource for testing ASR performance.

## 4.3 Evaluating Complex Pipelines

We use `pipeliner`, a tool we developed to declaratively define pipelines as a directed acyclic graph, where nodes are individual components and edges are the data flow between the components. Individual setups are modelled using this tool because each setup for a particular event or session is very individual and uses different components. The resulting pipeline declaration is a Python file that gets converted to a single bash script which launches the pipeline. The data and results used for these evaluations can be found in the `elitr-testset-evaluations` repository, specifically the *evaluations/20210608-pipelines* and *evaluations/20210623-slt* folders.

When a pipeline is described this way, `pipeliner` also allows to evaluate parts of the pipeline,

simply by selecting two nodes that have a path between them and providing an *index file* containing the source and reference files for the given part of the pipeline.

When evaluating, `pipeliner` takes the pipeline description and generates a folder structure based on the *index file*. For each entry in the *index file*, a separate directory is created, containing the source and reference files and a bash script that launches the part of the pipeline to be evaluated. The source file is used as the input of the pipeline and the output of the target node is captured. Then, `SLTev` can be used to automatically evaluate the output against the reference file.

To illustrate this concept, we evaluated a part of our the pipeline we used for the EUROSAI Congress against the ESIC dev2 set (version 1.0; more details in Macháček et al., 2021). This dataset contains English and German speech of speakers at European Parliament, coming from the original speaker and from a human interpreter, respectively. These speeches also have manually corrected transcriptions. The original English speeches also have official English transcription and German translation produced by the European Parliament.

The first evaluated part of the pipeline consists of English ASR of the original speaker followed by English-to-German MT. This particular MT system is actually a multilingual one, capable of translating from English into 42 target languages; due to the high number of languages, we call it the "Rainbow MT". We evaluate some of the target languages further below in this section but for now we focus only on the German target. For a better insight, we start by reporting the WER scores of the ASR only, against the English reference transcript. We then report the final translation scores, comparing the MT output against the two possible reference translations: (a) the transcript of the German interpreter, (b) the text-based translation of the English transcript.

The second evaluated part of the pipeline consists of German ASR followed by German-to-English MT. At the EUROSAI Congress, this was an alternative source of English when the original English speaker had problematic accent or speech style. In such situations, we could resort to following the German interpreter instead, translating it (via English) into the desired 42 target languages. While this is a natural and useful setting, ESIC v.1.0 data is not fit for it: ESIC contains the German *interpreter* speech and we can only use the transcript of the *original English speaker* as the reference. Technically, this forms a 'round trip': the original speaker was presenting in English which was human-interpreted into German. We are following the German interpretation, translate it live back to English, and evaluate against the transcript of the original speaker. We can only guess if more information was lost by the German interpreter, or by the subsequent cascade. For a rough indication, we report also the German-only and English-only ASR scores.

The scores in Table 5 are WER scores of the ASR components, obtained with *ASReval*, specifically the LPC variant (lowercased, removed punctuation, concatenated sentences). The goal was to find out what is the performance when obtaining English and German text, starting from English or German speech, while using `pipeliner` for component-based evaluation. The results indicate both systems, German ASR and English ASR perform roughly the same (0.155 and 0.169 WER score, respectively), with German ASR performing slightly better.

The scores in Table 6 and Table 7 are sacreBLEU scores of the German to English and English to German pipelines respectively, obtained using *SLTeval*. They differ in the used reference files, which will be discussed below. It is important to note these numbers are flawed: in the case of English to German evaluation, we are using the transcription of the English to German interpreter's speech as the reference file, but we are using the English speaker's speech as the audio source. Obviously, the interpreter is not doing a word-for-word translation, so there is some semantics shift, which causes the score to worsen. The same thing is happening for the German to English evaluation, where we use the transcription of the English speaker's speech as the reference and the English to German interpreter's speech as the audio source.

Additionally, the ESIC set contains two reference sources for each speech: one source are the manual transcriptions of the speeches, of the original speaker and of the interpreters,and the other is the official transcription and translations of this transcriptions provided by the

Europarliament itself. Table 6 is using the manual transcriptions and Table 7 is using the transcription and translations provided by European Parliament.

Aside from ESIC set, we also evaluate an index file called *auto-slt-langtools*, which consists of a short English speech about spoken language translation and corresponding golden transcript as well as reference translations. The languages for reference translation files are: Arabic, Czech, Spanish, Polish and Slovak. The results for this index file evaluation are present in Table 8.

| **File name in ESIC** | **de2de_asr** | **en2en_asr** |
|---|---|---|
| 003_007_EN_Swinburne | 0.163 | 0.129 |
| 003_017_EN_Allister | 0.113 | 0.137 |
| 003_021_EN_Nicholson | 0.085 | 0.230 |
| 003_070_EN_McGuinness | 0.123 | 0.056 |
| 006_005_EN_Malmstrom | 0.896 | 0.168 |
| 006_006_EN_Hannan | 0.153 | 0.151 |
| 006_016_EN_Batten | 0.160 | 0.073 |
| 007_002_EN_Malmstrom | 0.172 | 0.127 |
| 007_004_EN_Hannan | 0.074 | 0.104 |
| 008_014_EN_Dodds | 0.062 | 0.044 |
| 008_054_EN_Karim | 0.152 | 0.161 |
| 009_010_EN_Barroso | 0.081 | 0.122 |
| 011_008_EN_Andreasen | 0.069 | 0.137 |
| 012_005_EN_Andreasen | 0.139 | 0.094 |
| 012_008_EN_Ojuland | 0.163 | 0.204 |
| 012_024_EN_Arlacchi | 0.083 | 0.329 |
| 012_038_EN_Savi | 0.118 | 0.234 |
| 013_014_EN_Bloom | 0.093 | 0.180 |
| 013_052_EN_Mitchell | 0.218 | 0.173 |
| 014_007_EN_Wallis | 0.186 | 0.171 |
| 015_012_EN_Kasoulides | 0.116 | 0.167 |
| 015_013_EN_Deva | 0.107 | 0.092 |
| 015_014_EN_Corazza-Bildt | 0.091 | 0.163 |
| 015_018_EN_Gallagher | 0.085 | 0.762 |
| 016_005_EN_Nicholson | 0.121 | 0.130 |
| 016_017_EN_Hannan | 0.328 | 0.125 |
| 016_032_EN_Vassiliou | 0.046 | 0.138 |
| 017_038_EN_McGuinness | 0.156 | 0.140 |
| Average | 0.155 | 0.169 |

Table 5: WER score (LPC) of ASR pipeline components, obtained with *ASReval*. First column is the folder name from ESIC dev2 data set, second column is WER score for the German ASR using English to German interpreter's speech, compared with the manual transcription of the speech. Third column is English ASR using the original speaker's English speech, compared with the manual transcription of the speech.

As mentioned before, performance of the ASR + MT pipelines is quite poor, which is caused by using transcriptions as a reference for evaluating that do not match the speech used as the input of the pipeline. However, as a quick soundness check, we can see the *de2en_pairwise* pipeline performed very similarly across the two reference sources. This does make sense, because the references are the transcripts of the original speaker's English speech and should be very similar. The *en2de_rainbow* pipeline, however, performed differently: this also makes sense, because one of the references was the interpreter's transcription and the other was German translation of the official English transcription. The pipeline was evaluated using the English speech of the original speaker, so it makes sense it scored better (10.1 sacreBLEU vs 19.8 sacreBLEU). Based

on these scores, we can speculate the negative impact of using the interpreter's speech as an audio source results in 9.7 sacreBLEU score loss on the data we used for evaluation.

A possible solution to alleviate this issue would be a different evaluation metric measuring the *semantics* of the translation, i.e. the accuracy of the translation from an informational point of view. This could be implemented using *embeddings*: convert the output of the MT system and the transcription to vectors and then compare these. With enough luck, comparing these vectors could allow for a measurement of semantics: if the vectors are similar (precisely, the cosine distance between them is small), then the sentences are also close together on a semantic level. This is, however, not a trivial task.

Another issue was the recordings of the interpreter's speeches and the original speaker speeches did not exactly match: for example, it is common for the audio file of the interpreter's speech to contain initial greetings by the speaker, but the audio file of the speaker does not contain these formalities. Naturally, this phenomena also occurs in the transcriptions. When these transcriptions are used as a reference, these formalities, which appear only on one side, cause the WER score to worsen. This was also the reason we omitted evaluation of timing, which *SLTeval* provides.

| **Source German speech:**<br>**Reference English translation:** | **EN to DE interpreter**<br>**Manual transcription** | **EN to DE interpreter**<br>**Official transcription** |
|---|---|---|
| File name in ESIC | de2en (manual) | de2en (official) |
| 003_007_EN_Swinburne | 7.3 | 7.3 |
| 003_017_EN_Allister | 7.9 | 6.5 |
| 003_021_EN_Nicholson | 2.5 | 9.8 |
| 003_070_EN_McGuinness | 6.1 | 8.7 |
| 006_005_EN_Malmstrom | 0.2 | 0.2 |
| 006_006_EN_Hannan | 5.4 | 6.5 |
| 006_016_EN_Batten | 14.8 | 15.9 |
| 007_002_EN_Malmstrom | 7.8 | 8.3 |
| 007_004_EN_Hannan | 5.9 | 4.9 |
| 008_014_EN_Dodds | 9.0 | 9.0 |
| 008_054_EN_Karim | 2.8 | 2.4 |
| 009_010_EN_Barroso | 11.6 | 11.8 |
| 011_008_EN_Andreasen | 13.7 | 13.2 |
| 012_005_EN_Andreasen | 16.2 | 19.8 |
| 012_008_EN_Ojuland | 6.0 | 2.8 |
| 012_024_EN_Arlacchi | 6.5 | 4.6 |
| 012_038_EN_Savi | 11.0 | 11.0 |
| 013_014_EN_Bloom | 7.0 | 8.1 |
| 013_052_EN_Mitchell | 4.3 | 3.8 |
| 014_007_EN_Wallis | 4.5 | 6.5 |
| 015_012_EN_Kasoulides | 23.1 | 19.6 |
| 015_013_EN_Deva | 16.7 | 13.0 |
| 015_014_EN_Corazza-Bildt | 11.9 | 11.6 |
| 015_018_EN_Gallagher | 11.2 | 3.8 |
| 016_005_EN_Nicholson | 10.7 | 11.7 |
| 016_017_EN_Hannan | 9.9 | 9.9 |
| 016_032_EN_Vassiliou | 10.3 | 10.8 |
| 017_038_EN_McGuinness | 5.4 | 5.0 |
| Average | 8.9 | 8.8 |

Table 6: sacreBLEU scores (mwerSegmenter) of the German to English SLT pipeline, obtained with *SLTeval*, using the ESIC dev2 data set. The reference files are the transcriptions of the speaker's English speech: one manual, provided by the creators of ESIC and one official, provided by European Parliament.

| **Source English speech:**<br>**German translation reference:** | **Original EN speech**<br>**Interpreter transcription** | **Original EN speech**<br>**Official DE translation** |
|---|---|---|
| File name in ESIC | en2de (transcription) | en2de (translation) |
| 003_007_EN_Swinburne | 3.9 | 20.6 |
| 003_017_EN_Allister | 3.3 | 16.4 |
| 003_021_EN_Nicholson | 6.2 | 5.9 |
| 003_070_EN_McGuinness | 6.0 | 16.3 |
| 006_005_EN_Malmstrom | 11.9 | 23.7 |
| 006_006_EN_Hannan | 3.1 | 23.0 |
| 006_016_EN_Batten | 16.5 | 23.0 |
| 007_002_EN_Malmstrom | 8.6 | 23.7 |
| 007_004_EN_Hannan | 7.4 | 21.0 |
| 008_014_EN_Dodds | 3.7 | 28.9 |
| 008_054_EN_Karim | 8.7 | 29.6 |
| 009_010_EN_Barroso | 9.3 | 17.3 |
| 011_008_EN_Andreasen | 18.7 | 26.5 |
| 012_005_EN_Andreasen | 10.8 | 25.2 |
| 012_008_EN_Ojuland | 10.3 | 29.4 |
| 012_024_EN_Arlacchi | 13.2 | 9.4 |
| 012_038_EN_Savi | 12.2 | 13.1 |
| 013_014_EN_Bloom | 9.8 | 25.1 |
| 013_052_EN_Mitchell | 6.3 | 13.6 |
| 014_007_EN_Wallis | 4.6 | 24.6 |
| 015_012_EN_Kasoulides | 18.8 | 16.6 |
| 015_013_EN_Deva | 42.4 | 11.8 |
| 015_014_EN_Corazza-Bildt | 11.4 | 11.9 |
| 015_018_EN_Gallagher | 2.3 | 12.6 |
| 016_005_EN_Nicholson | 3.3 | 17.4 |
| 016_017_EN_Hannan | 6.7 | 24.9 |
| 016_032_EN_Vassiliou | 12.0 | 21.0 |
| 017_038_EN_McGuinness | 10.1 | 21.9 |
| Average | 10.1 | 19.8 |

Table 7: sacreBLEU scores (mwerSegmenter) of the English to German SLT pipeline, obtained with *SLTeval*. The reference files are the official German translation of the official English speech provided by European Parliament and the transcription of the English to German interpreter.

| **Data** | **en2ar** | **en2cs** | **en2es** | **en2pl** | **en2sk** |
|---|---|---|---|---|---|
| 20200323-second-pre-dry-run | 2.8 | 10.0 | 18.5 | 12.4 | 13.4 |

Table 8: sacreBLEU scores (mwerSegmenter) of English ASR + RainbowMT pipeline, obtained with *SLTeval*. Languages are from left to right: Arabic, Czech, Spanish, Polish and Slovak. The data used are in `elitr-testset`: *documents/langtools-workshop/20200323-second-pre-dry-run.*

# 5 Automatic Minuting

This section describes the data that has been created for Work Package 5, i.e., Automatic Minuting. The goal behind creating the data is to organize a shared task on this novel problem of automatically generating minutes from meeting proceedings, a shared task first of its kind. Through this shared task, we wish to encourage community participation in this seemingly simple yet complex NLP task.

## 5.1 Dataset Description

We develop a dataset of English and Czech technical project meetings. The dataset consists of 113 meetings for English and 53 meetings for Czech, covering more than 160 hours (Table 9). The meetings are recorded, transcribed, equipped with minutes and de-identified. The test set consists of meeting transcripts and minutes, also with the recording of the meetings. A unique feature of this dataset is that most meetings are equipped with more than one minute, each created independently. Our data thus allows studying differences in what people find important while taking the minutes. To our best knowledge, no datasets for automatic minuting have been created yet. The two existing benchmark meeting datasets, AMI (McCowan et al., 2005) and ICSI (Janin et al., 2003), are aimed to meeting summarization, containing meeting transcripts, extractive summaries (selected relevant transcript lines) and abstractive summaries in the form of coherent paragraphs. The goal of minuting is to emphasize on topical coverage of the meeting, generate some bulleted key points where textual coherence is lesser important.

Our dataset consists of de-identified project meetings transcripts in English and Czech and their corresponding minutes. The English part includes project meetings from the computer science domain with prevailing non-native speakers of English. The meetings in the Czech part are from computer science and public administration domains, all meeting participants are native speakers of Czech. The length of the meetings varies from 10 minutes to more than 2 hours, but most meetings are about one hour long. In the dataset, a meeting usually contains one manually corrected transcript, one original minutes (created by a meeting participant; in some cases these minutes are actually a detailed agenda which got further updated after the meeting) and one or more generated minutes (by annotators). For some meetings, original minutes are missing, but each meeting must contain at least one generated minutes.

The dataset preparation consisted of the following stages:

1. Recording the online meetings on different video-conferencing tools
2. Transcribing the meetings via in-house ASR workers for English and Czech
3. Speaker Diarization and manually correcting the ASR output (transcripts) according to some predefined guidelines
4. Generating minutes of the meeting based on the manually corrected transcript and the recording
5. De-identifying the PERSON names, ORGANIZATION, PROJECT names from the transcripts and minutes
6. Obtaining consents from meeting participants to allow us to use the de-identified transcript and minutes for our shared task.

## 5.2 Shared Task description

We launch the first shared task on **Automatic Minuting** as a ISCA-endorsed satellite event of Interspeech 2021. Along with this, the event would also invite submissions to SIGDial on meeting and dialogue summarization, which are close to automatic minuting, however not the

| | English | | Czech | |
|---|---|---|---|---|
| | #meetings | #hours | #meetings | #hours |
| Meetings minuted once | 24 | 22 | 2 | 2 |
| Meetings minuted twice | 64 | 65 | 20 | 20 |
| Meetings minuted more than twice | 25 | 22 | 31 | 31 |
| **Total meetings** | 113 | 109 | 53 | 53 |

Table 9: Basic transcript and minutes statistics for our dataset.

same. Meeting minutes keep a record of what was discussed at a meeting. It is usually a written document with little or no structure (perhaps a hierarchical bulleted list) to inform the participants and non-participants of what happened during the meeting. *Automatic minuting* (Nedoluzhko and Bojar, 2019) tools would be useful to quickly comprehend the meeting contents. People adopt different styles when *taking notes* or *recording minutes* of the meeting. The minutes also depend on the category, the intended audience, and the meeting's goal or objective. However, Automatic Minuting is challenging due to the absence of agreed-upon guidelines, a variety of minuting practices, and lack of extensive background research. Our shared task would consist of one main task and two supporting tasks for two different meeting scenarios (technical meetings and parliamentary proceedings). The technical meetings would consist of data in English and also in Czech. Earlier, a special session on Speech Summarization was carried out in 2006 [10]. We think it is the right time to launch such a shared task effort given the increased dominance of deep learning and efficient language models in Speech and Natural Language Processing. This special session aims to rejuvenate the community interest and instigate discussions on automatic minuting, which is so relevant to the current scenario when most of our meetings have gone virtual.

We propose one main task and two subsidiary tasks. The subsidiary tasks are optional.

- **Main Task A:** The main task consists of automatically creating minutes from multiparty meeting transcripts. The generated minute would be evaluated both via automatic and manual metrics.
- **Task B:** Given a pair of a meeting transcript and minutes, the task is to identify whether the minutes belongs to the transcript. We found that this task could be challenging during our data preparation from meetings on similar topics, given the similarity in various named entities.
- **Task C:** Given a pair of minutes, the task is to identify whether the two minutes belong to the same or different meetings. This sub-task is important as we want to uncover how minutes created by two different persons for the same meeting may differ in content and coverage.

[10] `http://homepages.inf.ed.ac.uk/jeanc/SpeechSummarization06.html`

# 6 Conclusion

In this deliverable, we presented new additions to `elitr-testset`, including parallel corpora for low-resource European languages, such as Montenegrin. We described `SLTev`, a tool for evaluating SLT components and its integration with `elitr-testset`. We presented our evaluation strategies: either evaluating individual components, or whole pipelines. Finally, we described the Automatic Minuting dataset and its shared task on Interspeech 2021.

# References

Ebrahim Ansari, Ondřej Bojar, Barry Haddow, and Mohammad Mahmoudi. 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, April 2021. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2021.eacl-demos.9`.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. The ICSI Meeting Corpus. pages 364–367, 2003.

Jonáš Kratochvíl, Peter Polák, and Ondřej Bojar. Large corpus of czech parliament plenary hearings. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6363–6367, 2020.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6124–6128. IEEE, 2020.

Dominik Macháček, Jonáš Kratochvíl, Tereza Vojtěchová, and Ondřej Bojar. A speech test set of practice business presentations with additional relevant texts. In *Statistical Language and Speech Processing*, pages 151–161, Cham, Switzerland, 2019. Springer Nature Switzerland AG. ISBN 978-3-030-31371-5. URL `http://hdl.handle.net/11234/1-3023`.

Dominik Macháček, Matúš Žilinec, and Ondřej Bojar. Lost in interpreting: Speech translation from source or interpreter? In *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association*. ISCA, to be published 2021.

I. McCowan, G. Lathoud, M. Lincoln, A. Lisowska, W. Post, D. Reidsma, and P. Wellner. The AMI Meeting Corpus. In *In: Proceedings Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research. L.P.J.J. Noldus, F. Grieco, L.W.S. Loijens and P.H. Zimmerman (Eds.), Wageningen: Noldus Information Technology*, 2005.

Anna Nedoluzhko and Ondrej Bojar. Towards automatic minuting of the meetings. In *ITAT*, pages 112–119, 2019.

# A SLTev: Comprehensive Evaluation of Spoken Language Translation

## 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



*Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 71–79
April 19 - 23, 2021. 

| | | | |
|---|---|---|---|
| P | 0 | 50 | Good |
| P | 0 | 65 | Good mor |
| P | 0 | 119 | Good morning how |
| P | 0 | 195 | Good morning. How are you? |
| C | 0 | 102 | Good morning. |
| P | 102 | 218 | How are you? I |
| C | 102 | 195 | How are you? |
| P | 195 | 239 | I am |
| ... | ... | ... | ... |

(a) Time-stamped transcript

| | | | | |
|---|---|---|---|---|
| P | 60 | 0 | 50 | Gut |
| P | 80 | 0 | 65 | Guten Morgen! |
| P | 130 | 0 | 119 | Guten wie morgen |
| P | 201 | 0 | 195 | Guten Morgen! Wie geht es dir? |
| C | 201 | 0 | 102 | Guten Morgen! |
| P | 220 | 102 | 218 | Wie geht es dir? Ich |
| C | 220 | 102 | 195 | Wie geht es dir? |
| P | 245 | 195 | 239 | Ich bin |
| ... | ... | ... | ... | ... |

(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 $j$th 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 $j$th word in the $i$th segment of the reference translation. First we estimate the times of *source* words in the $i$th 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:

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

$t_x$ is the expected time of the $x$th word of the *source* sentence $i$ and $\lfloor \cdot \rceil$, $\lceil \cdot \rceil$ 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 $j^{\text{th}}$ 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 $i$th 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

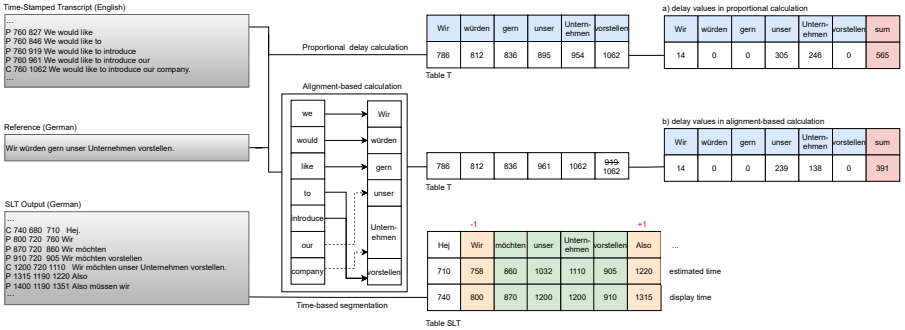


Figure 3: Time-based segmentation for proportional (a) and alignment-based (b) delay calculation. Using time-stamped transcript and reference translation, Table T is pre-computed. Then using timings in SLT output, word-level timestamps are estimated ("Table SLT"). The a) and b) value of delay is the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

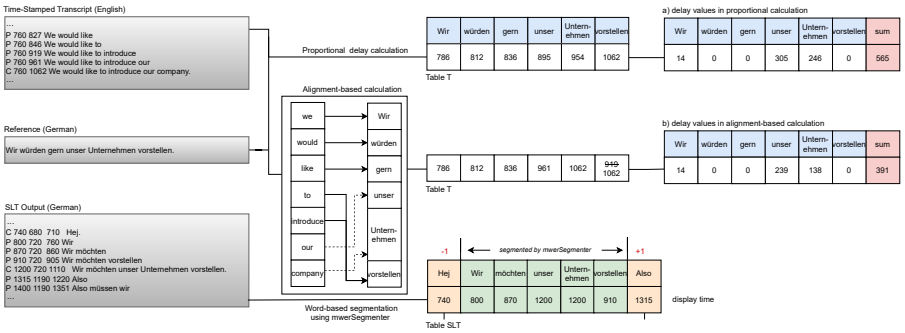


Figure 4: Word-based segmentation (*mwerSegmenter*) for proportional (a) and alignment-based (b) delay calculation. The main difference from Figure 3 is in finding the span of words that form the candidate segment, i.e. contribute to "Table SLT". Table SLT now needs to contain only display time of words.

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.