Change Log

Version	Date	Changes
1.3	2021-08-31	Added section 9, covering scoring procedures
1.2	2021-08-12	Section 4: Added clarification on Constrained-plus training condition
1.1	2021-08-06	Sections 4, 7.3: Added Constrained-plus Training condition
1.0	2021-07-30	Initial release

OpenASR21 Challenge
(Open Automatic Speech Recognition 2021 Challenge)
Evaluation Plan

Contents

1. INTRODUCTION

The OpenASR21 Challenge is the third open challenge created out of the Intelligence Advanced Research Projects Activity (IARPA) Machine Translation for English Retrieval of Information in Any Language (MATERIAL) program[1]. MATERIAL encompasses additional tasks, including cross-language information retrieval, domain classification, and summarization, as well as more languages. For every year of MATERIAL, the National Institute of Standards and Technology (NIST) has organized a simplified, smaller scale evaluation open to anyone wishing to participate, focusing on a particular technology aspect of MATERIAL. In 2019, Cross-Language Information Retrieval (CLIR) technologies were the focus of the open challenge, OpenCLIR.[2] In 2020, the focus was on Automatic Speech Recognition (ASR) under low-resource language constraints for the first time.[3] In 2021, ASR under low-resource language constraints is being offered again but with new languages and case-sensitive scoring added.

The capabilities tested in the open challenges are expected to ultimately support the MATERIAL task of effective triage and analysis of large volumes of data in a variety of less-studied languages.

The OpenASR21 Challenge is being implemented as a track of NIST's OpenSAT (Open Speech Analytic Technologies) evaluation series,[4] using the OpenSAT infrastructure for registration, submission, and scoring purposes.

This evaluation plan as well as any additional documentation and tools are available via NIST's OpenASR website.[5]

2. CHALLENGE TASK

The OpenASR21 Challenge task consists of performing ASR on audio datasets in 15 low-resource languages, producing the recognized written text. Participating teams may choose to attempt as many of the offered languages as they wish. Ten of the languages are those that were offered already in OpenASR20, and five are new for 2021. Three of the new languages will feature additional evaluation datasets for which the output will be scored using case-sensitive criteria. Case-sensitive scoring is used as a proxy for evaluating ASR performance on proper nouns.

3. METRICS

3.1 WORD ERROR RATE (WER)

The primary metric computed on the submitted output is Word Error Rate (WER), as implemented in the sclite tool of the Speech Recognition Scoring Toolkit SCTK available from NIST.[6] WER is computed as the sum of deletion, insertion, and substitution errors in the ASR output compared to the human reference transcription, divided by the total number of words in the human reference transcription:

$$WER = \frac{\#Deletions + \#Insertions + \#Substitutions}{\#ReferenceWords}$$

[1] https://www.iarpa.gov/index.php/research-programs/material
[2] https://www.nist.gov/itl/iad/mig/openclir-challenge
[3] https://www.nist.gov/itl/iad/mig/openasr-challenge#openasr20
[4] https://www.nist.gov/itl/iad/mig/opensat
[5] https://www.nist.gov/itl/iad/mig/openasr-challenge
[6] https://github.com/usnistgov/SCTK

WER for a dataset will be computed by the total number of errors over the total number of reference words in the dataset. WER will be calculated case-insensitive, with the exception of the designated case-sensitive tasks. Character Error Rate (CER) will be calculated and reported as well.

3.2 TIME AND MEMORY RESOURCES

Teams are required to self-report time and memory resources used by their ASR system(s). Time and memory resources reported are used to compute a run time factor (compared to the real time of the audio data processed) as a secondary metric to provide the community with information about the resources required to use their ASR systems. The requirements for reporting time and memory resources are specified in section 9.

4. TRAINING CONDITIONS AND LIMITATIONS

The OpenASR21 Challenge offers three different training conditions: Constrained, Constrained-plus, and Unconstrained. For any language processed, teams must make a submission for the Constrained Training condition. The other two training condition are optional but encouraged.

In the ***Constrained Training*** condition, the only *speech* data permissible for training is a 10-hour Build dataset provided by NIST for the language being processed. Additional text and non-speech acoustic data from publicly available (obtainable by anyone for free or a fee) resources is permissible for training in the Constrained Training condition.

The ***Constrained-plus Training*** condition follows the same training data restrictions as Constrained Training, but additionally allows publicly available *and previously existing* (obtainable by anyone for free or a fee since before the first day of registration) speech pretrained models as follows:

- Pretrained models created from *unlabeled* speech data in any language
- Pretrained models created from *labeled* speech data in any language *except* the language being processed

In the ***Unconstrained Training*** condition, teams may use all of the data allowed above as well as additional publicly available (obtainable by anyone for free or a fee) speech and text data and models from any language.

The training data provided by NIST under the OpenASR21 participation and data usage agreement does not qualify as publicly available. This means that, if a participant signs up for more than one language, they may not use any training data, speech or text, received for other languages for the language being processed, for any of the three conditions. The same holds when signing up for both case-insensitive scoring (CIS) and case-sensitive scoring (CSS) datasets for a particular language (see section 5). No training data provided for one of those datasets may be used for training for a submission on the other dataset.

Teams may not utilize native speakers for data acquisition, system development, or analysis in any of the training conditions.

Table 1 sums up the training data and resource limitations for the three training conditions, while Table 2 lists specific types of additional data and their permissibility for the three training conditions. Any permissible additional training data used in any of the training conditions must be specified in sufficient detail in the system description (see section 12).

Resource	Constrained Training	Constrained-plus Training	Unconstrained Training
Speech data and models	Limited to 10-hour Build dataset provided by NIST for the language being processed	Constrained Training data, plus publicly available: • Pretrained models from unlabeled speech data in any language • Pretrained models from labeled speech data in any language except the language being processed	Unlimited publicly available
Non-speech acoustic data and models	Unlimited publicly available		
Text data and models			
Native speaker expertise applied to language being processed	Not permissible		

Table 1: Training data and resource limitations by training condition

Publicly available **(obtainable by anyone for free or a fee) permissible data beyond designated Constrained Training data provided for the language being processed**	**Constrained Training**	**Constrained-plus Training**	**Unconstrained Training**
Speech data in any language (incl. vocal music and artificially generated babble)	No	No	Yes
Pretrained models trained on *unlabeled speech* data in any language (incl. vocal music and artificially generated babble)	No	Yes	
Pretrained models trained on *labeled speech* data in any language except the language being processed (incl. vocal music and artificially generated babble)	No	Yes	
Text-to-speech (TTS) output from TTS trained on designated Constrained Training data	Yes	Yes	
Non-speech acoustic data (non-vocal music, noise, filters, etc.)	Yes	Yes	
Text data in any language	Yes	Yes	
Pretrained models trained on labeled or unlabeled text data in any language	Yes	Yes	

Table 2: Additional data resource permissibility by training condition

5. LANGUAGES AND CASING

The OpenASR21 Challenge is offered for the fifteen low-resource languages listed in Table 3. All ten languages from OpenASR20 will be offered again. Five languages are new for 2021. There will be a main case-insensitive scoring (CIS) evaluation (Eval) dataset for all fifteen languages. Three of the new languages will be offered with an additional case-sensitive scoring (CSS) Eval dataset.

System output on the ***CIS Eval datasets*** will be scored using ***case-insensitive scoring***. For the ten languages that are being repeated from OpenASR20, these CIS Eval datasets will remain identical to OpenASR20, allowing for comparability over time.

System output on the additional, i.e. ***CSS Eval datasets*** offered for Kazakh, Swahili, and Tagalog will be scored using ***case-sensitive scoring***, i.e. words capitalized differently from the reference transcript will not count as a match.

For any language attempted, processing the CIS Eval dataset will be mandatory. Processing the additional CSS Eval dataset, where applicable for the language, will be optional.

Language	New in OpenASR2021	CIS (case-insensitive scoring)	CSS (case-sensitive scoring)
Amharic	-	Yes	-
Cantonese	-	Yes	-
Farsi	Yes	Yes	-
Georgian	Yes	Yes	-
Guarani	-	Yes	-
Javanese	-	Yes	-
Kazakh	Yes	Yes	Yes
Kurmanji Kurdish	-	Yes	-
Mongolian	-	Yes	-
Pashto	-	Yes	-
Somali	-	Yes	-
Swahili	Yes	Yes	Yes
Tagalog	Yes	Yes	Yes
Tamil	-	Yes	-
Vietnamese	-	Yes	-

Table 3: Languages and evaluation dataset types

6. DATA GENRES

The CIS Eval dataset offered for all languages will consist of only one data genre, conversational telephone speech. The CSS Eval datasets for three of the languages will consist of a mix of data genres. The genres are listed in Table 4.

Genre	Shorthand[7]	Description	CIS Eval dataset (all languages)	CSS Eval dataset (Kazakh, Swahili, Tagalog)
Conversational telephone speech	CS	Conversations between two people over the phone on a topic of their choosing	Yes	Yes
News broadcast	NB	Audio segments from news-related broadcasts	-	Yes
Topical broadcast	TB	Audio segments from broadcasts covering a topic in depth	-	Yes

Table 4: Data genres

7. RESOURCES

Datasets for system training (Build), development (Dev), and evaluation (Eval) will be made available under an OpenASR21 Challenge participation and data usage agreement for each of the languages. This agreement will be made available during the registration process. For the languages with CIS and CSS Eval sets, there will be separate Build and Dev sets to go with the two different Eval sets, reflecting the difference in genre makeup between them.

For CIS, the datasets for most of the languages stem from the IARPA Babel program[8]. The Somali and Farsi datasets stem from the IARPA MATERIAL program[9] .

For CSS, all datasets stem from the IARPA MATERIAL program.

Table 5 shows the approximate sizes of speech datasets provided per language and Eval set (CIS and CSS).

Build (training)	Dev	Eval
10 hours	10 hours	5 hours

Table 5: Provided speech dataset sizes per language and Eval set (CIS and CSS)

7.1 AUDIO DATA SPECIFICATIONS

Table 6 summarizes the main audio data specifications.

[7] In other contexts and programs, the abbreviations CTS (Conversational Telephone Speech) and BN (Broadcast News) are commonly used for the CS and NB genres respectively.

[8] https://www.iarpa.gov/index.php/research-programs/babel

[9] https://www.iarpa.gov/index.php/research-programs/material

Genre	Channels	Sampling rate	File format
CS	2 (each channel distributed separately)	8 kHz or 44.1 kHz or 48 kHz	.sph or .wav
NB	1		.wav
TB	1		

Table 6: Audio data specifications

The sampling rate is provided in each audio file's header.
The Babel CS data consists of conversations between two persons over the telephone on a topic of their choosing. Conversations vary in length, up to approximately 10 minutes. More details regarding the Babel CS audio data, including transcription conventions, can be found in section 3 of the IARPA Babel Data Specifications for Performers.[10]
MATERIAL CS data is either a subset of Babel CS data or newer sets collected and annotated using the Babel methodology. MATERIAL NB data consists of audio segments of approximately 2.5 minutes from widely distributed broadcasts as well as regional and local news covering news topics and current affairs. The broadcasts are studio quality, while the speech could be formal or informal depending on the segments. MATERIAL TB data is similar to NB in terms of audio quality and speech characteristics but devoted to in-depth topics and approximately five minutes in duration. A description of the MATERIAL transcription conventions is available on NIST's OpenASR website.[11]

7.2 Text Data Specifications

The format of the transcripts provided in the Build datasets is as follows:

```
[start time]
transcript
[start time]
transcript
[start time]
```

where the start time of the next transcript is also the end time of the previous transcript.

Example:

```
[1.34]
HOW ARE YOU
[5.10]
CAN YOU COME HERE
[6.78]
GREAT THANK YOU
[7.55]
BYE HAVE A NICE DAY <no-speech>
```

[10] https://www.nist.gov/system/files/documents/itl/iad/mig/IARPA_Babel_Performer-Specification-08262013.pdf

[11] https://www.nist.gov/system/files/documents/2021/07/27/MATERIALTranscriptionConventions.pdf

7.3 DATA USAGE RULES AND RESTRICTIONS

Participants will be required to sign and abide by a participation and data usage agreement. No data or annotations received under this agreement may be distributed outside of purposes directly related to the OpenASR21 Challenge.

The rules governing the use of the different dataset are outlined in Table 7.

Activity	**Build**	**Dev**	**Eval**
Manually examine data before the end of the evaluation	Yes	No	No
Manually examine data after the end of the evaluation	Yes	Yes	No[12]
Train models using released data	Yes	No	No
Parameter tuning	Yes	Yes	No
Score locally	Yes	Yes	No

Table 7: Dataset rules

The following additional rules and restrictions apply:

- Teams may examine the Build datasets for and use them for system training.
- Teams may use the Dev datasets to test their systems and may also use it as a held-out dataset to set the values of general system hyper-parameters. Teams may not use the Dev datasets for system training.
- In the Constrained Training condition, no additional speech training data beyond what is provided under the OpenASR21 Challenge participation and data usage agreement for the language being processed may be used.
- In the Constrained-plus Training condition, speech training data is limited to Build data provided under the OpenASR21 Challenge participation and data usage agreement for the language being processed, as well as publicly available (obtainable by anyone for free or a fee) pretrained models limited as specified in section 4.
- In the Unconstrained Training condition, teams may mine the web for additional publicly available (obtainable by anyone for free or a fee) training data. Any such data harvested for training must be specified in the system description.
- Teams may not utilize native speakers for data acquisition, system development, or analysis. For example, it is forbidden to use native speaker consultants to find or post-process any data.
- The Eval dataset must be treated as a blind test.
- Data crawling may not continue during the evaluation period. All machine learning or statistical analysis algorithms should complete training, model selection, and tuning prior to running on the Eval datasets. This rule does not preclude online learning/adaptation during Eval dataset processing at evaluation time, as long as the adaptation information is not reused for subsequent runs on the Eval datasets. Teams must document the ways their online learning and adaptation approaches incorporate information extracted from the Eval datasets in the system description.

[12] Sequestration for potential reuse to track progress in future evaluations

- Teams may not use third-party ASR commercial software in any part of their pipeline.

8. REFERENCE AND SYSTEM OUTPUT FILE FORMATS AND NORMALIZATION

The NIST scoring server uses segment time mark (STM) and conversation time mark (CTM) for its reference and system output file formats, respectively.

A toolkit for file conversion, normalization, and validation is available on NIST's OpenASR website.[13]

8.1 REFERENCE FILE FORMAT (STM)

The reference files on the scoring server follow one of the following two formats, depending on whether the audio was recorded in one or two channels:

`<DocID>_<ChannelID>.stm` (for one of two channels of two-channel recording)
`<DocID>.stm` (for single channel recording)

The following are examples of reference file names:

```
MATERIAL_OP2-3C_38293787_inLine.stm
MATERIAL_OP2-3C_38293787.stm
BABEL_BP_101_98675_20111117_190458_inLine.stm
MATERIAL_OP2-3S-BUILD_11416_20160410_185125_outLine.stm
```

The STM file consists of several fields to form a record. Each record is separated by a newline and contains: the waveform's filename, the channel identifier, the speaker ID, the begin time, the end time, and the transcript of the segment. Each record follows this Backus-Naur form (BNF) notation:

```
STM :== <F> <C> <S> <BT> <ET> transcript . . .
```

where:

`<F>` Waveform base filename. No path names or extensions expected.
`<C>` Waveform channel (numeric). 1 (inline, or default for single channel) or 2 (outline).
`<S>` Speaker id. No restrictions on this name.
`<BT>` Begin time (seconds) of segment.
`<ET>` End time (seconds) of segment.
`transcript` Transcript, in one of three forms:

1. a whitespace separated list of words
2. empty string
3. the string "`IGNORE_TIME_SEGMENT_IN_SCORING`". When the string "`IGNORE_TIME_SEGMENT_IN_SCORING`" is used as the transcript, the process that chops the hypothesis file into matching reference segments ignores all hypothesis words whose time-midpoints occur within the reference segment's beginning and ending time. The effect is to make these segment regions "out-of-bounds" for scoring, thus generating no errors from that time region.

[13] https://www.nist.gov/itl/iad/mig/openasr-challenge

Example:

```
BABEL_BP_101_98675_20111117_190458 1 BABEL_BP_101_98675_20111117_190458_1 1.34 3.84 HOW ARE YOU
BABEL_BP_101_98675_20111117_190458 1 BABEL_BP_101_98675_20111117_190458_1 5.10 6.78 CAN YOU COME HERE
BABEL_BP_101_98675_20111117_190458 1 BABEL_BP_101_98675_20111117_190458_1 9.01 10.56 GREAT THANK YOU
:
BABEL_BP_101_98675_20111117_190458 2 BABEL_BP_101_98675_20111117_190458_2 4.06 4.56 I AM GOOD
BABEL_BP_101_98675_20111117_190458 2 BABEL_BP_101_98675_20111117_190458_2 7.14 8.16 YES I CAN
BABEL_BP_101_98675_20111117_190458 2 BABEL_BP_101_98675_20111117_190458_2 11.40 12.05 SURE I AM COMING
```

8.2 SYSTEM OUTPUT FORMAT (CTM)

The system output files on the scoring server must follow one of the following two formats, depending on whether the audio was recorded in one or two channels:

`<DocID>_<ChannelID>.ctm` (for one of two channels of two-channel recording)
`<DocID>.ctm` (for single channel recording)

The basename of the file must match the name of the corresponding reference file.

For example, the system output files to match the reference file examples from section 8.1 would be named:

```
MATERIAL_OP2-3C_38293787_inLine.ctm
MATERIAL_OP2-3C_38293787.ctm
BABEL_BP_101_98675_20111117_190458_inLine.ctm
MATERIAL_OP2-3S-BUILD_11416_20160410_185125_outLine.ctm
```

The CTM file is a concatenation of time mark records for each word in each channel of a waveform. The records are separated with a newline. Each word token must have a waveform id, channel identifier, start time, duration, and word text. Optionally, a confidence score can be appended for each word. Each record follows this BNF notation:

```
CTM :== <F> <C> <BT> <DUR> word [ <CONF> ]
```

where:

<F> Waveform base filename. No path names or extensions expected.
<C> Waveform channel. 1 (inline, or default for single channel) or 2 (outline).
<BT> Begin time (seconds) of token, measured from start time of file.
<DUR> Duration (seconds) of token.
<CONF> Optional confidence score. Currently, this field is not being used in sclite.

Example:

```
BABEL_BP_101_98675_20111117_190458 1 11.34 0.2 YES
BABEL_BP_101_98675_20111117_190458 1 12.00 0.34 YOU
BABEL_BP_101_98675_20111117_190458 1 13.30 0.5 CAN
:
BABEL_BP_101_98675_20111117_190458 2 1.34 0.2 I
BABEL_BP_101_98675_20111117_190458 2 2.00 0.34 CAN
BABEL_BP_101_98675_20111117_190458 2 3.40 0.5 ADD
```

8.3 REFERENCE FILE NORMALIZATION

The transcripts provided in the Build datasets contain not only the transcription of the speech in the audio file but also speech aspects such as mispronunciations and non-speech aspects such as coughs. Additionally, depending on the language, the transcript may also contain zero-width non-joiner characters (ZWNJ, U+200C) for rendering purposes. Prior to scoring with sclite, normalization is performed so the WER obtained is as precise as possible. The normalization is outlined in Table 8.

Feature	**Normalization**	**Example, original**	**Example, converted for SCLITE scoring**
<hes>	optionally deletable	I <hes> would like	I (<hes>) would like
* *	optionally deletable	I don't like his *facade*	I don't like his (facade)
<foreign>	optionally deletable	<foreign> wait for me	(<foreign>) wait for me
<no-speech>	delete tag	<no-speech>	
<overlap>	exclude segment from scoring	<overlap>	IGNORE_TIME_SEGMENT_IN_SCORING
~	delete tag	~ contemplation	contemplation
(()) <sta> <int> <misc> <lipsmack> <breath> <cough> <laugh> <click> <ring> <dtmf> <male-to-female> <female-to-male>	delete tag	<lipsmack>	

Feature	**Normalization**	**Example, original**	**Example, converted for SCLITE scoring**
<prompt>	exclude segment from scoring	<prompt>	IGNORE_TIME_SEGMENT_IN_SCORING
_ (underscore)	change to space	N_I_S_T	N I S T
- (dash)	optionally deletable	I communica- to him	I (communica-) to him
/ /	delete tag	/B/	B
-- (double dash)	delete tag	I will go -- I will go there tomorrow	I will go I will go there tomorrow
%incomplete	delete tag	Go to the %incomplete	Go to the
. (period) ? (question mark) ! (exclamation point)	delete period, question mark, exclamation, point, and potential preceding space	I will go tomorrow. I will go tomorrow .	I will go tomorrow I will go tomorrow
, (sentence-internal comma)	delete sentence-internal comma	Since I will go there tomorrow, you won't have to	Since I will go there tomorrow you won't have to
" (double quote)	delete character	I will "go" tomorrow	I will go tomorrow
U+200C (zero-width non-joiner, ZWNJ)	delete character	I use English to demon\u200cstrate ZWNJ	I use English to demonstrate ZWNJ
= (used as ZWNJ in Farsi)	delete character	I use English to demon=strate ZWNJ	I use English to demonstrate ZWNJ

Table 8: Normalization

A clarification note on the normalization items marked *optionally deletable* in the table above, using the example of <hes>: The original reference marks hesitation noises (such as "um", "uh", "ah") as <hes>. <hes> is then converted to (<hes>) during normalization. The parentheses ( ) in the converted reference tells SCLITE to treat the token as "optionally deletable", which means if the system does not output it, the system will not be penalized. However, if the system does output it and the token is not identical, the system will be penalized. Therefore, it is to a system's advantage not to output optionally deletable tokens.

A description of the Babel transcription conventions can be found in section 5 of the IARPA Babel Data Specifications for Performers on NIST's OpenASR website. [14] A description of the MATERIAL transcription conventions is also available on NIST's OpenASR website. [15]

9. SCORING PROCEDURES AND EXAMPLES

Prior to scoring with sclite, the CTM system file is sorted using:

```
sort +0 -1 +1 -2 +2nb -3 hypFile.ctm > hypFile.ctm.sort
```

and the STM reference file is sorted using:

```
sort -k 1,2 -k4n refFile.stm > refFile.stm.sort
```

For CSS datasets, the `-s` option is used with sclite to perform case-sensitive calculation.

For Guarani, Kurmanji Kurdish, Mongolian, Vietnamese, or Kazakh, the case-conversion localization information is added to the end of the `-e` option `-e utf-8.`, e.g. `-e utf-8 babel_kazakh`.

The following commands are used for scoring. Each command is followed by two examples.

WER, case-insensitive calculation:

```
./bin/sclite -r refFile.stm.sort stm -h hypFile.ctm.sort ctm -F -D -O outputDir1/ -o sum rsum pralign prf -e utf-8 [ case-conversion-localization ]
```

WER, case-insensitive calculation *examples*:

```
./bin/sclite -r swahili_cis.stm.sort stm -h swahili_cis.ctm.sort ctm -F -D -O outputDir1/ -o sum rsum pralign prf -e utf-8
./bin/sclite -r kazakh_cis.stm.sort stm -h kazakh_cis.ctm.sort ctm -F -D -O outputDir2/ -o sum rsum pralign prf -e utf-8 babel_kazakh
```

WER, case-sensitive calculation:

```
/bin/sclite -r refFile.stm.sort stm -h hypFile.ctm.sort ctm -F -D -s -O outputDir1/ -o sum rsum pralign prf -e utf-8 [ case-conversion-localization ]
```

WER, case-sensitive calculation *examples*:

```
./bin/sclite -r swahili_css.stm.sort stm  -h swahili_css.ctm.sort ctm -F -D -s -O outputDir1/ -o sum rsum pralign prf -e utf-8
./bin/sclite -r kazakh_css.stm.sort stm -h kazakh_css.ctm.sort ctm -F -D -s -O outputDir2/ -o sum rsum pralign prf -e utf-8 babel_kazakh
```

CER, case-insensitive calculation:

```
./bin/sclite -r refFile.stm.sort stm -h hypFile.ctm.sort ctm -F -D -c NOASCII DH -O outputDir2/ -o sum rsum pralign prf -e utf-8 [ case-conversion-localization ]
```

CER, case-insensitive calculation *examples*:

```
./bin/sclite -r swahili_cis.stm.sort stm  -h swahili_cis.ctm.sort ctm -F -D -c NOASCII DH -O outputDir1/ -o sum rsum pralign prf -e utf-8
```

[14] https://www.nist.gov/system/files/documents/itl/iad/mig/IARPA_Babel_Performer-Specification-08262013.pdf
[15] https://www.nist.gov/system/files/documents/2021/07/27/MATERIALTranscriptionConventions.pdf

```
./bin/sclite -r kazakh_cis.stm.sort stm -h kazakh_cis.ctm.sort ctm -F -D -c NOASCII DH -O outputDir2/ -o sum rsum pralign prf -e utf-8 babel_kazakh
```

CER, case-sensitive calculation:

```
./bin/sclite -r refFile.stm.sort stm -h hypFile.ctm.sort ctm -F -D -c NOASCII DH -s -O outputDir2/ -o sum rsum pralign prf -e utf-8 [ case-conversion-localization ]
```

CER, case-sensitive calculation *examples*:

```
./bin/sclite -r swahili_css.stm.sort stm -h swahili_css.ctm.sort ctm -F -D -c NOASCII DH -s -O outputDir1/ -o sum rsum pralign prf -e utf-8
./bin/sclite -r kazakh_css.stm.sort stm -h kazakh_css.ctm.sort ctm -F -D -c NOASCII DH -s -O outputDir2/ -o sum rsum pralign prf -e utf-8 babel_kazakh
```

10. REPORTING TIME AND MEMORY RESOURCES

While teams are required to report time and memory resources as a secondary metric, it is important to note that wall-clock timing and memory usage are unstable measures; they are extremely sensitive to even minor changes in architectures and load. Differences of less than an order of magnitude are likely insignificant. Comparisons between systems based on these numbers should be performed with this in mind.

For processing the Eval datasets, teams must report, with each Eval submission:

1. Elapsed wall-clock time,
2. Total processing time,
3. Required memory.

All processing stages are counted together for the purpose of calculating these measures.

Teams are given some discretion in how to calculate time and memory resources.

`/usr/bin/time -v` is recommended as a low-overhead solution to report time and memory usage. It provides a single method to retrieve timing and memory usage information.

Other timing and memory profiling resources are acceptable, including custom code inserted into system modules. This timing must cover all operations as if the execution were timed by a wrapping utility such as `/usr/bin/time -v`.

It is easiest to calculate these numbers if each processing step is executed by a single command. In this case, simply timing these commands generates information for reporting elapsed time and memory. However, this is a simplified view of how systems may run. Information for resource reports may be constructed from information obtained from running sub-modules independently, as described below.

10.1 TIME

Time reporting requires two measures:

1. Elapsed wall-clock time: The amount of time that a user needs to wait for each processing stage to complete.
2. Total processing time: The total amount of central processing unit (CPU) and graphics processing unit (GPU) time required.

The rationale for including both measures is to provide information about how much improvement can be gained from additional cores (or alternatively, how much performance would suffer on an architecture with fewer available cores).

For parallel sub-processes on multiple cores:

- Grid Engine and other governing processes report timing information for spawned sub-processes. Also, many are compatible with time solutions that work for serial operations.
    - For Grid Engine, the `ru_wallclock` variable in the log file is an acceptable time option.
- For customized parallel solutions, performers are responsible for generating comparable timing solutions able to generate the maximum and total time required for parallel processing and for documenting their timing solution.

10.1.1 Elapsed Wall-Clock Time

Use "real" time consistent with that reported by `/usr/bin/time -v` for reporting elapsed wall-clock time.

The times of all serial processes are summed to compute elapsed wall-clock time.

For each step executed in parallel, the maximum time for the parallel processes is added to the elapsed wall-clock time.

10.1.2 Total Processing Time

Use "real" time consistent with that reported by `/usr/bin/time -v` for reporting total processing time.

The times of all serial and all parallel processes are summed to compute total processing time.

10.1.3 Time for GPUs

The use of GPUs can obfuscate the total CPU time required. GPUs contain thousands of cores; it is non-trivial to get information about the usage of each core during processing to report elapsed wall-clock time and total processing time in a way that is comparable to hundreds of traditional CPU cores.

Thus, GPU computation time is reported separately from traditional CPU time. GPU processes cover all modules that interact with a GPU. Sub-modules may perform pre-processing stages, then send data to a GPU for processing, then perform post-processing stages. If using a wrapping time procedure, this is considered a GPU process. If using a different timing solution, the time on CPU and time on GPU can be isolated and reported separately.

10.2 Memory

The goal in reporting memory usage is to describe the minimum memory required to execute the system processes within the times reported.

Minimally, performers should report the total available memory to ASR processes. However, the ASR software may not, in fact, need the total available memory; in this case, performers can use a profiler or other resource to identify the amount of required memory.

Since memory reporting is used to know how much memory an environment must have in order to run an ASR system, memory usage is calculated as the maximum memory used by any sub-process.

Parallel sub-processes on multiple cores can be measured independently with the maximum calculated as a post hoc processing for generating the report.

10.2.1 MEMORY FOR GPUS

If available, the maximum amount of memory concurrently allocated onto the GPU may be reported. However, the maximum memory used on a GPU is expected to be roughly equivalent to the maximum memory available on the GPU. Thus, rather than reporting the actual used memory, maximum available GPU memory may be reported.

GPU memory is reported separately from traditional memory.

10.3 SAMPLE REPORT

Below is a sample resource report. Teams are required to follow the formatting of this sample. The resource report is a required step in completing a submission.

```
Elapsed wall-clock time (hh:mm:ss) - 1:23:45.67
Total CPU time (hh:mm:ss) - 12:34:56.78
Total GPU time (hh:mm:ss) - 12:34:56.78
Maximum CPU memory (gigabytes) - 256
Maximum GPU memory (gigabytes) - 256
```

11. REGISTRATION, DATA ACCESS, SUBMISSIONS, AND SCORING

To participate in the OpenASR21 Challenge, participants must complete the registration process via NIST's OpenSAT web server[16] and abide by the participation and data usage agreement completed during the registration process. Dataset access is provided via a shared drive, and submissions must be made via the web server. NIST will provide automated scoring for valid submissions via the web server.

11.1 SUBMISSION LIMITS AND FEEDBACK

Teams can submit their system output on the different datasets for scoring to help their system development. Submission limits are listed in Table 9, along with the type of feedback provided.

Each submission will be validated prior to scoring. Only submissions that pass the validation step will count toward the submission limit. Submissions must follow the specified submission format. A toolkit for file conversion, normalization, and validation is available on NIST's OpenASR website.[17]

Submission quotas and feedback availability are specified in Table 9.

[16] https://sat.nist.gov
[17] https://www.nist.gov/itl/iad/mig/openasr-challenge

Timeline	Datasets	Limit per week[18], language, training condition, and dataset type (where applicable)	Feedback (score)
Development period	Dev	unlimited	yes
Evaluation period	Eval	5	yes, only overall score

Table 9: Submission quotas and feedback

11.2 SUBMISSION FORMAT

Each submission must be an archive file named as follows:

`<SysLabel>.tgz`

<SysLabel> is an alphanumeric [a-zA-Z0-9] label. This label is in part created from hard-coded information and team account information.

There should be no parent directory when the submission file is uncompressed.

The server validates the submission file content to make sure the system output files conform to the required system output format described in section 8.2.

In addition to the system output itself, the web server requires each submission to be accompanied by time and memory resource information for that submission, as described in section 9.

12. SYSTEM DESCRIPTION

To facilitate information exchange and understanding of the systems developed for the OpenASR21 Challenge, teams are required to submit a system description of at least five pages describing the designs and methods as well as any data harvested, pretrained models, and other external resources, and how they were used. A system description template detailing the format and expected content for the system description is available on NIST's OpenASR website.[19] The submitted system descriptions will be made public on NIST's OpenASR website after the evaluation period.

13. LEADERBOARD

NIST will display leaderboards of scores for submissions on the Dev datasets and Eval datasets, with different level of detail within teams (local) and across teams (global).

The Dev datasets leaderboards will be updated and displayed continuously as submissions are made and scored. The Eval datasets leaderboards will only be displayed after the evaluation period ends.

[18] The 7-day evaluation period (which covers parts of two calendar weeks) counts as one week for the purpose of submission limits.

[19] https://www.nist.gov/itl/iad/mig/openasr-challenge

Local leaderboards will show each team's graphs and tables of scores by language and training condition for all of that team's submissions only. Global leaderboards will show all teams tables of each team's highest score per language and training condition.

The leaderboard is not publicly viewable. After the evaluation, high-level results will be published on NIST's OpenASR web page.

14. RULES AND RESTRICTIONS FOR PUBLICATION OF RESULTS

- Participants are free to publish results for their own system but must not publicly compare their results with other participants (ranking, score differences, etc.) without explicit written consent from the other participants.
- While participants may report their own results, participants may not make advertising claims about their standing in the evaluation (regardless of rank), their winning the evaluation, nor claim NIST endorsement of their system(s). The following language in the U.S. Code of Federal Regulations (15 C.F.R. § 200.113)[20] shall be respected: NIST does not approve, recommend, or endorse any proprietary product or proprietary material. No reference shall be made to NIST, or to reports or results furnished by NIST in any advertising or sales promotion which would indicate or imply that NIST approves, recommends, or endorses any proprietary product or proprietary material, or which has as its purpose an intent to cause directly or indirectly the advertised product to be used or purchased because of NIST test reports or results.
- At the conclusion of the evaluation, NIST may generate a report summarizing the system results for conditions of interest. Participants may publish or otherwise disseminate these charts unaltered and with appropriate reference to their source.
- Any such report that NIST creates should not be construed or represented as endorsements for any participant's system or commercial product, or as official findings on the part of NIST or the U.S. Government.

[20] http://www.ecfr.gov/cgi-bin/ECFR

15. SCHEDULE

Milestone	**Date**
Evaluation plan release	July 2021
Registration period	**August 9 – October 15, 2021**
Development period	**August 9 – November 2, 2021 (potentially longer but excluding evaluation period)**
● Build and Dev datasets release	August 9, 2021
● Scoring server accepts submissions for Dev datasets	August 30 – November 2, 2021 (potentially longer but excluding evaluation period)
Registration closes	October 15, 2021
Evaluation period	**November 3 – 10, 2021**
● Release of Eval datasets	November 3, 2021
● Scoring server accepts submissions	November 4 – 10, 2021
● System output due at NIST	November 10, 2021
System description due at NIST	November 19, 2021

Table 10: Schedule