Change Log

Version	Date	Changes
1.4	2020-09-24	• Updated section 4 to further specify rules for the two training conditions • Updated section 6 to specify amount of evaluation data • Updated section 6.1 with updated audio data specifications
1.3	2020-08-19	(no major changes from initial public release up to this point)

OpenASR20 Challenge
(Open Automatic Speech Recognition 2020 Challenge)
Evaluation Plan

Contents

1. INTRODUCTION

The goal of the 2020 OpenASR20 (Open Automatic Speech Recognition 2020) Challenge is to assess the state of the art of ASR technologies under low-resource language constraints.

The OpenASR20 Challenge is the second open challenge created out of the Intelligence Advanced Research Projects Activity (IARPA) Machine Translation for English Retrieval of Information in Any Language (MATERIAL)[1] program that encompasses additional tasks, including cross-language information retrieval, domain classification, and summarization, and more languages. For every year of MATERIAL, the National Institute of Standards and Technology (NIST) organizes a simplified, smaller scale evaluation open to anyone wishing to participate, focusing on a particular technology aspect of MATERIAL. In 2019, CLIR technologies were the focus of the open challenge[2]; in 2020, it is ASR. 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 OpenASR20 Challenge is being implemented as a track of NIST's OpenSAT (Open Speech Analytic Technologies) evaluation series,[3] using the OpenSAT infrastructure for registration, data access, submission, and scoring purposes.

This evaluation plan as well as any additional documentation and tools are available via NIST's OpenASR main website.[4]

2. CHALLENGE TASK

The OpenASR20 Challenge task consists of performing ASR on audio datasets in up to ten different low-resource languages, producing the recognized written text. Participating teams may choose to attempt as many of the offered languages as they wish.

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.[5] 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:

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

WER for a dataset will be computed the total number of errors over the total number of reference words in the dataset.

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

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 ASR systems. The requirements for reporting time and memory resources are specified in section 8.

4. TRAINING CONDITIONS

The OpenASR20 Challenge offers two different training conditions, namely Constrained and Unconstrained. For any language processed, teams must make a submission for the Constrained Training condition. The Unconstrained Training condition is optional but encouraged.

In the *Constrained Training* condition, the only speech data permissible for training is a 10-hour subset of the Build dataset provided for the language being processed, clearly marked for that purpose. Additional text data, either from the provided Build dataset or publicly available resources, is permissible for training in the Constrained Training condition. Any such additional text training data must be specified in sufficient detail in the system description. Teams may not hire native speakers for data acquisition, system development, or analysis.

In the *Unconstrained Training* condition, teams may use speech data outside of the 10-hour subset marked for the Constrained Training condition for the language being processed, as well additional publicly available speech and text training data from any language. Any such additional training data must be specified in the system description. Teams may not hire native speakers for data acquisition, system development, or analysis.

Table 1: Permissible additional data resources by training condition contrasts types of additional data permissible for the two training conditions:

Data resource beyond designated Constrained Training data provided for the language being processed	**Constrained Training condition**	**Unconstrained Training condition**
Speech data in any language	No	Yes
Text data in any language	Yes	Yes
Pretrained models trained on speech data in any language	No	Yes
Pretrained models trained on text data in any language	Yes	Yes
Text-to-speech (TTS) output from TTS trained on the designated Constrained Training data	Yes	Yes
Non-speech acoustic data (noise etc.)	Yes	Yes

Table 1: Permissible additional data resources by training condition

5. LANGUAGES

The OpenASR20 Challenge is offered for the following ten low-resource languages:

- Amharic
- Cantonese
- Guarani
- Javanese
- Kurmanji Kurdish
- Mongolian
- Pashto
- Somali
- Tamil
- Vietnamese

6. DATA RESOURCES

Datasets for system training, development, and evaluation will be made available under an OpenASR20 Challenge participation and data license agreement for each of the languages. The datasets for most of the languages stem from the IARPA Babel program[6]; the Somali datasets stem from the IARPA MATERIAL program[7] .

Table 2 shows approximate dataset sizes per language:

Modality	**Build (training), Constrained**	**Build (training), Unconstrained**	**Dev**	**Eval**
Audio	10h	unlimited	10h	5h
Text	unlimited	unlimited	n/a	n/a

Table 2: Datasets

6.1 AUDIO DATA SPECIFICATIONS

The audio data provided consists of conversational telephone speech, in separate channels for each speaker.

The data were sampled at 8kHz, 44.1kHz, or 48kHz (specified in the header of each file) and are provided in .sph or .wav format, depending on the language. More details regarding the audio data can be found in section 3 of the IARPA Babel Data Specifications for Performers.[8]

6.2 DATA USAGE RULES AND RESTRICTIONS

Participants are required to sign and abide by a participation and data license agreement. No data or annotations received under this agreement may be distributed outside of purposes directly related to the OpenASR20 Challenge.

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

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

[8] https://www.nist.gov/document/iarpababelspecification-02062013pdf

The rules governing the use of the different dataset are outlined in Table 3:

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	Yes
Train models using released data	Yes	No	No
Parameter tuning	Yes	Yes	No
Score locally	Yes	Yes	No

Table 3: Dataset rules

- Teams may use the Build datasets for system training as well as examine it in detail.
- 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 parameters. Teams may not use the Dev datasets for system training.
- In the Constrained Training condition, no additional acoustic training data beyond what is provided under the OpenASR20 Challenge data license agreement may be used.
- In the Unconstrained Training condition, teams may mine the web for additional publicly available training data. Any such data harvested for training must be specified in the system description. Teams may not hire 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.
- Teams may not use third-party ASR commercial software in any part of their pipeline.

7. 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.

7.1 REFERENCE FILE FORMAT (STM)

The reference files on the scoring server are named as follows:

`<DocID>_<ChannelID>.stm`

For example:

`BABEL_BP_101_98675_20111117_190458_inLine.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 [1 | 2], the speaker's 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>` The waveform filename. NOTE: no pathnames or extensions are expected.
`<C>` The waveform channel(numeric). Either 1 or 2. "inLine" is 1 and "outLine" is 2.
`<S>` The speaker id, no restrictions apply to this name.
`<BT>` The begin time (seconds) of the segment.
`<ET>` The end time (seconds) of the segment.
`transcript` The transcript can take on 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 which chops the hypothesis file to 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.

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
```

7.2 SYSTEM OUTPUT FORMAT (CTM)

The system output files on the scoring server are named as follows:

`<DocID>_<ChannelID>.ctm`

For example:

`BABEL_BP_101_98675_20111117_190458_inLine.ctm`

The basename of these files must match the name of the corresponding reference files.

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 [1 | 2], 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> is the waveform base filename. NOTE: no pathnames or extensions are expected.
<C> is the waveform channel. Either 1 or 2. “inLine” is 1 and “outLine” is 2.
<BT> is the begin time (seconds) of the token, measured from the start time of the file.
<DUR> is the duration (seconds) of the token.
<CONF> is an 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
```

7.3 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 features (e.g., cough, mispronunciation, etc.). Prior to scoring, normalization is performed on the speech features as outlined in Table 4:

Babel Tag/Marker	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 the segment from scoring	<overlap>	IGNORE_TIME_SEGMENT_IN _SCORING
~	delete tag	~ contemplation	contemplation
(()) <sta> <int> <lipsmack> <breath> <cough> <laugh> <click> <ring> <dtmf> <male-to-female> <male-to-female>	delete tag	<lipsmack>	
<prompt>	exclude the segment from scoring	<prompt>	IGNORE_TIME_SEGMENT_IN _SCORING
_	change to space	N_I_S_T	N I S T
-	optionally deletable	I communica- to him	I (communica-) to him
/ /	delete tag	/B/	B

Table 4: Normalization

A description of the Babel transcription conventions can be found in section 5 of the IARPA Babel Data Specifications for Performers.[9]

A Python script converting from the transcript to STM format as well as performing normalization is available on NIST's OpenASR main website.[10]

[9] https://www.nist.gov/document/iarpababelspecification-02062013pdf
[10] https://www.nist.gov/itl/iad/mig/openasr-challenge

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

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

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>
```

8. 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.

Teams must report:

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 and 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.

8.1 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 CPU/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.

8.1.1 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.

8.1.2 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.

8.1.3 TIME FOR GPUS

The use of GPUs can obfuscate the total CPU time required. GPUs contain 1000s 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 100s 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.

8.2 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.

8.2.1 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.

8.3 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
```

9. REGISTRATION, DATA ACCESS, SUBMISSIONS, AND SCORING

To participate in the OpenASR20 Challenge, participants must complete the registration process via NIST's OpenSAT web server[11] 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.

9.1 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 4, 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. Submission quotas and feedback availability are specified in Table 5.

Timeline	**Datasets**	**Limit per week[12] and training condition**	**Feedback (score)**
Development period	Dev	unlimited	yes
Evaluation period	Eval	5	yes, only overall score

Table 5: Submission quotas and feedback

[11] https://sat.nist.gov

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

9.2 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 7.2.

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 8.

10. SYSTEM DESCRIPTION

To facilitate information exchange and understanding of the systems developed for the OpenASR20 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 and how it was used. A system description template detailing the format and expected content for the system description is available on NIST's OpenASR main website.[13]

11. 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.

Local leaderboards will show each team 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.

12. PUBLICATION OF RESULTS RULES AND RESTRICTIONS

- 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, or winning the evaluation, or claim NIST endorsement of their system(s). The following language in the U.S. Code of Federal Regulations (15 C.F.R. § 200.113)[14] 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

[13] https://www.nist.gov/itl/iad/mig/openasr-challenge
[14] http://www.ecfr.gov/cgi-bin/ECFR

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.

13. SCHEDULE (TENTATIVE)

Milestone	**Date**
Evaluation plan release	July 2020
Registration period	**August 7 – October 15, 2020**
Development period	**August 7 – November 2020 (potentially longer)**
● Build and Dev datasets release	August 7, 2020
● Scoring server accepts submissions for Dev datasets	August 26 – November 2020 (potentially longer)
Registration closes	October 15, 2020
Evaluation period	**November 3 – 10, 2020**
● Release of Eval datasets	November 3, 2020
● Scoring server accepts submissions	November 4 – 10, 2020
● System output due at NIST	November 10, 2020
System description due at NIST	November 23, 2020

Table 6: Schedule