DRAFT KWS14 KEYWORD SEARCH EVALUATION PLAN

Table of Contents

1 INTRODUCTION

This document contains the evaluation plan for the 2014 Keyword Search (KWS) evaluations that includes both the Babel Program Development Language and the OpenKWS14 evaluations. The languages used for the Babel Development Language evaluations are specified in Appendix E while the languages for the OpenKWS evaluation, will include only previous OpenKWS test language data and a new "surprise" language for which the identity is revealed at the time of release of the data.

The evaluation plan covers the data resources, KWS task definitions, the Speech-To-Text task definition, file formats both for system inputs and outputs, evaluation metrics, scoring procedures, and the results submission protocols.

2 DATA RESOURCES

Language resources will be provided to developers according to the Babel Data Specification Document and the Language Specific Peculiarities (LSP) documents (one per language). Data will be released in a build pack, an evaluation pack, and an IndusDB containing scoring files.

The "build pack" contains audio, transcripts, and lexica that can be used for system training and tuning. The development test data must be used only for parameter tuning and should not be incorporated into training material during the preparation of evaluation system models[1]. Note: the build pack lexicon contains entries for both the training and development test data. The lexical items that exist only in the development test data must be excluded during model training.

The "evaluation pack" contains only audio data for the evaluation.

The "IndusDB" releases are cumulative *tar* archives that contain (1) reference transcripts for the development test material and (2) keyword files and experiment control files for both the development and evaluation test material for the languages available to the site (i.e., OpenKWS participants will receive only surprise language materials).

3 THE KEYWORD SEARCH TASK

The KWS task is to find all of the occurrences of a "keyword", a sequence of one or more words in a corpus of un-segmented speech data, transcribed in a language's original orthography.

[1] While an operational system would likely incorporate development test data to maximize performance, existing evaluation participants have agreed not to do so in order to focus on other, language-oriented techniques to improve performance rather than simply adding data.

3.1 Evaluation Conditions, and Evaluation Condition Keywords

The goal of the KWS evaluation is twofold: to build KWS systems given a limited amount of time and data resources, and to understand the differences in system performance given common constraints.

To facilitate cross-site system comparisons without unduly constraining creativity, three overarching evaluation conditions were defined in 2013. They are defined in the following subsections.

In order to support greater insights into KWS systems, a mechanism has been developed for participants to expand upon the 2013 evaluation conditions. Appendix I "Experimental Conditions for KWS Evaluations" provides a description of how to create novel dataset definitions and use them along with standard dataset definitions to describe what data is used to build a system and how it is used. This information will be encoded in a **<DATADEF>** string that will be included in the system description that accompanies evaluation submissions, as specified in Appendix G. DATADEF strings are the mechanism NIST will use to facilitate analysis of results across sites.

3.1.1 Build Pack Utilization

There are two predefined usage conditions for the build pack of the test language that differ in the amounts of transcribed training material used:

- The full language pack (**FullLP**) – all data resources provided by the program are used for development.
- The limited language pack (**LimitedLP**) – a 10-hour subset of the FullLP training set is used (transcriptions, metadata, and lexicon) and all the rest of the training audio data of the FullLP (without transcriptions or lexicon).

3.1.2 Language Resources Utilization

The use of additional language resources (LRs) will likely impact system performance. In order to differentiate the amount of additional resources used, three levels are defined:

- Baseline LRs (**BaseLR**): The used LR is constrained to only the test language's, project-supplied build pack[2].
- Non-test language Babel LRs (**BabelLR**): Additional LRs consist of any/all Babel-supplied language packs. (Note: OpenKWS participants can use the Babel Vietnamese data set for this condition; whereas, Babel performers use can all Babel data sets. The experimental condition notation specified in Appendix I will differentiate the amount/number of resources used by the system.)
- Other LRs (**OtherLR**): Additional LRs consist of team-, community-, or pre-existing-LRs. The LRs used must be described using the evaluation keywords as described in Appendix G.

3.1.3 Test Audio Re-Use

Processing of the audio after knowledge of the keywords will likely impact performance. In order to differentiate systems that do not reprocess the audio from those that do, two levels are defined:

- No test audio re-use (**NTAR**): the system does not re-process the test audio after keywords are provided[3].
- Test audio re-use (**TAR**): the system re-processes the audio with knowledge of the search keywords.

3.1.4 Required Systems

Appendix E enumerates the required evaluation conditions by test language. Appendix I defines the notation used to define new optional evaluation conditions that must be included in the system descriptions.

3.1.5 Primary vs. Contrast Systems

For a given evaluation condition, as specified by a unique DATADEF string from Appendix I, a site may choose to submit multiple runs. If they do, the sites must designate one of them as their "primary" run and the rest (if applicable) as "contrastive" runs. The primary run should be the run on which the site expects to perform best for the given condition prior to receiving performance assessments. The designation of primary vs. contrastive will be specified in the Experiment IDs as specified in Appendix B. NIST will check to make sure primary runs exist for all team-submitted DATADEF strings and re-name them as appropriate.

3.2 Keywords

Keywords, a sequence of contiguous lexical items, will be specified in the language's UTF-8 encoded, native orthographic representation as typified in the provided language resources. Example keywords are "grasshopper", "New York", "overly protective", "Albert Einstein", and "Giacomo Puccini". All transcribed words as specified by the Babel Data Specification Document are potential keywords.

The existence of a spoken keyword will be judged solely on the orthographic transcription of the speech. Therefore:

- Homographs will be not be differentiated.

[2] BaseLR is a "flat start" condition where models may only be initialized with build pack data. Participants are encouraged to contact NIST if they are uncertain if a technique meets this stringent guideline or if they are unable to build a BaseLR system.

[3] In practice, developers should follow the spirit of the rule. Re-processing of spectral features derived from the audio would be a TAR system even though the "audio" is not reprocessed.

- Morphological variations of a keyword will not be considered positive matches.

When possible, transcript comparisons will be case insensitive (e.g., /bill/ and /Bill/ are not differentiated).

Reference occurrences will be found automatically by searching the reference transcript supplied in an RTTM file[4]. Reference keywords will be identified according to the following rules:

1. A keyword is a contiguous sequence of RTTM LEXEME records of all LEXEME subtypes including hesitations, filled pauses, fragments, and truncations.
2. Every word in a keyword must be from the same file and channel.
3. Non-lexical tokens, lip-smack, click, dtmf, etc., are ignored during the search for keywords.
4. The silence gap between adjacent words in a keyword must be ≤ 0.5 second.

3.3 Transcription Normalization

Each language will require differing text normalization strategies to accommodate language specific transcription practices. Appendix E describes the text normalization steps for each language.

3.4 Non-Scored Regions

Segments where forced alignment of the audio to the reference transcript fails are not scored which means that all system-generated output during the specified segment will be ignored during scoring.

3.5 System Output

For each keyword supplied to the system, a KWS system will report the following information for each putative occurrence.

- the begin and end time of the keyword occurrence
- a score indicating how likely the keyword exists with more positive values indicating more likely occurrences, and
- a hard (YES/NO) decision as to whether the detection is correct.

A system output will be considered correct if a reference keyword appears in the transcript at a time corresponding to the system generated time as described in Section 5.

The score for each keyword occurrence can be of any scale (NIST recommends a log likelihood[5]). However, since the scores will be used to derive keyword-weighted Detection Error Tradeoff (DET) curves, scores across keywords should be commensurate to ensure minimum DET curves; otherwise, a non-optimal DET curve will be generated.

[4] See Appendix C for a definition of RTTM files.

[5] The log likelihood, with base e, is suggested, so that the system may be evaluated in a variety of application scenarios that exhibit different prior probabilities.

Developers are encouraged to over-generate putative keyword occurrences beyond the system's hard decision boundary so that DET curves cover a wider range of operating points.

3.6 Scoring Command

The command to score a KWS system is as follows. Appendix A defines the file types used in this command. This invocation is the default usage. Language specific options are identified in Appendix E.

```
% KWSEval -e <ECF> -r  <RTTM> -t <KWLIST> \
         -s <KWSLIST>  -c -o -b -d  \
         -f <RESULTROOT>
```

4 THE SPEECH TO TEXT TASK

The Speech-To-Text (STT) task is to produce a verbatim, case insensitive transcript of uttered lexical items. Systems will output a stream of Conversation Time Marked (CTM) lexical tokens reporting the token's begin and end time within the recording, a confidence score value [0,1] indicating the system's confidence that the token is correct, and lexical sub-type information.

The STT task is a diagnostic task used to quantify the performance of underlying, word-based, STT in the context of performing KWS. Therefore, systems are not expected to be optimized for STT error rates but rather for KWS.

4.1 STT Evaluation Conditions

STT systems will be differentiated using the same KWS evaluation conditions.

The "primary" vs. "contrastive" definitions for the KWS task apply to the STT task.

4.2 Lexical Tokenization

Lexical tokenization must follow the standard used in the language pack.

4.3 Lexical Token Scoring

The following rules define three types of tokens: Scored tokens (tokens that must be recognized), optionally deletable tokens (tokens that may be omitted by the STT system without penalty), and non-scored tokens (tokens removed from both the reference and STT transcripts prior to scoring).

- Scored tokens
  - All words transcribed as specified by the Babel Data Specification Document.
- Optionally deletable tokens
  - Fragments (marked with a -) in the reference transcript. System tokens with token-initial text matching the fragment's text will be scored as correct (e.g. /theory/ would be correct for fragment /th-/). The same test is applied to the obverse, token-final fragments /-tter/ matching /latter/.
  - The hesitation tags (<hes>).
- Non-scored tokens
  - Codeswitch tags.

- Speaker change tags.
- Unintelligible speech tags.
- Non-lexical punctuation.
- Non-lexical, speaker-produced sounds (<lipsmack>, <cough>, <breath>, etc. as defined in the data specification document).

4.4 Non-scored Speech Segments

Segments containing the <overlap>, unintelligible [(()) tags], and <prompt> tags will not be scored. In addition, segments containing transcript tokens that were not forced aligned in the reference will not be scored.

4.5 Scoring Procedures

The NIST SCTK toolkit will be used to evaluate the performance of STT systems. System-generated STT output must be in CTM format. (See Appendix C) The following command will score a CTM-formatted file with sclite using a segment time marked (STM) formatted file supplied with the reference. This invocation is the default usage. Language specific options are identified in Appendix E.

```
% sclite –h file.ctm ctm –t file.stm stm -o sum rsum pra
-D -F -e utf-8
```

4.6 Required and Optional STT Evaluation Conditions

Babel performers are required to submit STT system outputs used for all primary KWS system classes while OpenKWS participants (who use word-based models) are encouraged to submit STT system output(s) for their KWS submissions.

5 KEYWORD SEARCH EVALUATION

Keyword detection performance will be measured as a function of Missed Detection and False Alarm error types.

The official scoring protocol will be the "Keyword Occurrence Scoring" protocol that evaluates system accuracy based on a temporal alignment of the reference keywords to system-hypothesized keywords.

The Keyword Occurrence Scoring protocol includes three steps to evaluate system performance: (1) reference-to-system keyword alignment, (2) performance metric computation, and (3) DET curves. In addition, the NIST scoring tool supports diagnostic measure computations defined in Appendix F.

5.1 Reference-to-System Keyword Occurrence Alignment

KWS systems detect keyword occurrences in un-segmented audio. In order to evaluate the performance of the system, the first step is to find the minimal cost, 1:1 alignment (or mapping) between the known locations of the reference occurrences for a given keyword and the putative system occurrences for a given keyword. The KWS evaluation uses the Hungarian Solution to the Bipartite Graph matching problem[6] to compute the 1:1 mapping using the kernel function *K()* that numerically compares the mapping of system and reference occurrences, as well as the missed detections and false alarms.

The kernel function first determines if the ref/sys occurrences are mappable by requiring the sys occurrence to be within a temporal tolerance collar (Δ_T) of the reference occurrence. Specifically, the midpoint of the system occurrence must be within the interval from Δ_T before the beginning to Δ_T after the end of the reference occurrence as determined by forced alignment of the reference transcript to the audio. If the occurrences are mappable, the comparison of a ref/sys pair is 1 plus a weighted sum of the occurrences' temporal overlap and the percentile of the system occurrence's detection score. The formulas are as follows.

$K(O_{r,i}) = 0$; if ref occurrence *i* is not mapped (i.e., a miss)

$K(O_{s,j}) = -1$; if sys occurrence *j* is not mapped (i.e., a false alarm)

$$K(O_{r,i},O_{s,j}) = \begin{cases} UnMapped; \begin{cases} \text{if } Mid(O_{s,j}) > En(O_{r,j}) + \Delta_T \text{ or} \\ \text{if } Mid(O_{s,j}) < Be(O_{r,j}) - \Delta_T \end{cases} \\ 1 + \varepsilon_{tm} * TmCgr(O_{r,i},O_{s,j}) + \varepsilon_{scr} * ScrCgr(O_{r,i},O_{s,j}) \end{cases}$$

Where:

$O_{r,i}$ = the reference occurrence i of the keyword

$O_{s,j}$ = the system occurrence j of the keyword

$Mid()$= the midpoint of an occurrence

$En()$= the ending time of an occurrence

$Be()$= the beginning time of an occcurence

$$TmCgr() = \frac{Min(En(O_{s,j}),En(0_{r,i})) - Max(Be(O_{s,j}),Be(O_{r,i}))}{Max(0.00001,En(0_{r,i}) - Be(0_{r,i}))}$$

$$ScrCgr() = \frac{Scr(O_{s,j}) - LowestScr(Sys_s)}{Max(0.0001,LargestScr(Sys_s) - LowestScr(Sys_s))}$$

Δ_T = The temporal tolerance collar ; 0.5 second

ε_{tm} = The weight for time congruence ; 1e-08

ε_{scr} = The weight for decision scores ; 1e-06

$LowestScr(Sys_s)$ = The smallest detection score of the keyword of System s

$LargestScr(Sys_s)$ = The largest detection score of the keyword for System s

Including ScrCgr() ensures that if there are two system occurrences that are both permissible matches to only one known occurrence (and *vice versa*), then the mapping will remain 1:1 while minimizing the error rates.

[6] Harold W. Kuhn, "The Hungarian Method for the assignment problem", *Naval Research Logistic Quarterly*, **2**:83-97, 1955.

5.2 Keyword Occurrence Performance Metric Computation

Overall system detection performance will be measured in terms of an application model by assigning a value to each correct output and a cost (i.e., negative value) to each incorrect output via the term-weighted value function[7].

Term-weighted value (***TWV***) is 1 minus the weighted sum of the term-weighted probability of missed detection ($P_{Miss}(\theta)$) and the term-weighted probability of false alarms ($P_{FA}(\theta)$).

$$TWV\ (\theta) = 1 - \left[\mathrm{P}_{Miss}(\theta) + \beta \cdot \mathrm{P}_{\mathrm{FA}}(\theta) \right]$$

where:

θ = The criteria used to determine if the system-detected *kw* is scored. Various methods will be used.

$$P_{Miss}(\theta) = \sum_{kw=1}^{K} \left[N_{Miss}(kw,\theta) / N_{True}(kw) \right] \Big/ K$$

$$P_{FA}(\theta) = \sum_{kw=1}^{K} \left[N_{FA}(kw,\theta) / (N_{NT}(kw)) \right] \Big/ K$$

K = # of keywords with 1 or more reference occurrences

$$\beta = \frac{C}{V} \cdot \left(\mathrm{Pr}_{kw}^{-1} - 1 \right)$$

C = The cost of an incorrect detection; defined as 0.1

V = The value of a correct detection; defined as 1

Pr_{kw} = The prior probability of a keyword; defined as 10^{-4}

$N_{Miss}(kw,\theta)$ = # of missed detection of keyword *kw* for θ

$N_{FA}(kw,\theta)$ = # of false alarms of keyword *kw* for θ

$N_{True}(kw)$ = # of reference occurrences of keyword *kw*

$N_{NT}(kw)$ = # of non-target trials for keyword *kw*

Since there is no discrete specification of "trials" in un-segmented audio, the number of Non-Target trials for a term, $N_{NT}(term)$, will be defined somewhat arbitrarily to be proportional to the number of seconds of speech in the data under test. Specifically:

$$N_{NT}\left(kw\right) = n_{tps} * T_{speech} - N_{true}(kw)$$

where:

n_{tps} is the number of trials per second of speech (n_{tps} will be set arbitrarily to 1), and

T_{speech} is the total amount of evaluated speech in the test data. Non-evaluated audio does not included in T_{speech}. The unit is seconds. The following domain specific rules apply to calculating T_{speech}:

- For Babel's split-channel conversational telephone speech (splitcts), 0.5 the duration of each channel is used.

The TWV of a perfect system is 1. A system that outputs nothing is 0. TWV can go to -∞ since false alarm errors are included in the measure.

5.2.1 Primary Measure of Performance: Actual TWV

The primary measure of performance for KWS is Actual TWV (ATWV). ATWV is a measure of system performance in the context of a hypothetical application that requires a system to (1) harmonize the domain and range of the detection scores across keywords and (2) select a detection threshold that maximizes ATWV for a given β. Setting β *a-priori* ensures developers optimize system performance to the same P_{Miss}/P_{FA} tradeoff space. Actual TWV corresponds to the TWV using the system's 'YES' hard decision threshold.

5.3 Detection Error Tradeoff Curves

The detection scores output by a system permit error analysis over a wide range of operating points by computing error rates for all detection score thresholds (i.e. θ = {detection scores}). The resulting Detection Error Tradeoff (DET) curve depicts the tradeoff between the probabilities of missed detections versus false alarms.

6 SPEECH-TO-TEXT EVALUATION

STT performance will be measured as a function of deletion, insertion and substitution error types. System evaluation will occur in three steps: (1) text normalization, (2) reference-to-system token alignment, and (3) performance metric computation. In addition, the NIST scoring tool supports diagnostic STT measure computations defined in Appendix F.

6.1 Token normalization

Text will be pre-filtered to appropriately handle the speech phenomena as described in Section 3.2.

6.2 Token Alignment

Scorable tokens, as defined in Section 3.2, are aligned using the Dynamic Programming solution to string alignments[8]. The weights used for substitutions, insertions, deletions, and correct recognition are 4, 3, 3, and 0 respectively.

6.3 STT Performance Metric Computation

An overall Word Error Rate (WER) will be computed as the fraction of token recognition errors per reference token:

$$WER = \left(\mathrm{N}_{\mathrm{Del}} + \mathrm{N}_{\mathrm{Ins}} + \mathrm{N}_{\mathrm{Subst}} \right) \Big/ \mathrm{N}_{\mathrm{Ref}}$$

where

N_{Del} = the number of unmapped reference tokens,

[7] The TWV metric uses "Term" in its name. For the KWS evaluation, "keyword" and "term" mean the same thing.

[8] www.itl.nist.gov/iad/mig/publications/storage_paper/lrec06_v0_7.pdf

N_{Ins} = the number of unmapped STT output tokens,

N_{Subst} = the number of mapped STT output tokens with non-matching reference spelling, and

N_{Ref} = the maximum number of reference tokens[9]

7 EVALUATION RULES

The following rules apply to all evaluation conditions.

7.1.1 Keyword Interactions

Each keyword must be processed separately and independently during keyword detection. The system-generated detection outputs for a keyword (as derived from processing the test data audio) must not influence the detection of other keywords. The search results for each keyword are to be output prior to performing detection on the next keyword.

7.1.2 Human Interactions

No manual or human interaction with the test audio data is allowed. No pronunciations should ever be created by hand, but must be created automatically (e.g., using G2P). In general, human-in the loop on evaluations is not allowed. No listening to the audio, no mechanical Turk, etc. are allowed. When in doubt, contact NIST for guidance.

7.1.3 LSP Resources

The LSP contains a full inventory of phones for the language. Evaluation participants are allowed to leverage that information. (There is no guarantee that all borrowings are covered in the LSP.)

8 PUBLICATION OF RESULTS

Results will first be reported to the KWS participant community for vetting. Then, once NIST and the community review the results, NIST will declare the scores final, and the vetted results will then be ready for dissemination outside the KWS community.

Publication of vetted results is encouraged and should be in accordance with the evaluation agreement, the data license, and (for Babel performers) individual Babel contracts.

9 EVALUATION DATA: SYSTEM INPUTS AND OUTPUTS

Appendix A provides an overview of evaluation input data files provided by NIST for running a system evaluation, as well as the expected format of a system output to for scoring.

System inputs include the test data and keywords required for the evaluation as specified in Section A.1 and A.2 of Appendix A. System outputs of keyword detections will be stored in XML-formatted text files as specified in Section A.3 of Appendix A.

10 SUBMISSION OF RESULTS

Submissions will be made via the Babel Scoring server as specified in Appendix B which explains the submission protocol. Submissions must include a system description that documents the algorithms, computing architecture, runtimes, and data resources. Appendix G gives extensive guidance for preparing a system description.

11 SCHEDULE

Consult the evaluation schedule on the Babel Sharepoint site and/or the OpenKWS web site (http://www.nist.gov/itl/iad/mig/openkws.cfm) for the current year.

[9] N_{Ref} includes all scorable reference tokens (including optionally deletable tokens) and counts the maximum number of tokens. Note that N_{Ref} considers only the reference transcript and is not affected by scorable tokens in the system output transcript, regardless of their type.

Appendix A: KWS Test Input and Output File Formats

Figure 1 shows the system input/output files and how they relate to system operation and evaluation. This appendix documents the file formats for each system input and output.

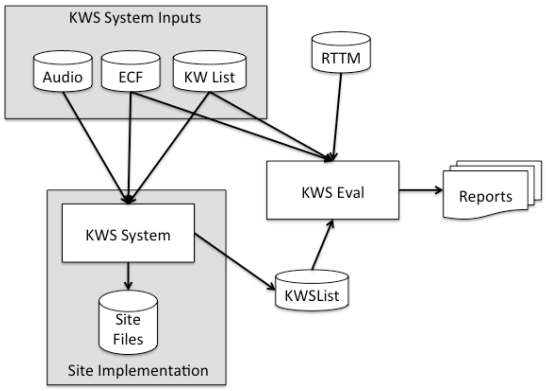

Figure 1: System and evaluation inputs and outputs

The three input files to a site-implemented KWS system are as follows[10].

- Audio files: include SPHERE and WAV formatted waveform files in an evaluation pack.
- Experiment Control File (ECF): defines the excerpts within audio files to be used for a specific evaluation and the language/source type of each file.
- KWList: defines the keywords to search for in the indexed corpus.

Once the site's indexer and searcher completes processing the data, a KWSList file (Appendix A.4) is generated and used by the evaluation code along with the reference Rich Transcription Time Marked (RTTM) (Appendix C) file to produce the performance analysis reports.

The remainder of this appendix defines the input and output file formats.

A.1 Evaluation Input: Experiment Control Files

NIST-supplied ECFs are the mechanism the evaluation infrastructure uses to specify time regions within an audio recording, the language, and the source type specified for the experimental condition. A *system input ECF* file will be provided for all tasks to indicate what audio data is to be indexed and searched by the system. The evaluation code also uses an ECF file to determine the range of data to evaluate the system on. In the event a problem is discovered with the data, a special *scoring ECF* file will be used to specify the time regions to be scored.

The ECF file is an XML-formatted text file.

ECF File Naming

ECF files end with the '.ecf.xml' extension.

ECF File Format Description

An ECF consists of two hierarchically organized XML nodes: "ecf", and "excerpt". The XML scheme for an ECF file can be found in the F4DE software package. The following is a conceptual description of an ECF file.

The "ecf" node contains a list of "excerpt" nodes. The "ecf" node has the following attributes:

- source_signal_duration: a floating point number indicating the total duration in seconds of recorded speech
- version: A version identifier for the ECF file
- language: language of the original source material.

Each "excerpt" tag is a non-spanning node that specifies the excerpt from a recording that is part of the evaluation. The "excerpt" has the following attributes:

- audio_filename: The attribute indicates the file id, excluding the path and extension of the waveform to be processed.
- source_type: The source type of the recording either "bnews", "cts", "splitcts", or "confmtg".
- channel: The channel in the waveform to be processed.
- tbeg: The beginning time of the segment to processes. The time is measured in seconds from the beginning of the recording which is time 0.0.
- dur: The duration of the excerpt measured in seconds.

For example:

```
<ecf source_signal_duration="340.00"
     version="20060618_1400" language="english" >
<excerpt
     audio_filename="audio/dev04s/english/confmtg/NIS
     T_20020214-1148" channel="1" tbeg="0.0"
     dur="291.34" source_type="confmtg"/>
<excerpt
     audio_filename="audio/eval03/english/bnews/ABC_
     WNN_20020214_1148.sph" channel="1"
     tbeg="0.0" dur="291.34" source_type="bnews"/>
…
</ecf>
```

[10] The diagram is a stylized representation of site implemented system operations and developers are free to organize their systems at their discretion.

A.2 Evaluation Input: KWList Files

The NIST-supplied Keyword List file is an XML-formatted text file that defines the search keywords to be processed by a KWS system. Each keyword is identified by KWID which is used to track keywords through the evaluation process and specify keyword texts with a flexible set of attributes.

Keyword List File Naming

Keyword List files end with a .kwlist.xml extension.

Keyword List File Format Description

The Keyword List file consists of three hierarchically organized XML nodes: "kwlist", "kw", and potentially several nodes under "kw". The XML scheme for a KWList file can be found in the F4DE software package. The following is a conceptual description of a KWList file.

The "kwlist" node contains a list of "keyword" nodes. The "kwlist" has the following attributes:

- ecf_filename: The basename[11] of the ECF file associated with this Kwlist file.
- version: A version identifier for the file.
- language: Language of the original source material.
- encoding: The character encoding of the text data. Only "UTF-8" is currently accepted.
- compareNormalize: The function used to normalize the text before comparison. Current legal values are blank (which applies no normalization) and "lowercase".

Each "kw" node is a spanning XML tag that contains a set of additional XML nodes to specify the keyword. There is a single attribute 'kwid'.

- kwid: A string identifying the keyword.

The "kw" tag contains two sub-nodes "kwtext" (which is the keyword text) and the "kwinfo" tag (which contains a flexible attribute/value structure).

The "kwtext" tag is a spanning tag than contains the CDATA (character) string for the keyword. The leading and trailing white space of the keyword string is NOT considered part of the keyword while single internal white space(s) are.

The "kwinfo" tag is a spanning tag that contains one or more "attr" tags that specify an attribute name and value with a "name" and "value" tag respectively. Both contents of "name" and "value" tags are CDATA.

The following is an example KWlist file:

```
<kwlist ecf_filename="english_1" version ="20060511-
      0900"
      language="english" encoding="UTF-8"
      compareNormalize="lowercase">
<kw kwid="dev06-0001">
      <kwtext>find</kwtext>
      <kwinfo>
        <attr>
          <name>NGram Order</name>
          <value>1-grams</value>
        </attr>
      </kwinfo>
 </kw>
<kw kwid="dev06-0002">
      <kwtext>many items</kwtext></kw>
       <kwinfo>
        <attr>
          <name>NGram Order</name>
          <value>2-grams</value>
        </attr>
       </kwinfo>
      <./kw>
</kwlist>
```

A.3 Evaluation Output: KWSList Files

The system-produced KWSList file is an XML-formatted file produced by a KWS system. It contains all the runtime information as well as the search output generated by the system.

KWSList File Naming

Since KWSLists are produced by a KWS system, they apply to a particular ECF and KWlist. KWSList file are named with the .kwslist.xml extension.

KWSList File Format Description

A KWSList file is an XML file with three hierarchically organized XML nodes: "kwslist", "detected_kwlist", and "kw". The "kwslist" records the system inputs and parameters used to generate the results. The "detected_kwlist" is a collection of "kw" nodes which are the putatively detected keywords. The XML scheme for a KWSList file can be found in the F4DE software package. The scheme is the authoritative source. Below is a content description of the XML nodes and attributes.

The "kwslist" node contains a set of "detected_kwlist" nodes: one for each search keyword. The "kwlist" node contains the five attributes:

- kwlist_filename: The name of the KWList file used to generate this system output.
- language: Language of the source material.
- system_id: A text field supplied by the participant to describe the system.

Each "detected_kwlist" node contains the system output for a single keyword. It consists of a set of "kw" nodes; each "kw" node specifying the location of single detected keyword. The three attributes of a "detected_kwlist" are:

- kwid: The keyword id from the KWList file.

[11] The basename of a file excludes the directory names and extensions. For example the basename of "the/directory/file.txt" is "file".

- search_time: (optional for backward compatibility) A floating point number indicating the number of CPU seconds spent searching the corpus for this particular keyword.
- oov_count: An integer reporting the number of tokens in the keyword that are Out-Of-Vocabulary (OOV) for the system and/or the training and development language data. If the system does not use a word dictionary, the value should be "NA".

Each "kw" node is a non-spanning XML node that contains the location and detection score for each detected keyword. The six attributes are:

- file: The basename of the audio file as specified in the ECF file.
- channel: the channel of the audio file were the keyword was found.
- tbeg: The beginning time of the keyword expressed in seconds with 0.0 being the beginning of the audio recording.
- dur: The duration of the keyword in seconds.
- score: The detection score indicating the likelihood of the detected keyword.
- decision: [YES | NO] The binary decision of whether or not the keyword should have been detected to make the optimal score.

The KW tag can include optional user data that is structured as an attribute value, non-spanning node "user_data" with that contains 2 required attributes:

- name: the name of the attribute as a string
- value: the value of the attribute as a string

An example KWSList file is:

```
<kwslist
      kwlist_filename="expt_06_std_eval06_mand_all_sp
      ch_expt_1_Dev06.tlist.xml"
      language="english"
      system_id="Phonetic subword lattice search">
<detected_kwlist kwid="dev06-0001"
            search_time="24.3" oov _count="0">
   <kw file="NIST_20020214-1148_d05_NONE"
            channel="1"      tbeg="6.956"      dur="0.53"
      score="4.115" decision="YES"/>
   <kw file="NIST_20020214-1148_d05_NONE"
            channel="1"      tbeg="45.5"       dur="0.3"
      score="4.65" decision="NO">
        <user_data name="rank" value="30" />
         <user_data  name="confidence_audio"  value="30"
      />
   </kw>
</detected_kwlist>
</kwslist>
```

Appendix B: Submission and Scoring of System Outputs

Babel will make extensive use of the NIST Indus scoring server. There are 4 steps to submit a system output for scoring: (1) evaluation condition specification via an Experiment Identifier, (2) system output formatting and naming, (3) system documentation via a system description, and (4) scoring locally or via the Indus scoring server.

The final section of this Appendix contains a high-level checklist to complete an evaluation.

B.1 Experiment Identifiers

The packaging and file naming conventions for system outputs rely on Experiment Identifiers (EXPID) to organize and identify the files for each evaluation condition and link the system inputs to system outputs. Since EXPIDs may be used in multiple contexts, some fields contain default values. The following section describes the EXPIDs.

The following Extended Backus-Nuar Form (EBNF) describes the EXPID structure.

EXPID :== KWS14_<TEAM>_<CORPUS>_<PARTITION>_<SCASE>_<TASK>_<PRIM>-< SYSID>_<VERSION>

Where:

<TEAM> :== your team name. Only alphanumerical characters are allowed, with no space(s).

<CORPUSID> :== The id of the corpus used as the source of the audio data. For the Cantonese B data set, the value is "babel101b-v0.4c".

<PARTITION> :== conv-dev | conv-eval

<SCASE> :== BaDev | BaEval (See Scoring Cases below for descriptions)

<TASK> :== KWS | STT

<PRIM>-<SYSID> :== a site-specified string (that does not contain underscores or spaces) intended to differentiate runs from the same team. <PRIM> must be either *p* indicating a primary system or c indicating a contrastive system for a given system. <SYSID> is a site-determined identifier for the system. For example, this string could be *p-baseline* or *c-contrast*. This field is intended to differentiate between runs for the same evaluation condition. Therefore, a different SYSID should be used for runs where any changes were made to a system. SY

<VERSION> :== 1..n (with values greater than 1 indicating resubmitted runs of the same experiment/system). This tag is meant as a simple version control to differentiate superseded submissions for reasons such as bug fixes. Teams desiring a higher level of versioning should encode that information in the SYSID field.

An example EXPID is:

KWS14_Seeker_babel101b-v0.4c_conv-dev_BaDev_KWS_p-4gramCIPhone_1

NIST will match KWS and STT submissions using the EXPIDs. Sites are asked to harmonize their EXPIDs associating a same/similar EXPIDs for STT submissions so that KWS/STT associations are easily made. NIST will disregard the <VERSION> field during this mapping.

B.1.1 Scoring Server Cases

The Indus scoring server supports a variety of reporting options based on whether the reference transcripts are available for researchers to use or remain within the server. The submission routines and server will accept only legal data/server case combinations The following describes the level of detail provided for each use case.

BaDev - All the reports, DET curves, Threshold plots, and serialized DET curves (.srl.gz files usable with DETUtil to re-plot curves at the site) will be returned when scored.

BaEval - Reports will be delayed until NIST checks results from the evaluation. When NIST releases the scores, reference transcriptions will remain hidden for all (or part) of the data. After the formal evaluation for a given language, the scoring server will be set to automatically return results for the evaluation data if the string "POSTEVAL" is part of the <SYSID>.

B.2 KWS and STT System Output Formatting and Naming

System output files must be named with a valid EXPID and file extension. KWS system output must be formatted as KWSList files as described in Appendix A and use the extension 'kwslist.xml'. STT system output files must be formatted as CTM files as described in Appendix D and use the extension 'ctm'.

B.3 System Descriptions

Documenting each system is vital to interpreting evaluation results and disseminating systems to potential end users. As such, each submitted system, (determined by unique experiment identifiers), must be accompanied by a system description with the information specified in Appendix G.

B.4 System Output Submission and Scoring.

In order to make KWS or STT submission, you must have installed the NIST F4DE Package (including adding F4DE programs to your path) and completed the installation of data transfer license keys. Contact NIST (openkws-poc@nist.gov) for help completing the installation of these tools. The submission will be validated prior to upload to NIST.

- To make a KWS submission, execute the command:

% SubmissionHelper.sh –S <SYSTEM_DESCRIPTION>.txt <EXPID>.kwslist.xml

- To make an STT submission, execute the command:

  % SubmissionHelper.sh –S <SYSTEM_DESCRIPTION>.txt <EXPID>.ctm

The file "KWSEval/BABEL/Participants/README" contains several tips to use the SubmissionHelper.

B.5 Self-Validation Prior to Submission (optional)

The F4DE validation tools can be used by the site prior to submission using the following commands.

- For KWS:

  % KWS13-SubmissionChecker.sh -d KWS13-dbDir file.kwslist.xml

- For STT:

  % KWS13-SubmissionChecker.sh -d KWS13-dbDir file.ctm

B.6 Submission Checklist

- Create the system following the eval plan rules and ensuring the harvesting of runtimes as specified in Appendix H.
- Identify the DATADEF string for the system as described in Appendex I ensuring all dataset definitions exist or defining them as necessary.
- Develop the Experiment ID for the run.
- Run the data set though your system ensuring KWS and/or STT output is formatted according to Appendix A and validating the output per Appendix B Section B.5.
- Write the system description following the guidance in Appendix G.
- Submit the system output and system description per Appendix B Section B.4.

Appendix C: RTTM File Format Specification

Rich Transcription Time Marked (RTTM) files are space-separated text files that contain meta-data 'Objects' that annotate elements of the recording. Each line represents the annotation of 1 instance of an object. The RTTM file format is a cross-evaluation file format. As such, Object types can be used or not used depending on the particular evaluation.

For the KWS evaluations, the RTTM files are used as reference transcription files by the KWS scoring code. They are derived from the Babel transcripts. Speaker and Token object times were obtained by forced alignment of the transcripts to the audio.

There are ten fields per RTTM line. They are:

Table C.1 RTTM Field Names

Field 1	2	3	4	5	6	7	8	9	10
Type	file	chnl	tbeg	tdur	ortho	stype	name	conf	Slat

Fields 1 and 7: Object types (*type*) and object subtypes (*stype*): There are three general object categories represented in the Babel language packs: they are STT objects, source (speaker) objects, and structural objects. Each of these general categories may be represented by one or more types and subtypes, as shown in Table C.2.

Table C.2 RTTM object types and subtypes

Categories	Type	Subtype values (as text strings)
Structural	**SEGMENT**	**eval**, or <NA>
	SPEAKER	<NA>. Used to identify the smoothed segment boundaries fore Babel data.
	NOSCORE	<NA>
	NO_RT_METADATA	<NA>
STT	**LEXEME**	**lex, fp, frag, un-lex**[12]**, for-lex, alpha**[13]**, acronym, interjection, propernoun**, and **other**
	NON-LEX	**Laugh**, **breath, lipsmack, cough,** (translated from Babel's <laugh>, <breath>, <lipsmack>, and <cough> tags respectively), **sneeze** and **other**
	NON-SPEECH	**noise** (translated from Babel's <sta> tag)**, music**, and **other** (translated from Babel's, <click>, <ring>, <dtmf>, <prompt>, <overlap>, and <int> tags)
Source Info	**SPKR-INFO**	**adult_male, adult_female, child**, and **unknown** (if not available)

Field 2: File name (*file*): The waveform file base name (i.e., without path names or extensions).

Field 3: Channel ID (*chnl*): The waveform channel (e.g., "**1**" or "**2**").

Field 4: Beginning time (*tbeg*): The beginning time of the object, in seconds, measured from the start time of the file.[14] If there is no beginning time, use `tbeg` = "**<NA>**".

[12] Un-lex tags lexemes whose identity is uncertain and is also used to tag words that are infected with or affected by laughter.

[13] This subtype is an optional addition to the previous set of lexeme subtypes, which is provided to supplement the interpretation of some lexemes.

Field 5: Duration (*tdur*): The duration of the object, in seconds[14] If there is no duration, use `tdur` = "**<NA>**".

Field 6: Orthography field (*ortho*): The orthographic rendering (spelling) of the object for STT object types. If there is no orthographic representation, use `ortho` = "**<NA>**".

Field 8: Speaker Name field (*name*): The name of the speaker. `name` must uniquely specify the speaker within the scope of the `file`. If `name` is not applicable or if no claim is being made as to the identity of the speaker, use `name` = "**<NA>**".

Field 9: Confidence Score (*conf*): The confidence (probability) that the object information is correct. If `conf` is not available, use `conf` = "**<NA>**".

Field 10: Signal Look Ahead Time (*slat*): The "Signal Look Ahead Time" is the time of the last signal sample (either an image frame or audio sample) used in determining the values within the RTTM Object's fields. If the algorithm does not compute this statistic, `slat` = "<NA>".

This format, when specialized for the various object types, results in the different field patterns shown in Table C.3.

Table C.3 Format specialization for specific object types

Field 1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Type	*File*	*chnl*	*tbeg*	*tdur*	*ortho*	*stype*	*name*	*conf*	*SLAT*
SEGMENT	File	chnl	tbeg	tdur	<NA>	eval or <NA>	name or <NA>	conf or <NA>	<NA>
SEGMENT	File	chnl	tbeg	tdur	<NA>	<NA>	name or <NA>	<NA>	<NA>
NOSCORE	File	chnl	tbeg	tdur	<NA>	<NA>	<NA>	<NA>	<NA>
NO_RT_METADATA	File	chnl	tbeg	tdur	<NA>	<NA>	<NA>	<NA>	<NA>
LEXEME **NON-LEX**	File	chnl	tbeg	tdur	ortho or <NA>	stype	Name	conf or <NA>	slat or <NA>
NON-SPEECH	File	chnl	tbeg	tdur	<NA>	stype	<NA>	conf or <NA>	slat or <NA>
SPKR-INFO	File	Chnl	<NA>	<NA>	<NA>	stype	Name	conf or <NA>	<NA>

[14] If `tbeg` and `tdur` are "fake" times that serve only to synchronize events in time and that do not represent actual times, then these times should be tagged with a trailing asterisk (e.g., `tbeg` = **12.34*** rather than **12.34**).

Appendix D: CTM File Format Specification

Conversation Time Marked (CTM) files are space-separated text files that contain tokens output by the Speech-To-Text system. Each line represents a single token emitted by the system.

There are six fields per CTM line. They are:

Table C.1 CTM Field Names

Field 1	2	3	4	5	6
`File`	`chnl`	`tbeg`	`tdur`	`ortho`	`conf`

Field 1: File name (*file*): The waveform file base name (i.e., without path names or extensions).

Field 2: Channel ID (*chnl*): The waveform channel (e.g., "**1**").

Field 3: Beginning time (*tbeg*): The beginning time of the object, in seconds, measured from the start time of the file.

Field 4: Duration (*tdur*): The duration of the object, in seconds.

Field 5: Orthography field (*ortho*): The orthographic rendering (spelling) of the token.

Field 6: Confidence Score (*conf*): The confidence (probability with a range [0:1]) that the token is correct. If `conf` is not available, omit the column.

Appendix E: Language Resources

This appendix covers the build and evaluation pack resources for both the Babel Development language evaluations and the OpenKWS evaluations for the Babel Program Base Period (BP) and Option 1 Period (OP1).

The "Tokenization" column indicates variations from the most common transcript processing steps/scoring protocols for the language. The common scoring protocol relies on the language-specific transcription practices described in the LSP. In general, transcript tokenization is at the word level which is used by the KWS and STT evaluation protocols unchanged. The exceptions are:

- SplitCharacter -> non-ASCII UTF-8 characters are split into separate tokens. Hyphens are deleted if they become separate tokens. The additional options for "-x charsplit -x deleteHyphens -x notASCII" force this behavior.
- UnderAsSpace -> Underscore characters are transformed into spaces making them an extra tokenizing character. This applies to STT scoring and is implemented by pre-filtering the STT reference and system transcripts using the language specific Global Mapping file supplied in the IndusDB releases.
- STT:CER -> Character Error Rate is the preferred STT measure of performance instead of word error rate (WER). The additional sclite options "-c NOASCII DH" force this behavior.
- SyllableWER -> Reported WER statistics are technically syllable error rates because the transcripts are syllable-tokenized. No special scoring options are needed.
- TurkCConv -> Apply special case conversion prior to STT scoring for Babel Turkish data. The sclite options used are "-e utf-8 babel_turkish".

For each language, the required evaluation condition(s) are identified using the notation of Appendix I. The table shows the required conditions for the given evaluation "Year". Previous years' languages are enumerated for the record, but not required.

Babel Development Languages (applies to only Babel Performers)

Language ID-Name	Tokenization		Required Evaluation Condition Keyword Labels	
	Transcript	Scoring	DATADEF String	Year
101 Cantonese	Word	SplitCharacter, UnderAsSpace, STT:CER	BaseLR{101FullLP}: AM{101FullLP},LM{101FullLP},PRON{101FullLP},AR{None}	2012 BP
104 Pashto	Word	Word	BaseLR{104FullLP}: AM{104FullLP},LM{104FullLP},PRON{104FullLP},AR{None}	
105 Turkish	Word TurkCConv	Word	BaseLR{105FullLP}: AM{105FullLP},LM{105FullLP},PRON{105FullLP},AR{None}	
106 Tagalog	Word	Word	BaseLR{106FullLP}: AM{106FullLP},LM{106FullLP},PRON{106FullLP},AR{None}	
102 Assamese	Word	Word	BaseLR{102FullLP}: AM{102FullLP},LM{102FullLP},PRON{102FullLP},AR{None} BaseLR{102LLP}: AM{102LLP},LM{102LLP},PRON{102LLP},AR{None}	2013 OP1
103 Bengali	Word	Word	BaseLR{103FullLP}: AM{103FullLP},LM{103FullLP},PRON{103FullLP},AR{None} BaseLR{103LLP}: AM{103LLP},LM{103LLP},PRON{103LLP},AR{None}	
201 Haitian Creole	Word	Word	BaseLR{201FullLP}: AM{201FullLP},LM{201FullLP},PRON{201FullLP},AR{None} BaseLR{201LLP}: AM{201LLP},LM{201LLP},PRON{201LLP},AR{None}	

203 Lao	Token	Token	BaseLR{203FullLP}: AM{203FullLP},LM{203FullLP},PRON{203FullLP},AR{None} BaseLR{203LLP}: AM{203LLP},LM{203LLP},PRON{203LLP},AR{None}	
206 Zulu	Word	Word	BaseLR{206FullLP}: AM{206FullLP},LM{206FullLP},PRON{206FullLP},AR{None} BaseLR{206LLP}: AM{206LLP},LM{206LLP},PRON{206LLP},AR{None}	

Surprise Languages

Language ID-Name	**Tokenization**		**Required Evaluation Condition Keyword Labels**	
	Transcript	**Scoring**	**DATADEF String**	**Year**
107 Vietnamese	Syllables	SyllableWER	BaseLR{107FullLP}: AM{107FullLP},LM{107FullLP},PRON{107FullLP},AR{None} *BaseLR{107LimitedLP}: AM{107LimitedLP},LM{107LimitedLP},PRON{107LimitedLP},AR{None}	2013 BP
			BaseLR{…FullLP}: AM{…FullLP},LM{…FullLP},PRON{…FullLP},AR{None} *BaseLR{…LimitedLP}: AM{…LimitedLP},LM{…LimtedLP},PRON{…LimitedLP},AR{None}	2014 OP1

* Required for Babel Performers, encouraged for OpenKWS participants.

Appendix F: Auxiliary KWS and STT Performance Metrics

While the ATWV (Section 5.2.1) primary KWS measure estimates performance for a theoretic application that takes into account detection accuracy, detection score normalization and threshold setting, and the WER (Section 6.3) primary STT measure estimates performance of all space-separated tokens, it is useful to understand system performance vis-à-vis several measures. This Appendix defines alternative KWS and STT metrics that evaluate system performance from different perspectives.

F.1 Variant Measures of Performance for KWS

The measures below evaluate system performance without modifying the system output based on the reference annotations. These measures ignore some part of the system output without taking into account the reference annotation.

F.1.1 Ratio-specific TWV

"Ratio Term-Weighted Value" (RTWV@r) is the TWV obtained where the system's DET Curve intersects the line whose slope is the pre-determined ratio of $P_{Miss}/P_{FA} = r$. This particular TWV removes from performance assessment the system's ability to pick an optimum threshold and compares multiple systems at a common operating point defined by the ratio r.

F.1.2 Mean Average Precision

Mean Average Precision (MAP) measures the system's ability to rank true keyword occurrences among competing false alarms not taking into account the confidence score nor the detection threshold. MAP is the average of the average precision for

$$MAP \equiv \sum_{kw=1}^{K} \sum_{t=1}^{N_{true}(kw)} \frac{t}{rank(kw,t)}$$

Where *rank(t)* is the rank, obtained by sorting detection scores numerically, of keyword *kw*'s t^{th} true keyword occurrence.

F.1.3 TWV Including Keywords with No Reference Occurrences

TWV is calculated over terms with reference occurrences because $P_{Miss}(\theta)$ ($N_{Miss}(kw, \theta)/N_{True}(kw)$) of a non-occurring keyword is undefined when $N_{True}(kw)$ is 0. As a variant, TWV can be calculated using a different $P_{FA}(\theta)$ that incorporates false alarms for keywords without reference occurrences.

$$P_{Miss}(\theta) = \sum_{kw=1}^{K} \left[N_{Miss}(kw,\theta) / N_{True}(kw) \right] \Big/ K$$

$$P_{FA}(\theta) = \sum_{kw=1}^{K'} \left[N_{FA}(kw,\theta) / (N_{NT}(kw) - N_{True}(kw)) \right] \Big/ K'$$

K = # of keywords with 1 or more reference occurrences

K' = # of all keywords

When non-occurring keywords are included in the TWV calculations, the modified formula will be identified as being used in the analysis.

F.2 Oracle Measures for KWS

Oracle measures assess performance if a specific system-required response is performed without error. The following measures modify the TWV measure defined in Section 5 which are:

Let kw be a keyword and K the number of keywords. Given a detection threshold θ, define the KW-specific TWV at θ to be

$$TWV(\theta) = 1 - \left[P_{Miss}(\theta) + \beta \cdot P_{FA}(\theta) \right]$$

$$P_{Miss}(\theta) = \sum_{kw=1}^{K} \left[N_{Miss}(kw,\theta) / N_{True}(kw) \right] \Big/ K$$

$$P_{FA}(\theta) = \sum_{kw=1}^{K} \left[N_{FA}(kw,\theta) / (N_{NT}(kw)) \right] \Big/ K$$

F.2.1 Maximum TWV

"Maximum Term-Weighted Value" (MTWV) gives the upper bound TWV given perfect global thresholding. It is based on analyzing of the system's DET curve to find the maximum TWV found over the range of all possible values of θ. The difference between ATWV and MTWV, and the difference between the detection score thresholds for them, are an indication of how well the hard decision threshold was set.

F.2.2 Optimal TWV

The Optimal TWV (OTWV) oracle measure gives an upper bound on a system's performance under the assumption that it has perfect KW-specific thresholding. For each kw, we can choose the detection threshold $\hat{\theta}_{kw}$ that maximizes the keyword-specific TWV for a keyword's $P_{Miss}(kw,\theta)$ and $P_{FA}(kw,\theta)$,

$$\hat{\theta}_{kw} = \min_{\theta}(1 - [\, P_{Miss}(kw,\theta) \; + \beta \cdot P_{FA}(kw,\theta)])$$

Then we define the first oracle to be

$$OTWV \equiv 1 - \left[\sum_{kw=1}^{K} \frac{P_{Miss}(kw,\hat{\theta}_{kw})}{K} + \beta \sum_{kw=1}^{K} \frac{P_{FA}(kw,\hat{\theta}_{kw})}{K} \right]$$

OTWV is an upper bound on a system's ATWV since by construction for all θ.

$$ATWV(\theta) \leq OTWV$$

F.2.3 Supremum TWV

The Supremum TWV (STWV) oracle measure gives an upper bound on a system's performance under the assumption that we have perfect KW-specific detection probabilities and thresholding. The detection probabilities are modified using knowledge of ground truth to set a FA's probability to 0 and a true hit's probability to 1. For each keyword, let $N_{true}(kw)$ be the number of occurrences of kw

in the test set and let $N_{Hyp}(kw)$ be the number of hypothesized occurrences of kw in the system's posting lists. If we set the detection threshold θ = 0.5 then for each kw,

$$TWV(\theta) = N_{Hyp}(kw)/N_{true}(kw)$$

which is the system's recall of the KW if it had perfect detection probabilities. Then we define the Supremum TWV measure to be the average of these KW- specific recalls:

$$STWV \equiv \sum_{kw=1}^{K} \frac{N_{Hyp}(kw)/N_{true}(kw)}{K}$$

Clearly

$$ATWV(\theta) \leq OTWV \leq STWV$$

F.3 STT Variant Measures of Performance for STT

The following metrics evaluate STT performance from different perspectives.

F.3.1 Confidence Score Normalized Cross Entropy

As an additional performance measure, the quality of the token confidence scores will be evaluated. The confidence score represents the system's estimate of the probability that the output token is correct and must have a value between 0 and 1 inclusive.

The performance of this confidence measure will be evaluated using Normalized Cross Entropy (NCE). It is assumed that the role of the confidence score is to distribute the probability mass of a correct recognition (i.e. the percent correct) across all the system transcribed words.

$$NCE = \left\{ H_{\max} + \sum_{w=1}^{CorrectWord} \log_2(\hat{p}(w)) + \sum_{w=1}^{IncorrWord} \log_2(1-\hat{p}(w)) \right\} \Big/ H_{\max}$$

Where:

$$H_{\max} = -n\log_2(p_c) - (N-n)\log_2(1-p_c)$$

n = the number of correct system words

N = the total number of system words

$p_c = n/N$; the average prob. that a system word is correct

$\hat{p}(w)$ = the confidence of system word w

F.3.2 STT Character Error Rate

For some languages, e.g., Cantonese and Mandarin, Character Error Rate (CER) is a useful method to avoid the ambiguities of word segmentation procedures in the scoring process. In order to compute CER, both the reference and STT transcripts are modified by converting multi-character, non-roman text tokens into a separate text token for each character. After conversion, the WER error metric is applied in the character context.

Appendix G: System Descriptions and Auxiliary Condition Reporting

System descriptions are expected to be of sufficient detail for a fellow researcher to both understand the approach and the data/computational resources used to train and run the system.

The proposed structure of the system description includes the following six sections:

Section 1: Abstract

Section 2: Notable Highlights

Section 3: Data Resources

Section 4: Description

Section 5: Hardware Description

Section 6: Timing

Each system description section is covered below in a separate sub-section of this appendix. The Indus scoring server will require system descriptions to be included along with the system output.

In order to make system description preparation as simple as possible, developers are encouraged to write a single, detailed description that is referred to in contrastive (and potential cross-system) system descriptions. The author should expect the reader is already familiar with the Babel project, this evaluation plan, keyword spotting, and speech recognition in general.

Section 0: Evaluation Condition Keywords

This section contains the experimental keywords for the system as described in Appendix I. There is only one element in this section consisting of the value of <DATADEF> as defined in Appendix I.

Section 1: Abstract (<300 words)

A few paragraphs describing the system at the highest level. This should help orient the reader to the type of system being described and how the components fit together.

Section 2: Notable Highlights (<500 words)

A few paragraphs on the major differences between this system and a "conventional" system. Questions often answered are: How is this system different from a system published in a conference proceedings a few years ago? How is it different from all the other teams' submissions?

Section 3: Training Resources (as many words as needed)

This section describes the data resources used by the system and for which major components the resources were used. Developers will report the training data resources and evaluation conditions as described in "Appendix I: Experimental Conditions for Evaluations". The content of this section will be the value of the EBNF symbol <DATADEF> from the Appendix I EBNF. This string is required.

Section 4: Description (<2500 words for a full system desc.)

Sufficient detail should be provided for each component of the system such that a practitioner in the field can understand how each phase was implemented. You should be very brief or omit altogether components that are standard in the field.

For system combinations, there should be a section for each subsystem.

For each subsystem, there should be subsections for each major phase. They may be excluded if not relevant or if only standard methods are used (e.g. no need to describe how MFCCs are computed or tell us 25ms window and 10ms step). They may also refer to other subsystems or referent system descriptions if they share components.

Suggested Subsections:

- Signal processing - e.g., enhancement, noise removal, crosstalk detection/removal.
- Low level features - e.g., PLP, Gabor filterbank.
- Speech/Nonspeech –
- Learned features – e.g., MLP tandem features, DNN bottleneck features, etc.
- Acoustic Models – e.g., DNN, GMM/HMM, RNN, etc.
- Language Models – methods used
- Adaptation – e.g., speaker, channel, etc.
- Normalization - Normalizations not covered in other sections
- Lexicon – methods used to update
- Decoding – e.g., Single pass, multipass, contexts, etc.
- OOV handling – e.g., Grapheme, syllable, phoneme, etc.
- Keyword index generation –
- Keyword search –
- System combination methods – e.g., posting list, score, features, lattices.

Section 5: Hardware description

Requirements on the description of architecture will be here. Reporting of the following environment elements relate directly to the reporting of time and memory requirements.

- OS (type, version, 32- vs 64-bit, etc.)
- Total number of used CPUs
- Descriptions of used CPUs (model, speed, number of cores)
- Total number of used GPUs

- Descriptions of used GPUs (model, number of cores, memory)
- Total available RAM
- RAM per CPU
- Used Disk Storage (Temporary & Output)

Section 6: Timing

System execution times must be reported as shown in the "Sample Resource Report" of Appendix H: System Time Reporting.

Section 7: References

This section contains references to research papers and other KWS/STT system descriptions submitted for the KWS evaluation..

Sites are encouraged to choose a single system to provide an extensive system description for and then reference it in other system descriptions. The appropriate citation for a system description is the EXPID of the system and (if the referent system was already submitted to Indus) its Remote SHA.

Appendix H: System Timing Reporting

Overview

To provide the community with information about the resources required to use BABEL Keyword Search (KWS) systems, this appendix provides instructions for how to report time and memory requirements. The intention of these instructions is to balance the interests of the community with the burden on performers to produce these reports.

> *Caveat:* Wall-clock timing and memory usage are very 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.

Processing Stages

Under the NTAR (no audio reuse) condition, KWS is decomposed into two processing stages for the purposes of reporting resource usage.

1. **Ingestion**
    a. **Ingestion** is all processing that can be done without any query keywords. Decoding begins at the first command initiated in setting up the KWS system.
    b. This may not be the first command that "touches" the language pack. Initialization scripts, etc. are considered part of **Ingestion**
    c. **Ingestion** ends when all resources required for search have been prepared.
2. **Search**
    a. **Search** begins at the introduction of keywords.
    b. **Search** includes any pre-processing of keywords, including but not limited to:
        i. iv/oov testing
        ii. query expansion
        iii. phoneme conversion
    c. **Search** includes any pre-processing/re-processing of the index, including but not limited to:
        i. Re-scoring phone lattices
        ii. Query term whitelisting
    d. **Search** includes any post processing of scores, including but not limited to
        i. threshold learning
        ii. score normalization
        iii. system output combination
    e. **Search** ends with the construction of a final posting list file.

Note: Due to the diverse range of possible TAR strategies, this document does not define TAR processing stages. Performers reporting TAR conditions should define processing stages that reflect the major steps in their system. These definitions must be included in the system description along with the reported resource use.

Reporting Memory Usage and Wall-Clock Time

Performers are required to report 1) the elapsed wall-clock time, 2) the total processing time and 3) the required available memory.

Performers are given wide discretion in how to calculate these numbers.

/usr/bin/time –v is recommended as a low-overhead solution to report memory usage and time. As all major Unix distributions include this utility, this eliminates any installation requirement on performers. Moreover this provides a single method to retrieve timing and memory usage information.

Other timing and memory profiling resources are acceptable, including custom code inserted within ingestion and search modules. This timing must cover all operations in ingestion and search as if the execution were timed by a wrapping utility like '/usr/bin/time –v'.

Throughout the remainder of this document timing operations including "/usr/bin/time –v", and custom solutions will be written as **time**.

It will be easiest to calculate these numbers if **Ingestion** and **Search** are each executed by a single command. I.e. "sh ingest.sh /home/data/BABEL_107 /home/kws/107-index" and "sh search.sh /home/kws/107-index kwslist.xml". In this case, simply **time**ing these commands should generate information for reporting Elapsed time and memory. However, we realize this is an idealized view of how systems may run during development and evals. Information for resource reports may be constructed "by hand" from information obtained from running sub-modules independently. Guidance for how to do this is described in the following sections.

No reporting of system load is expected or required.

Reporting Time

Time reporting requires two measures: 1) **Elapsed** wall-clock time and 2) **Total** processing time. **Elapsed** time is the amount of time that a user needs to wait for each processing stage to complete. **Total** time is the total amount of CPU/GPU time each processing stage required. The rationale for including both measures is to provide information about how much improvement can be gained from additional cores (or alternately, how much performance would suffer on an architecture with fewer available cores).

For parallel subprocesses on multiple cores:

- Grid Engine and other governing processes will report timing information for spawned subprocessed.

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.

Elapsed Time:

Use "real" time consistent with that reported by '/usr/bin/time –v' on the line "Elapsed (wall clock) time (h:mm:ss or m:ss): X:XX.XX".

The times of all serial processes are *summed* to generate the **Elapsed** time.

For each step that is executed in parallel, the *maximum* time for the parallel processes is added to the **Elapsed** time.

Total Time:

Use "real" time consistent with that reported by '/usr/bin/time –v' on the line "Elapsed (wall clock) time (h:mm:ss or m:ss): X:XX.XX".

The time of all serial processes are *summed* to generate the **Total** time.

The time of all parallel processes are *summed* and added to the **Total** time.

Time Reporting for GPUs

The use of GPUs can obfuscate the total CPU time required. GPUs contain at least 100s of cores; it is non-trivial to impossible to get information about the usage of each core during processing to report **Elapsed** and **Total** time in a way that is comparable to 100s of traditional cores.

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

Reporting Memory usage

The goal in reporting memory usage is to describe the minimum memory required to execute the **Ingestion** and **Search** processes within the times reported.

Minimally, performers could report the total available memory to KWS processes. However, the KWS 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.

One option is to use maximum resident set size as reported by '/usr/bin/time –v' on the line "Maximum resident set size (kbytes): XXXXXX".

Since memory reporting is used to know how much memory an environment must have in order to run a KWS system, memory usage is calculated as the *maximum* memory used by any subprocess.

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

Memory Reporting for GPUs

If available, the maximum amount of memory concurrently allocated onto the GPU may be reported. However, it is expected that the maximum memory used on a GPU will 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.

Reporting Training Time and Memory

Training often involves restarting, debugging, and incremental processing that can make it difficult to measure the time and memory requirements for training as an end-to-end process. Thus participants report best effort "oracle" training times. Specifically, submodule training time and memory can be calculated independently, reconstructing total requisite training resources after the fact. This will provide best-case information about how long would it have taken and how much memory would be required if a user didn't have to restart any training process and if the last training recipe was in place from the start. This oracle reporting eliminates the need to retrain from scratch to generate this reporting component.

Determining parallel training processes

When a training process is inherently parallel, the resource reporting should be performed as described above.

Because the "oracle" resources are reporting, the training of independent components that *could* be performed in parallel may occur asynchronously as many submodules do not depend on one another.

Unless these independent components are, in fact, trained in parallel, these should be considered serial processes for resource reporting.

Sample Resource Report

Below is a sample resource report, which could be appended to a system description. Including the processing stage on each line facilitates easy **grep**ing. The '-' delimiter lends itself to easy **awk**ing. The colon is used in the time reporting.

Ingestion Elapsed Time (hh:mm:ss) - 1:23:45.67

Ingestion Total CPU Time (hh:mm:ss) - 12:34:56.78

Ingestion Total GPU Time (hh:mm:ss) - 12:34:56.78

Ingestion Maximum CPU Memory (gbytes) - 256

Ingestion Maximum GPU Memory (gbytes) - 32

Search Elapsed Time (hh:mm:ss) - 1:23:45.67

Search Total CPU Time (hh:mm:ss) - 12:34:56.78

Search Total GPU Time (hh:mm:ss) - 12:34:56.78

Search Maximum CPU Memory (gbytes) -256

Search Maximum GPU Memory (gbytes) - 256

Appendix I: Experimental Conditions for KWS Evaluations

This appendix describes the convention to be used to provide test condition descriptions of a submission. It enables more fine-grained declaration of standard resources used and enables the creation of new testing conditions. New conditions must be declared to enable other teams to participate in those conditions and to allow the analysis of these conditions.

The Base Period Babel Evaluation Plan outlined several terms for labeling experimental conditions:

1. BaseLR (Baseline language resource, i.e., the standard language pack) versus BabelLR (use of any exposed Babel language packs) versus OtherLR (use of additional non-babel resources)
2. FullLP (full language pack with 80 hrs of conversational audio of which some % is transcribed depending on the period of the program) versus LimitedLP (uses transcriptions in the subtrain directory plus all audio in training from FullLP)
3. NTAR (no test audio reuse) versus TAR (test audio reuse) which distinguishes between whether the audio is reused in any way after test keywords are made known.

Here we define a grammar that expands upon these conventions to enable the declaration of new experimental conditions. The grammar will declare the resources that are used and how they are applied in the model. Data set conditions come in three varieties: BaseLR, OtherLR, and BabelLR. There are four components that are affected by the data resource used: Acoustic Model (AM), Language Model (LM), Pronunciation Lexicon (PRON), and Audio Reuse (AR) of test audio.

Data set resources, corresponding to BaseLR, BabelLR, and OtherLR, will be declared as:

BaseLR{<resource>}, BabelLR{<resource-list>}, and OtherLR{<resource-list>},

respectively. If the resources are well defined, such as 107FullLP, then that resource name should be used; otherwise, a new unique identifier should be created and it should be posted together with a full description of the resource at <TBD>. Please note that BaseLR and BabelLR resources will either be the full set or subsets of FullLP or LimitedLP data. If the resource is a subset of a build pack, it is sufficient to provide a list of transcription files used as the description. On the other hand, OtherLR will involve new resources that are developed by performers (e.g., harvested text for an LM). If those resources can be shared with other teams, they will be deposited when the resource name is declared; otherwise, a description of the resource should be provided.

Data resources can be used by the AM, LM or PRON, and so a full experimental condition declaration must indicate how the resources are utilized in these components using the following form:

AM{<resource-list>}, LM{<resource-list>}, PRON{<resource-list>}

In addition, audio can be reused during Keyword search; hence, it would be ideal to declare resources using an audio reuse (AR) form:

AR{<resource-list>}

Here the resource list can be None, indicating the NTAR condition, or it can be subsets of the evaluation data for the TAR condition.

To create a new experimental condition, the above components must be combined into a full experimental condition string. See the grammar at the end of this section for a full description. Here we provide several examples:

1. BaseLR using 107FullLP for all components in the NTAR condition:

   BaseLR{107FullLP}: AM{107FullLP},LM{107FullLP},PRON{107FullLP},AR{None}

2. BaseLR using 107FullLP together with BabelLR resources 101FullLP,104 FullLP, 105 FullLP, 106FullLP in the NTAR condition:

   BaseLR{107FullLP},BabelLR{101FullLP+104FullLP+105FullLP+106FullLP}:AM{107FullLP+104FullLP+105FullLP+106FullLP},LM{107FullLP},PRON{107FullLP},AR{None}

   Or if a new resource ID 'BPDevFullLP' is defined:

   BaseLR{107FullLP},BabelLR{BPDevFullLP}:AM{107FullLP+BPDevFullLP},LM{107FullLP},PRON{107FullLP},AR{None}

3. BaseLR using 107FullLP for all components in the NTAR condition together with harvested text used for the LM and PRON:

   BaseLR{107FullLP},OtherLR{107Harvested1}:AM{107FullLP},LM{107FullLP+107Harvested1},PRON{107FullLP+107Harvested1},AR{None}

4. 107FullLP with automatically derived pronunciations:

BaseLR{107FullLP}:AM{107FullLP},LM{107FullLP},PRON{None},AR{None}

5. A baseline TAR condition:

   BaseLR{107FullLP}:AM{107FullLP},LM{107FullLP},PRON{107FullLP},AR{107EvalLP}

6. A baseline TAR condition where 10 seconds of audio per keyword was used which was defined as a new resource:

   BaseLR{107FullLP}:AM{107FullLP},LM{107FullLP},PRON{107FullLP},AR{107EvalLPMagic10}

7. A modified LimitedLP condition that uses a single hour of transcribed data which was defined as a new resource:

   BaseLR{107LLP1hrSubset}:AM{107LLP1hrSubset},LM{107LLP1hrSubset},PRON{107LLP1hrSubset},AR{None}

Teams must register all new experimental condition descriptors at <TBD>, so that they can be tracked and shared with other teams. Below is an EBNF grammar to define these strings.

Experimental Condition EBNF

DATADEF :== 'DATADEF :== ' <LRDEFS> ':' <USAGEDEFS>

LRDEFS :== 'BaseLR{'<RESID>'}' (',BabelLR{'<RESSet>'}')? (',OtherLR{'<RESSet>'}')?

- BaseLR is required, must contain a single FullLP or LimitedLP RESID.
- BabelLR and OtherLR are optional and can contain a set of RESIDs.

USAGEDEFS :== 'AM{'<RESSet>'},LM{'<RESSet>'},PRON{'<RESSet>'} ,AR{'<RESSet>'}'

- 'AM' -> Acoustic model training resources. This includes the audio an any supplied transcripts supplied with the data
- 'LM' -> Language model training resources. This includes only textual data to build the lexicon and token-to-token models.
- 'PRON' -> Pronunciation building resources. This includes the supplied pronunciations applicable to the lexicon for the LM.
- 'AR' -> Audio Reuse of test material. This specifies the audio reprocessed.

RESSet :== <RESID> ('+' <RESID>)*

RESID :== A dictionary-defined text string that must not contain a '+', ':', or ',', e.g.,

- '101FullLP' -> the 101 Full Language Pack
- '101LimtedLP' -> the 101 Limited Language Pack
- 'None' -> no resources were used
- …
- 'BPDevFullLP' -> 101FullLP, 104FullLP, 105FullLP, and 106FullLP
- 'Harvest107' -> Jim Bob spent 2 hours searching the web for parallel texts for 107
- '107LimitedLP1hrSubset' -> The 107LimitedLP transcripts were further reduced to 1 hour of transcripts

Resource IDs for further constraining the use of an evaluation data set

- '101EvalLPMagic10' -> The top 10 hits per KW in the 107 evaluation pack audio where use to adapt the phones.