OpenKWS13 Keyword Search Evaluation Plan

1 INTRODUCTION

This document contains the evaluation plan for the 2013 Open Keyword Search (OpenKWS13). The OpenKWS evaluation will include only a "surprise" language for which the identity is revealed ad the time of release of the data.

Language resources will be provided to developers according to the Language Specific Peculiarities documents (one per language) and the Babel Data Specification Document.

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

Data sets and documentation will be provided to researchers in Babel language packs, Language Specific Peculiarities documents, and the Babel Data Specification Document. There will be two audio data distributions for each language, a build and evaluation pack, along with an IndusDB that contains 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 in only the development test data must be excluded during model training.

The build pack can be used in two ways representing two different amounts of transcribed material.

- The full language pack (**FullLP**) – all data resources provided by the program can be used for development.
- The limited language pack (**LimitedLP**) – a 10-hour subset of the transcriptions, metadata, and lexicon of the full language pack and all the audio data of the full language pack.

Developers have the option to use private resources however developers must self-report team supplied data resources per section 3.1.2[2].

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 an original language's orthography, in a corpus of un-segmented speech data.

3.1 KWS Evaluation Conditions

The goal of the KWS evaluation is two fold: 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, teams will classify their submissions across the following three conditions: (1) the build pack of the test language, (2) the additional language resources brought to bear, and (3) the amount of test audio re-use.

3.1.1 Test Language Build Pack

The build pack of the test language (which is defined in Section 2) can be either the **FullLP** or the **LimitedLP**.

3.1.2 Additional Language Resources

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[3].
- Non-test language Babel LRs (**BabelLR**): Additional LRs consist of any/all Babel-supplied language packs. No distinction will be made between FullLP and LimitedLP usage. (Note: This level, and data supporting it, is only available to Babel performers.)
- Other LRs (**OtherLR**): Addition LRs consist of team-, community-, or pre-existing-LRs.

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[4].
- Test audio re-use (**TAR**): the system re-processes the audio with knowledge of the search keywords.

[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.

[2] The Babel PM must acknowledge Team-developed resources prior to use.

[3] 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.

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

3.1.4 Required and Optional Evaluation Conditions

For the surprise language, all participants must submit a FullLP+BaseLR+NTAR system. Babel performers are required to submit a LimitedLP+BaseLR+NTAR system while OpenKWS13 participants are encouraged to submit a system for the LimitedLP+BaseLR+NTAR condition.

3.1.5 Primary vs. Contrast Systems

For a given combination of the three conditions, a site may choose to submit multiple runs sharing the same levels. Sites must designate one of their multiple runs for single ensemble of condition levels as their "primary" run the rest as "contrastive" runs. The primary should be their expected, best performing run. For example, a team may submit 2 primary runs, one for FullLP+BaseLR+NTAR and one for LimitedLP+BaseLR+NTAR

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.
- 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[5]. There are two methods of determining the presence of a keyword production based on the style of alignment and scoring protocol.

For keyword occurrence alignments, the following rules apply (Section 5.1):

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.

For inferred-segment alignment, the following rules apply (Section 5.2):

1-3. Rules 1-3 for occurrence alignments
4. System identified keywords are assigned to the segment containing the keyword's mid-point.

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

3.3 Transcription Normalization

Each language will require differing text normalization strategies to accommodate language specific transcription practices. The auxiliary document, "Text Normalizations for the Babel KWS and STT Evaluations"[6], describes text normalization steps for each language.

3.4 Non-Scored Regions

Segments containing the <overlap> and <prompt> tags will not be 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[7].) 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.

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:

```
% 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 RTTM 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 type information.

[6] The "Text Normalizations for the Babel KWS and STT Evaluations" document will be available on the evaluation website along with this document.

[7] 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

The STT task is a diagnostic task 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 but rather optimizing STT for KWS.

4.1 STT Evaluation Conditions

STT systems will be differentiated using the two KWS data conditions: (1) the test language build pack and (2) the additional language resource conditions.

Sites who use word-based models are encouraged to submit STT outputs supporting their KWS systems. As a voluntary task, there are no required STT evaluation conditions levels that participants must submit runs for.

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 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 hesitations 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. Assuming the SCTK tools have been installed and added to your path variable, the following commands will convert an RTTM formatted system output to a CTM-formatted file for scoring with sclite.

```
% rttm2ctm.pl < file.rttm > file.ctm

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

5 KEYWORD SEARCH EVALUATION

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

Two different scoring protocols will be used for KWS scoring. The first, "Keyword Occurrence Scoring", is the official metric for performance assessment, and the second, "Inferred Segment Scoring", is an experimental protocol. Both protocols use the same three steps to evaluate a system: (1) reference-to-system keyword alignment, (2) performance metric computation, and (3) diagnostic measure computation.

5.1 Keyword Occurrence Scoring Protocol

The keyword occurrence scoring protocol evaluates system accuracy based on a temporal alignment of the reference keywords to system-hypothesized keywords.

5.1.1 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[8] 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:

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

$O_{r,i}$ = 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.

5.1.2 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[9].

5.1.2.1 Term Weighted Value

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 ($P_{FA}(\theta)$).

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

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

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

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:

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

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.
- For Broadcast News and Conference Meeting data, the duration of the signal is used.
- For 2-channel conversational telephone speech, the union of the evaluated time 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.

5.1.2.2 Actual TWV

TWV can be calculated through the complete range of a system's detection score values. In order to ensure developers optimize system performance to the same operating point, the KWS evaluation uses the TWV for system occurrences with 'YES' hard decisions as the primary evaluation measure. This measure is referred to as Actual TWV.

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

5.1.3 Keyword Occurrence Diagnostic Measures

The KWS evaluation will experiment with several diagnostic evaluations methods and measures. This section describes the current set of diagnostics.

5.1.3.1 Detection Error Tradeoff

The detection scores output by a system permits 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 visualizes the tradeoff between the probabilities of missed detection versus false alarm.

5.1.3.2 Maximum TWV

"Maximum Term-Weighted Value" (MTWV) is, based on an an analysis of the system's DET curve, 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.

5.1.3.3 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, labels will indicate the modified formula was used.

5.2 Inferred Segmentation Keyword Search Evaluation Procedure

The Inferred-Segmentation (IS) keyword search evaluation procedure will be used as an alterative to the keyword occurrence alignment procedure as outlined in Section 5.1.

The IS procedure is similar to the occurrence alignment approach in that keywords are evaluated independently and reference keywords are identified in the same manner. However, the two methods differ in step 1, the alignment phase, and in step 2, the metric computation formulas. Step 3, diagnostic computation, is essentially the same except that it uses the modified metric formulas.

5.2.1 Inferred Segmentation Reference-to-System Alignment

After the reference keywords are identified, both the reference and system identified keywords are mapped onto the speech/non-speech segmentation of the recording. The reference segmentation will be modified to ensure no keywords span speech/non-speech boundaries. The system identified keywords are mapped to the segments using the temporal mid-point.

The speech/non-speech segments become the "trials" (the unit of performance measurement) for computing system performance. The following summarizes how the segments are scored:

- Correct Detection: a segment containing one or more reference keyword occurrence(s) and one or more midpoints of system identified keyword occurrence(s).
- Correct Non-Detection: a segment not containing a reference keyword occurrence or a midpoint of system identified keyword occurrence.
- Missed Detection: a segment containing one or more occurrence(s) of a reference keyword but no midpoints of system identified occurrences of that keyword.
- False Alarm: a segment not containing an occurrence of a reference keyword but containing one or more midpoints of system identified occurrences of that keyword.

5.2.2 Inferred Segmentation Performance Metric Computation

Term Weighted Value measures false alarms as a fraction of false alarms per expected number of non-target trials. The use of speech/non-speech segments provides a trial counting mechanism that enables the computation for segment-based missed detection and false alarm probabilities. The two measures are combined in the Segment-based Term Weighted Value (STWV).

5.2.2.1 Segment-based Term Weighted Value

Segment-based term-weighted value (*STWV*) is a 1 minus the weighted sum of the term-weighted probability of missed detection segments ($P_{MissSeg}(\theta)$) and the term-weighted probability of false alarm segments ($P_{FASeg}(\theta)$).

$$STWV(\theta) = 1 - \left[P_{MissSeg}(\theta) + \beta \cdot P_{FASeg}(\theta) \right]$$

Where:

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

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

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

K = # of keywords with 1 or more reference occurrences

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

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

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

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

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

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

$N_{TarSeg}(kw)$ = # segments containing *kw*

$N_{NonTarSeg}(kw)$ = # segments not containing *kw*

5.2.3 Inferred Segmentation Diagnostic Measures

The same diagnostic methods as described in Section 5.1.2 and 5.1.3 will be used for the inferred segmentation scoring protocol.

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, (3) performance metric computation, and (4) diagnostics measure computation.

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. 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:

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

where

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

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[10]

6.4 Diagnostic Measures for STT

This section describes the current set of STT diagnostics.

6.4.1 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

6.4.2 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.

6.4.3 STT Syllable Error Rate

For languages where lexical items include multiple syllables, Syllable Error Rate (SER) will be calculated by transforming both the reference transcript and system-generated transcripts into syllable units using the lexicon provided in the language packs.

Syllables will be constructed by concatenating phones of a syllable into a single lexical unit. When multiple pronunciations exist for a lexical item, the first pronunciation will be used for the reference transformations (to enforce a common reference across systems) and all possible pronunciations will be used for the system-generated transcripts.

After conversion, the WER error metric is applied in the syllable context.

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 with Test Audio

No manual or human interaction with the test audio data is allowed.

8 PUBLICATION OF RESULTS

Publication of vetted results is encouraged and should be in accordance with your Babel contract.

9 DATA STRUCTURES AND FORMATS

System output will be stored in an XML-formatted text files as specified in Appendix A.

[10] 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.

10 SUBMISSION OF RESULTS

Submissions will be made via the Babel Scoring server as specified in Appendix B which explains the submission protocol. In addition to the system output results as specified above, a system description is also required for each submission. This description must include a detailed description of the architecture and algorithms used in the system.

In order to simplify writing system description and to make system descriptions somewhat homogenous across sites, participants are encouraged to fully document one of their primary systems (NIST suggests the primary, surprise language, FullLP+BaseLR+NTAR configuration or similar) and then document differences in the rest of the system descriptions.

11 SCHEDULE

Consult evaluation schedule on the OpenKWS13 web site.

Appendix A: KWS Evaluation Implementation Details

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 input and system output.

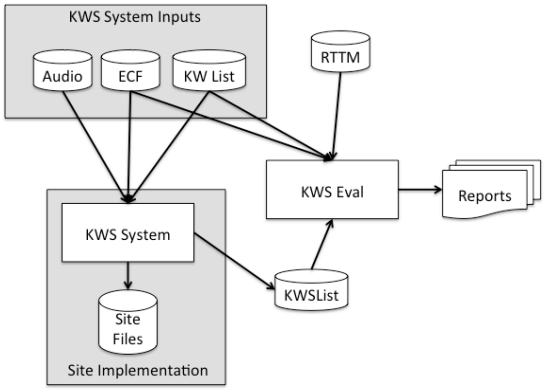

Figure 1: System and evaluation inputs and outputs

The three input files to the site-implemented KWS system are as follows[11].

- Audio files: SPHERE formatted waveform files organized as originally distributed.

  Experiment Control File (ECF): ECF files define the excerpts within audio files to be used for specific experiments and the language/source type of each file.

- KWlist: The list 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 RTTM (Appendix C) file to produce the performance analysis reports.

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

A.1 Experiment Control Files

Experiment Control Files (ECF)s 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 a 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/NIST_2
     0020214-1148" channel="1" tbeg="0.0" dur="291.34"
     source_type="confmtg"/>
<excerpt
     audio_filename="audio/eval03/english/bnews/ABC_WN
     N_20020214_1148.sph" channel="1" tbeg="0.0"
     dur="291.34" source_type="bnews"/>
…
</ecf>
```

A.4. KWList Files

A 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

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

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[12] 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.5 KWSList Files

The 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 “kw” nodes which are the putative detected keywords. The XML scheme for an 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.
- 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.

[12] The basename of a file excludes the directory names and extensions. For example the basename of “the/directory/file.txt” is “file”.

- 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.

An example KWSList file is:

```
<kwslist
    kwlist_filename="expt_06_std_eval06_mand_all_spch_e
    xpt_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"/>
</detected_kwlist>
</kwslist>
```

Appendix B: System Output Submission and Scoring

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.

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 :== KWS13_<TEAM>_<CORPUS>_<PARTITION>_<SCASE>_<TASK>_<LP>_<LR>_<AUD>_<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 | BaSurp (See Scoring Cases below for descriptions)

<TASK> :== KWS | STT

<LP> :== FullLP | LimitedLP (See Section 3.1.1)

<LR> :== BaseLR | BabelLR | OtherLR (See Section 3.1.2)

<AUD> :== NTAR | TAR (See Section 3.1.3)

<SYSID> :== a site-specified string (that does not contain underscores) designating the system used. The SYSID string must be present. It is to begin with *p-* for the one and only primary system (i.e., your single best system for a given set of <LP>, <LR>, and <AUD>) or with *c-* for any contrastive systems. It is then followed by an identifier for the system (only alphanumerical characters allowed, no spaces). 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.

<VERSION> :== 1..n (with values greater than 1 indicating resubmitted runs of the same experiment/system)

Currently, the following EXPIDs are supported. If the element is of the form "<..>" any legal BNF value is accepted.

KWS13_<TEAM>_babel101b-v0.4c_conv-dev_BaDev_<TASK>_<LP>_<LR>_<AUD>_<SYSID>_<VERSION>

KWS13_<TEAM>_babel101b-v0.4c_conv-dev_BaEval_<TASK>_<LP>_<LR>_<AUD>_< SYSID>_<VERSION>[13]

B.1.1 Scoring Server Cases

The Indus scoring server supports a variety of reporting options based on if the references are available for researchers to use. The submission routines and server will enforce compliance for compliance. The following describes the level of detail provided for each use case.

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

BaEval - Reports will be delayed until NIST checks results from the evaluation. When NIST releases the scores, references will remain hidden. The specific contents are TBD.

BaSurp - Reports will be delayed until NIST checks results from the evaluation. When NIST releases scores, a select set of reports will be returned. The specific contents are TBD.

B.2 System Output Formatting and Naming

[13] Note: BaEval will not be available until the Dry Run starts.

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 **RTTM** files as described in Appendix C and use the extension 'rttm'.

B.3 System Descriptions

Documenting each system is vital to interpreting evaluation results. As such, each submitted system, (determined by unique experiment identifiers), must be accompanied by a system description with the following information.

Section 1 Experiment Identifier(s)

List all the EXPIDs for which system outputs were submitted. EXPIDs are described in further detail above.

Section 2 System Description

Provide a brief technical description of your system; if a contrastive test, contrast with the primary system description. Please include contact information (name and email) for the submission.

Section 3 Indexing Hardware Description and Runtime Computation

Describe the hardware setup(s) (an aggregatation of computation components used to perform a processing step) and report the Total Processing Time (TPT) for each phase. The evaluation defines two phases: decoding/indexing and search. Each phase will be cdumented in a separate sub section:

- **Section 3.1 Decoding and Indexing**: processing the test audio and building the data structures to prepare for keyword searches.
- **Section 3.2 Search**: the process of finding and reporting keyword hits in the test corpus.

Each phase may be broken down into sub-steps in which case the hardware and processing time of each sub-step must be documented as a textual description. The following is an example:

Section 3.1 Decoding and Indexing:

Two compute clusters were used: Castor and Polydeuces

- Castor: NVIDIA Quadro 6000 GPU, 448 CUDA cores, 6GB shared memory
- Polydueces: a 16-node, Dual Quad Core 2.26 GHz Intel Xeon, 24GB RAM per node, with a 10TB Data Server.

Decoding and indexing were broken down into 3 sub-phases distributed to the two clusters:

- Feature extraction - Caster – 0.53 hours
- Lattice generation –Polydueces – 63.6 hours
- Indexing – Polydueces (1 node) – 1.4 hours

Section 4 Training data and knowledge sources

List the resources used for system training, development, and runtime knowledge sources beyond the provided Babel corpora.

Section 5 References

Provide a list of pertinent references.

Figure 2: System Description Template

B.4 System Output Submission and Scoring.

Currently, submission instructions are only available for the KWS task. Submission instructions for the STT task will be added later. In order to make a 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 (Martial Michel, martial.michel@nist.gov) for help completing the installation of these tools.

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

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.

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>
	NOSCORE	<NA>
	NO_RT_METADATA	<NA>
STT	**LEXEME**	**lex, fp, frag, un-lex**[14]**, for-lex, alpha**[15]**, 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.[16] If there is no beginning time, use `tbeg` = "**<NA>**".

Field 5: Duration (*tdur*): The duration of the object, in seconds[16] 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>**".

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

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

[16] 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**).

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>
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>