The Rich Transcription Fall 2003 (RT-03F) Evaluation Plan

1 INTRODUCTION

The goal of this document is to define the evaluation tasks, performance measures, and test corpora to support the 2003 Rich Transcription Fall (RT-03F) evaluation. Rich Transcription (RT) is broadly defined to be a fusion of speech-to-text (STT)[1] technology and metadata extraction (MDE) technologies which will provide the basis for the generation of more usable transcriptions of human-human speech for both humans and machines. This (Fall RT-03) evaluation is a follow-on adjunct to the main (Spring) RT-03 evaluation, which was the second evaluation in a series intended to support the research and development of individual RT transcription and extraction components. This series provides the evaluation mechanisms to support DARPA's Effective, Affordable, Reusable Speech-to-text (EARS) Program.[2] Note, however, that in addition to EARS contractors, this evaluation is open to all interested volunteers. All participants will be required to attend the RT-03 Fall Workshop which will follow the evaluation.

The RT-03F evaluation will focus on areas that were deferred from the main Spring evaluation. Evaluation will be supported for six tasks:

Edit Word Detection

Filler Word Detection

IP Detection

SU Boundary Detection

Speaker Attributed STT

RT03 Rich Transcription

The RT-03F evaluation will be limited to English language only.

2 BACKGROUND

Beginning in the early 1980's, evaluation of automatic speech recognition (ASR) stabilized on the current performance measure of word error rate (WER). This measure scores ASR performance using a case-less lexicalized form of ASR output known as the "standard normalized orthographic representation" (SNOR) format.[3] The WER is defined as the sum of all ASR output token errors divided by the number of scoreable tokens in a reference transcription of the test data. There are three types of errors, these being tokens that are missed (deletion errors), inserted (insertion errors), and incorrectly recognized (substitution errors).[4]

While the traditional STT evaluations have helped to provide a mechanism for evaluating word accuracy, it is clear that words alone are insufficient in formulating a transcription of speech which is readable by humans and understandable by machines. A verbatim transcription of the speech stream into a string of lexical tokens yields a transcript that is often extremely difficult to understand. This is because spoken language is much more than just a string of lexical tokens. It contains information about the speaker, prosodic cues to the speaker's intent, and much more. Spoken language also contains disfluencies, which speakers correct and which textual renderings should delete. All of this makes the task of rendering spoken language into text a great challenge, especially with less-than-perfect ASR performance.

Solving these problems is the challenge that the EARS program takes as its objective and what the RT evaluation series seeks to assess – namely to develop technology that transforms spoken language into a form that is maximally informative. This requires new approaches to acoustical modeling and insightful models of disfluencies, dialogue and other relevant speaker behaviors.

3 THE NATURE OF DISFLUENCIES (IN BRIEF)

In EARS, disfluencies are portions of speech in which a speaker's utterance is not complete and fluent but that the speaker corrects, repeats, or abandons. Disfluencies are fully discussed and explained in the Simple MDE Annotation Specification[5] ("SimpleMDEV5.0"). The EARS motivation to detect disfluencies is to enable "clean up" of rendered text. Portions of disfluencies can be removed, or processed another way, to improve the readability of transcribed texts.

The two disfluency types, edits and fillers, have similar structures, but they are independent speech events. Thus, their detection has been divided into separate tasks.

There is a common structure to disfluencies. They consist of a DEPOD, an interruption point, and optionally a correction. The ordering of these elements and their presence defines the type of disfluency. Disfluencies are often nested; however, the EARS program has decided to address only the top most level at this time.

- A DEPOD is an EARS neologism defined as the DEletable Part Of a Disfluency. It is either a filler, (e.g., um, uh) or the speaker's initial attempt at an utterance. Note that because edit and filler disfluency tasks are treated independently, structures in which a

[1] formerly known as automatic speech recognition (ASR)

[2] The EARS research effort is dedicated to developing powerful new speech transcription technology that provides substantially richer and more accurate transcripts than are currently possible. The research focus is on natural, unconstrained speech from broadcasts and telephone conversations in a number of languages. The program objective is to create core enabling technology suitable for a wide range of advanced applications.

[3] Since some languages' written forms are not word-based, this concept has been extended to cover lexemes – a representation of a written unit of meaning within a language. Thus, this document frequently refers to lexemes, lexical tokens, or tokens rather than words. For English, these terms may be treated more or less equivalently.

[4] Underlying the tabulation of errors is a requirement to align the tokens in the system output transcript with the tokens in the reference transcript. Traditionally, this has been done using a dynamic programming algorithm that searches for an alignment that minimizes the WER.

[5] http://macears.ll.mit.edu/macears_docs/data/SimpleMDE_V5.0.pdf

filler directly follows an edit DEPOD are treated as two separate DEPODs (the edit DEPOD and the filler DEPOD). DEPODS are candidates for deletion (or other types of special handling) in the production of a rich transcript.

- An interruption point (IP) is a prosodic phenomenon that indicates a transition from fluent to non-fluent speech.
- A correction consists of the portion of the utterance that has been repaired and is fluent. Correction portions are not always present.

Edit disfluencies have a DEPOD, one or more IPs and optionally a correction. There are four edit disfluency subtypes: repetitions, revisions, restarts and complex. Complex edits are nested disfluencies and have multiple IPs. IPs for edits are on the right edge of the DEPOD, (and within the DEPOD for complex edits).

Filler disfluencies exhibit a DEPOD and an IP. Unlike the edit disfluencies, the IP is on the left edge of the DEPOD. There are four subtypes of fillers defined by the Simple MDE Annotation specification: filled pauses, discourse markers, explicit editing terms, and asides and parentheticals[6]. Asides and parentheticals are not evaluated.

4 THE RT-03F TASKS

RT-03F features a variety of tasks that are each being evaluated for the first time. Two task and scoring frameworks have been developed in support of the evaluation. Though in many cases the frameworks share concepts and terminology, there are enough differences to warrant separate treatment in this evaluation plan. Evaluation participants may choose to optimize their systems to either of the two frameworks. However, system outputs will be scored using both frameworks.

4.1 SCOREABLE TOKENS

Both frameworks described below use sequences of tokens to represent acoustic events in the speech signal. Such token sequences can be used for two purposes. First, they can be used to align the system output with the reference. Second, they can be used to measure the accuracy of the system output against the reference. The underlying definitions of token types are common between the two frameworks even though the alignment and scoring methods are different.

We refer to four token types as "scoreable". Only tokens of these four types influence the alignment process. Additionally, in metadata detection tasks, systems detect or ascribe attributes to all tokens of these types, and error measures will be computed by considering all tokens of only these types appearing in the reference and system token sequences. These four scoreable token types are:

- lexeme – a representation of a word spoken in the primary language.
- foreign-lexeme –a representation of a word spoken in a language different from the primary language.
- fragment – a representation of a partial word spoken in any language.
- filled-pause – a representation of a hesitation sound that speakers employ to indicate uncertainty or to maintain control of the conversation while thinking of what to say next (see SimpleMDEV5.0 Section 2.1).

For all scoreable tokens, identical, un-cased orthography matches between the reference and system outputs constitute a correct match during the token alignment and token scoring process (see GLM processing in Appendix C for additional information). For fragment and filled pauses, if the orthographies do not match but the types do match, the tokens are considered a correct match during the token alignment and token scoring process.

Certain token types can occur in reference token sequences but will never fall within an evaluable region of the evaluation data. There is one such token type:

- unclear-lexeme – a representation of a word whose identity is not clear to the human transcriber.

Three other token types may be emitted by systems or may be present in the reference. These tokens will be ignored by scoring algorithms whether they occur in system output or reference token sequences:

- non-lexeme – a representation of the five canonical vocal noises produced by the speaker: cough, sneeze, breath, lipsmack, and laugh.
- non-speech – a representation of a non-speech acoustic phenomenon like door-bang, etc.
- miscellaneous[7] – a representation of any other event that the system outputs for its own purposes.

4.2 NIST METADATA EXTRACTION FRAMEWORK

The NIST metadata extraction framework defines four metadata detection tasks for RT-03F. Systems are given only a digital audio signal as input. Some of the tasks are defined in terms of detection of "extent", i.e. the system must detect and output one or more spans indicating the locations and durations of particular metadata events. Others tasks require the detection of "points", i.e. the system must detect and output events that occur at a particular instant in time. Though four tasks are defined, a system may implement any combination of the four tasks.

Though this framework is very general, for the purposes of RT-03F specific framework options will be invoked. For example, the framework allows a system to produce metadata without requiring it to produce any corresponding STT output. However, in RT-03, STT output will be required from each system to allow for alignment between reference and system hypotheses and for the congruence of STT and metadata events.[8]

STT output will be in the form of a sequence of tokens. Start times and durations must be included for each such token.

4.2.1 EDIT WORD DETECTION

The Edit Word Detection task is to detect regions of the input signal containing the DEPODs of edit disfluencies

[6] Asides and paretheticals are treated as one subtype in SimpleMDEV5.0 and are not evaluated in the RT-03F evaluation.

[7] CTM records of type "MISC" will be translated into RTTM records of type "NON-SPEECH" with subtype "other".

[8] For diagnostic purposes, performance will also be reported without applying this STT-based alignment, but this will be a secondary measurement.

("edit DEPODs"). Edit DEPODs are defined in SimpleMDEV5.0 Section 3.2. For RT-03F evaluation, the detection task requires the system to specify the start and duration of all regions of the input signal containing tokens of edit DEPODs.

There is no reward or penalty for splitting a single detected region into two or more contiguous regions having identical overall extent. Nor is there any reward or penalty for combining two or more contiguous detected regions into a single detected region of identical extent.

Filler tokens may occur within an edit DEPOD (SimpleMDEV5.0 Section 3.7.2). For the purposes of the Edit Word Detection task, the fact that an edit DEPOD token is also a filler token is irrelevant, i.e. regions containing such tokens should be detected as part of this task.

For RT-03F evaluation, automatic identification of edit disfluency subtype is not part of the Edit Word Detection task.

4.2.2 FILLER WORD DETECTION

The Filler Word Detection task is to detect regions of the input signal containing DEPODs of filler disfluencies ("filler DEPODs"). Fillers are defined in SimpleMDEV5.0 Section 2. For RT-03F evaluation, the detection task requires the system to specify the start and duration of all regions of the input signal containing tokens of fillers.

There is no reward or penalty for splitting a single detected region into two or more contiguous regions having identical overall extent. Nor is there any reward or penalty for combining two or more contiguous detected regions into a single detected region of identical extent.

Filler tokens may occur within an edit DEPOD (SimpleMDEV5.0 Section 3.7.2). For the purposes of the Filler Word Detection task, the fact that the filler token is within an edit DEPOD is irrelevant, i.e. regions containing such tokens should be detected as part of this task.

Section 2 of SimpleMDEV5.0 defines four subtypes of fillers (filled-pauses, discourse markers, explicit editing terms, asides/parentheticals). For the purposes of the Filler Word Detection task, regions containing fillers of subtype "aside/parenthetical" should not be detected, i.e. the current task is limited to detection of regions containing three subtypes of fillers: filled-pauses, discourse markers, and explicit editing terms.

For RT-03F evaluation, automatic identification of filler subtype is not part of the Filler Word Detection task.

4.2.3 IP DETECTION

The interruption point (IP) detection task is to produce the locations in time where interruption points occur. Interruption points are defined in Footnote 1 and Section 3.2 of SimpleMDEV5.0. For RT-03F evaluation, the detection task requires the system to specify the location in time of each interruption point.

When a filler follows an edit DEPOD, systems should output either one or two IPs according to the following rule.

When a filler follows an edit DEPOD within the same SU[9] (i.e. not separated by an incomplete or complete SU boundary) and when there are no intervening tokens of type "lexeme" (see Appendix A) between the edit DEPOD and the filler, a single, shared IP should be output. The location of such a shared IP should be specified as the time of the end of the edit DEPOD. This sharing is independent of the gap in time between the end of the edit DEPOD and the filler. If these conditions are not met, two IPs should be emitted.

For RT-03F evaluation, automatic identification of IP subtype is not part of the IP detection task.

4.2.4 SU BOUNDARY DETECTION

The SU Boundary Detection task is to detect SU endpoints. The definition of an SU is provided in SimpleMDEV5.0 Section 4. For RT-03F evaluation, the detection task requires the system to specify the location in time of the end of each SU. However, this specification is made indirectly – the system outputs an SU extent (start time and duration) from which the end time is computed.

For RT-03F evaluation, automatic identification of the subtype of SU is not part of the SU Boundary Detection task.

4.3 BBN RICH TRANSCRIPTION FRAMEWORK

The BBN Rich Transcription Framework defines one comprehensive RT task (Section 4.3.6) whose goal is to produce a sequence of scoreable tokens (Section 4.1) augmented with five metadata attributes, given only digital audio signals as input. RT systems may elect to implement all five or any subset of the metadata tasks described below (Sections 4.3.1 – 4.3.5).

4.3.1 EDIT WORD DETECTION

The Edit Word Detection task is to produce a sequence of scoreable tokens and to detect the tokens that are contained within the DEPODs of edit disfluencies ("edit DEPODs"). Edit DEPODs are defined in SimpleMDEV5.0 Section 3.2. For RT-03F evaluation, the detection task requires the system to specify whether each token is part of an edit DEPOD or not.

Filler tokens may occur within an edit DEPOD (SimpleMDEV5.0 Section 3.7.2). For the purposes of the Edit Word Detection task, the fact that an edit DEPOD token is also a filler token is irrelevant, i.e. such tokens should be detected as part of this task.

For RT-03F evaluation, automatic identification of edit disfluency subtype is not part of the Edit Word Detection task.

4.3.2 FILLER WORD DETECTION

The Filler Word Detection task is to produce a sequence of scoreable tokens and to detect the tokens that are contained within the DEPODs of filler disfluencies ("filler DEPODs"). Filler DEPODs are defined in SimpleMDEV5.0 Section 2. For RT-03F evaluation, the detection task requires the system to specify whether each token is part of a filler DEPOD or not.

[9] SUs have been variously defined as "slash units", "sentence units", "semantic units" and "structural units".

Filler tokens may occur within an edit DEPOD (SimpleMDEV5.0 Section 3.7.2). For the purposes of the Filler Word Detection task, the fact that the filler token is within an edit DEPOD is irrelevant, i.e. such tokens should be detected as part of this task.

Section 2 of SimpleMDEV5.0 defines four subtypes of fillers (filled-pauses, discourse markers, explicit editing terms, asides/parentheticals). For the purposes of the Filler Word Detection task, fillers of subtype "aside/parenthetical" should not be detected, i.e. the current task is limited to detection of only three subtypes of fillers: filled-pauses, discourse markers, and explicit editing terms.

For RT-03F evaluation, automatic identification of filler subtype is not part of the Filler Word Detection task.

4.3.3 IP DETECTION

The IP Detection task is to produce a sequence of scoreable tokens and to detect the locations of the Interruption Points (IP). Interruption Points are defined in Footnote 1 and Section 3.2 of SimpleMDEV5.0. For RT-03F evaluation, the detection task requires the system to specify whether an IP occurs at the left edge, right edge, or both edges of each token.

A merged IP occurs at the left edge of a filler when the filler follows an edit DEPOD within the same SU (i.e. not separated by an incomplete or complete SU boundary) and when there are no intervening tokens of type "lexeme" (see Appendix A) between the edit DEPOD and the filler.

For RT-03F evaluation, automatic identification of IP subtype is not part of the IP detection task.

4.3.4 SU BOUNDARY DETECTION

The SU Boundary Detection task is to produce a sequence of scoreable tokens and to detect the tokens that are located at the end of an SU. The definition of an SU is provided in SimpleMDEV5.0 Section 4. For RT-03F evaluation, the detection task requires the system to specify whether each token is located at the end of an SU or not.

For RT-03F evaluation, automatic identification of the subtype of SU that has ended is not part of the SU Boundary Detection task.

4.3.5 SPEAKER ATTRIBUTED STT

The Speaker Attributed STT task is to produce a sequence of scoreable tokens and to identify the speaker for each token. By "identify", we mean that the system must make an N-way decision as to the identity of the speaker of each token. "N" is the total number of speakers within a single input audio signal. "N" is not known to the system. All tokens spoken by the same speaker should be given the same, but arbitrary, speaker identification label. Tokens spoken by different speakers should be assigned different speaker identification labels.

For RT-03F evaluation, automatic identification of the proper name of the speaker is not part of the Speaker Attributed STT task.

4.3.6 RT03 RICH TRANSCRIPTION

The RT03 Rich Transcription task is to produce a sequence of scoreable tokens and for each token, to detect whether that token is part of an edit DEPOD, whether it is part of a filler DEPOD, whether a single IP occurs at the left, right, or both edges of the token, whether it is located at the end of an SU, and to identify the speaker of the token.

5 PERFORMANCE MEASURES AND TOOLS

Separate performance measures are defined for each of the major EARS tasks. The number of errors for each task are accumulated over all of the files and channels then normalized into one average. RT-03F supports three performance measurement tools: s*u-eval* and *df-eval* accept input in RTTM format (see Appendix A) and *rteval* accepts either RT XML (see Appendix B) or RTTM formats. Participants may choose either format for their system to output.

The filler word detection, edit word detection, and IP detection tasks are scored with either *df-eval* or *rteval*. SU boundary detection is scored with either *su-eval* or *rteval*. The speaker attributed STT and RT-03 rich transcription tasks are scored with *rteval*.

The residual of this section describes the scoring approaches for the three tools. While the evaluation concepts, (scoreable lexical tokens, un-evaluable regions as defined by the scoring UEMs), are similar between the tools, the actual error metric formulas are different.

5.1 SU-EVAL AND DF-EVAL METRICS FOR METADATA

The *su-eval* and *df-eval* evaluation tools follow a similar approach to metadata alignment and scoring. First, the metadata are mapped via a branch-and-bound optimization algorithm. Second the mapped metadata are scored.

The tools support metadata mapping in four different ways based on two independent options: the times of the system metadata can be aligned, (or warped), to coincide with the reference metadata times and the optimization function to map metadata can use metadata's time, or the aligned word texts.

System Metadata Time Alignment

The time-register between the system and reference may be off. To correct for this, the system and reference metadata times are warped by first aligning the system and reference words using a brand-and-bound algorithm to minimize the word-edit distances, and then warping the system metadata and system word times to match the reference times.

Metadata Mapping Optimization Function

The system and reference metadata are mapped using a branch-and-bound algorithm. There is a choice of two optimization functions: time overlap of metadata, or aligned word overlap. Time overlap optimization maximizes the amount of time shared by system and reference metadata. The aligned word optimization maximizes the number of aligned words between the system and reference metadata.

Default *su-eval* and *df-eval* command line options:

The default command line options for evaluating all four of the tasks below is '-w -W -t 1.00'. These options make the evaluation tools align the metadata times and use word identity mapping optimization function.

5.1.1 MEASURE FOR EDIT WORD DETECTION

An overall edit word detection error score will be computed as the average number of misclassified reference tokens per reference edit token.

$$Error_{depod} = \frac{\left(\begin{array}{l}\#\text{ of depod ref tokens not covered by sys edit depods} \\ +\#\text{ of non - depod ref tokens covered by sys edit depods}\end{array}\right)}{\#\text{ of ref tokens in the ref edit depods}}$$

A token is "covered" by a DEPOD if the midpoint (i.e., the average of beg time and end time) of the token falls within the DEPOD time interval.

5.1.2 MEASURE FOR FILLER WORD DETECTION

An overall filler word detection error score will be computed as the average number of misclassified reference tokens per reference filler token.

$$Error_{depod} = \frac{\left(\begin{array}{l}\#\text{ of depod ref tokens not covered by sys filler depods} \\ +\#\text{ of non - depod ref tokens covered by sys filler depods}\end{array}\right)}{\#\text{ of ref tokens in the ref filler depods}}$$

A token is "covered" by a DEPOD if the midpoint (i.e., the average of beg time and end time) of the token falls within the DEPOD time interval.

5.1.3 MEASURE FOR IP DETECTION

The overall IP error rate will be simply the average number of missed IP detections and falsely detected IPs per reference IP:

$$Error_{IP} = \frac{\left(\#\text{ of missed IP's} + \#\text{ of false alarm IP's}\right)}{\#\text{ of ref IP's}}$$

5.1.4 MEASURE FOR SU BOUNDARY DETECTION TASK

An overall SU error score will be computed as the average number of missed SU end point detections and falsely detected SU end points per reference SU:

$$Error_{SU} = \frac{\left(\begin{array}{l}\#\text{ of missed SU end points} + \\ \#\text{ of false alarm SU end points}\end{array}\right)}{\#\text{ of ref SU end points}}$$

5.2 *RTEVAL* METRICS FOR METADATA AND RICH TRANSCRIPTION IN RT03F

The *rteval* evaluation procedure consists of three stages. First, the scoreable token sequences from the reference and system output are aligned using Dynamic Programming to compute the minimum Edit Distance[10] (Levenshtein Distance[11]) between the two token sequences. This alignment is fixed for the remainder of the *rteval* evaluation procedure.

Second, Linear Programming[12] is used to solve a Bipartite Graph Matching problem to compute a minimum error mapping of the system speaker labels to the reference speaker labels. After the optimal mapping is determined, the speaker labels on the system output tokens are changed into their mapped reference equivalents.

In the third stage, a Slot Error Rate[13] (SER) is calculated for each of the five RT-03F Metadata tasks and an overall Token Error Rate (TER) is calculated for the comprehensive Rich Transcription task defined for RT03F.

All *rteval* metrics are defined in terms of the alignment produced in the first stage. This common alignment operates on the set of scoreable tokens as defined in Section 4.1. In computing the Edit Distance between the reference and system output token sequences, tokens are considered matched if their un-cased orthographic representations are the same. In addition, tokens of type *filled-pause* and *fragment* are matched based on their type only without regard to their orthography. Any of the four scoreable token types are allowed to align to any other type.

While the common alignment is governed principally by the token orthography and type, metadata (expressed as token subtypes or attributes) also exerts an influence upon the alignment whenever the orthographies differ between the tokens being compared. In other words, metadata is not permitted to dislodge a token from an alignment that results in an orthographic or type match, but wherever the orthographies or types are mismatched, the alignment is optimized jointly for all the metadata. This is implemented as a simple table of substitution weights used in computing the Edit Distance.

For calibration purposes, *rteval* computes a Word Error Rate for the common alignment based upon the orthographic and type matches only, disregarding the metadata attributes. We call this the "RT1" condition. RT1 measures the common baseline error from which all of the Metadata tasks begin.

For additional calibration purposes, *rteval* also computes the STT WER as defined in the RT Spring 2003 Evaluation Plan, v4, dated 2/25/03. This requires a separate alignment in *rteval* based on the STT definitions of scoreable tokens and their specific defined behavior and constraints under the alignment operation. *Rteval* runs in a dedicated mode to compute the STT WER.

For each of the Metadata tasks in the next five subsections, the Slot Error Rate has the following form:

$$SER = \frac{100 * \left(\#\text{ sub} + \#\text{ del} + \#\text{ ins}\right)}{\#\text{ ref slots}}$$

The SER metric for metadata is computed over the common *rteval* alignment by comparing only the particular metadata attributes specified below for the particular task. In other words, for the SER metric, metadata attributes are matched without regard to token orthography or type.

[10] M. Atallah, Editor, "*Algorithms and Theory of Computation Handbook*", CRC Press LLC, 1999. pp. 13-5.

[11] V. I. Levenshtein: "*Binary Codes Capable of Correcting Deletions, Insertions and Reversals*", in Soviet Physics Doklady, Vol. 10, Nr. 8, Feb. 1966, pp. 707 – 710.

[12] Christos H. Papadimitriou, Kenneth Steiglitz, "*Combinatorial Optimization: Algorithms and Complexity*", Prentice-Hall Publishers, 1982, Chapter 11, Section 11.2.

[13] John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel, "*Performance Measures for Information Extraction*", Proceedings of the DARPA Broadcast News Workshop, Herndon, Virginia, Morgan Kaufmann Publishers, Feb. 28-March 3, 1999, pp. 249-252.

5.2.1 FILLER WORD DETECTION

A Filler token is indicated by a non-null RT XML *filler_type* attribute. In RTTM, a Filler token is indicated by being located within the time span of an MDE FILLER object.

sub = 0

del = number of reference Filler tokens that fail to align to system Filler tokens

ins = number of system Filler tokens that fail to align to reference Filler tokens

ref slots = number of reference Filler tokens

5.2.2 EDIT WORD DETECTION

An Edit token is indicated by a non-null RT XML *edit_type* attribute. In RTTM, an Edit token is indicated by being located within the time span of an MDE EDIT object.

sub = 0

del = number of reference Edit tokens that fail to align to system Edit tokens

ins = number of system Edit tokens that fail to align to reference Edit tokens

ref slots = number of reference Edit tokens

5.2.3 IP DETECTION

IP edges are indicated by non-null RT XML *ip_label* attributes, which can assume eight values – "1L", "1R", "2L", "2R", "1L1R", "2L1R", "1L2R", and "2L2R". "1L" denotes a single IP at the left edge of the (filler) token. "2L" denotes a merged IP at the left edge of the (filler) token. By convention, a "2R" must occur at the right edge of the preceding (edit) token. The other *ip_label* values are interpreted similarly. Therefore, *ip_labels* with both L and R components indicate that IPs occur at both edges of the token.

As noted, merged IPs are indicated on both edges of the adjacent tokens but the two labels count for only one IP slot. Hence, the total number of IP slots is equal to the number of 1's in the *ip_label*s plus half the number of 2's. During scoring, a single IP can match either a single IP or a merged IP that occurs on the same edge of the aligned tokens.

In RTTM, an IP slot is indicated by an MDE IP object at the location of the IP. Furthermore, a single IP object is used to denote either a single or merged IP.

sub = 0

del = number of reference IP slots that fail to align to system IP slots

ins = number of system IP slots that fail to align to reference IP slots

ref slots = number of reference IP slots

5.2.4 SU BOUNDARY DETECTION

An SU Boundary token is indicated by a RT XML *sentence_boundary* attribute with a value of "end" or "begend". In RTTM, an SU Boundary token is indicated by being located at the end of the time span of an MDE SU object.

sub = 0

del = number of reference SU Boundary tokens that fail to align to system SU Boundary tokens

ins = number of system SU Boundary tokens that fail to align to reference SU Boundary tokens

ref slots = number of reference SU Boundary tokens

5.2.5 SPEAKER ATTRIBUTED STT

A Speaker Attributed token is indicated by a RT XML *token_type* attribute with a value of "lexeme", "fragment", or "foreign".

sub = number of reference Speaker Attributed tokens that align to system Speaker Attributed tokens with non-matching mapped *speaker_id* attributes

del = number of reference Speaker Attributed tokens that fail to align to any token (due to RT1 deletion)

ins = number of system Speaker Attributed tokens that fail to align to any token (due to RT1 insertion)

ref slots = number of reference Speaker Attributed tokens

5.2.6 RT03 RICH TRANSCRIPTION

A Rich Transcription token is indicated by a RT XML *token_type* attribute with a value of "lexeme", "fragment", or "foreign".

RT-03 Rich Transcription is evaluated using Token Error Rate, which has the following form:

$$TER = \frac{100 * \left(\#\,\text{sub} + \#\,\text{del} + \#\,\text{ins}\right)}{\#\,\text{ref tokens}}$$

sub = number of reference tokens aligned to system tokens for which any of the following is true:

- the reference *token_symbol* attribute does not match the system *token_symbol* attribute
- the reference token is a Filler token and the system token is not (or vice versa)
- the reference token is an Edit token and the system token is not (or vice versa)
- the reference token is adjacent to one or two IPs and the system token is not (or vice versa)
- the reference token is an SU Boundary token and the system token is not (or vice versa)
- the reference *speaker_id* attribute does not match the mapped system *speaker_id* attribute

del = number of reference Rich Transcription tokens for which there is no corresponding system Rich Transcription token (due to RT1 deletion)

ins = number of system Rich Transcription tokens for which there is no corresponding reference Rich Transcription token (due to RT1 insertion)

ref tokens = number of reference Rich Transcription tokens

5.3 EVALUATION UN-PARTITIONED EVALUATIONS MAPS (UEM)

Un-partitioned evaluation maps (UEM)s are the mechanism the evaluation infrastructure uses to specify time regions within an audio recording.

There are several uses for UEM files within the evaluation infrastructure. The UEM file structure and UEM usages are defined as follows.

5.3.1 UEM FILE STRUCTURE

The UEM file format is a concatenation of time mark records for a segment of audio in a speech waveform. The records are separated with a newline. Each word token must have a file id, channel identifier [1 | 2], begin time, and end time. Each record follows this BNF format:

UEM :== <F><SP><C><SP><BT><SP><ET>

where,

<SP> indicates a space (" ").

<F> indicates the path, filename, and extension of the waveform to be processed.

<C> indicates the waveform channel can have a value of "1" or "2".

<BT> indicates the beginning time of the segment measured in seconds from the beginning of the file which is time 0.

<ET> indicates the ending time of the segment measured in seconds from the beginning of the file which is time 0.

For example:

audio/dev/english/cts/sw_47620.sph 1 0 291.34
audio/dev/english/cts/sw_47621.sph 1 0 301.98

….

5.3.2 SYSTEM INPUT UEM FILES

A UEM file is provided within the evaluation data to define the regions of the audio that the system must process.

5.3.3 RT-03F MDE SCORING UEM FILES

A UEM file is provided within the evaluation data to define regions within the audio recording for which the system is to be evaluated. Evaluation systems may not use this file except to score their system output.

The MDE scoring UEM file is designed for scoring the six RT-03f MDE evaluation tasks. The following conditions cause regions to be excluded from evaluable regions:

1. Untranscribed segments of speech: These non-transcribed regions include commercials, reporter chit-chat outside the context of a story, station identifications, promotions for upcoming broadcasts, public-service announcements, and long musical interludes.
2. Overlapping speech: These regions of time include speech from multiple speakers in the same channel.
3. Unannotated SUs: SUs can have the type "unannotated" if the LDC was unable to perform SU annotation on that stretch of speech.
4. Unannotated Metadata Regions: Any region within marked within the transcript with the NO_RT_METADATA annotation applied.
5. Empty Segments: Any SEGMENT for which the speaker is "unknown". (Note that the fundamental reference data is missing for these segments, so that they are effectively non-transcribed).

5.3.4 SPEAKER DIARIZATION SCORING UEM FILES

There is a specifically created UEM file for scoring the speaker diarization evaluation. Its existence is noted for completeness.

6 CORPORA RESOURCES

6.1 EVALUATION TRAINING DATA

Any released broadcast news or conversational telephone speech (CTS) LDC corpora may be used for the training and development of the MDE tasks. Note that **all** material used in **any** way for training and development for the broadcast news recognition tasks must predate the test epoch (February 2001) as specified in Section 8.1.2.

The LDC has begun annotating data to the Simplified MDE Annotation Specification. Consult the MACEARS web site, **http://ears.ll.mit.edu/,** for currently released data.

6.2 DRY RUN EVALUATION/DEVELOPMENT TEST DATA

The dry run test corpora will consist of the RT-03 set2 data. The Broadcast News files are:

20010206_1830_1900_ABC_WNT
20010221_1830_1900_NBC_NNW
20010225_0900_0930_CNN_HDL

The Conversational Telephone Speech files are:

fsh_60386	fsh_60398	fsh_60441	fsh_60477	fsh_60568
fsh_60613	fsh_60668	fsh_60682	fsh_60784	fsh_60817
fsh_60818	fsh_60844	fsh_60874	fsh_61113	fsh_61130
fsh_61148	fsh_61225	fsh_61228	sw_45104	sw_45237
sw_45481	sw_45626	sw_45837	sw_45856	sw_45973
sw_46028	sw_46168	sw_46455	sw_46565	sw_46671
sw_46732	sw_46938	sw_47038	sw_47073	sw_47175
sw_47282				

6.3 RT-03 FALL EVALUATION TESTING DATA

The RT-03 Fall evaluation test corpora will come from the RT-03-set1 data set. The Broadcast News files are:

20010217_1000_1030_VOA_ENG
20010220_2000_2100_PRI_TWD
20010228_2100_2200_MNB_NBW

The Conversational Telephone Speech files are:

fsh_60262	fsh_60354	fsh_60416	fsh_60463	fsh_60493
fsh_60549	fsh_60571	fsh_60593	fsh_60627	fsh_60648
fsh_60650	fsh_60720	fsh_60732	fsh_60797	fsh_60862
fsh_60885	fsh_61039	fsh_61192	sw_45097	sw_45142
sw_45355	sw_45454	sw_45586	sw_45654	sw_45713
sw_45727	sw_45819	sw_46140	sw_46412	sw_46512
sw_46615	sw_46677	sw_46789	sw_46868	sw_47346
sw_47411				

The RT-03 Fall evaluation is not a "blind" evaluation. Participants have had access to transcribed data since the RT-03

Spring evaluation. Further, the now-designated evaluation set was intended to be the development test set. As such, participants began working in earnest developing systems until the community realized the wrong data set was annotated for the development test set. Rather than delay the Fall evaluation, the development and evaluation data sets were swapped and participants were instructed to discontinue work with the evaluation data set. The following rules governing the use of the RT-03 data were instituted on July 7, 2003 per the EARS Executive Board.

Individual researchers who are likely to participate in the RT-03F evaluation should only use the dev set (RT-03 set 2, see Section 6.2) and should cease using evaluation data set (RT-03 set 1). Participants must disclose in their system description how they used the Fall evaluation data set (RT-03 set 1), if at all, prior to the July 7th decision to swap test sets. The disclosure is intended to publicly acknowledge usage of the test set. Researchers who are not intending to participate in RT-03F can use sets 1 and 2 as they see fit.

7 EVALUATION CONDITIONS

There are many different conditions under which system performance may be evaluated. This section identifies those conditions for which performance will be computed and, of those, which are to be designated as the required evaluation conditions.

The following list of evaluation conditions apply to all six RT-03 Fall Evaluation tasks.

Language:

- English only

Domain:

- Broadcast news and conversational telephone speech. Participants may build systems to address either or both domains.

Input:

- Speech input. Any desired fully-automatic signal processing approaches may be employed (including the use of a site developed STT system). This is the required evaluation condition for Input.
- Speech plus the reference transcriptions: The function of this evaluation condition is to serve as a perfect-STT control condition. It is an optional contrast evaluation condition. The system inputs will be RTTM formatted files derived from the reference RTTM files and placed in the 'input' directory (described in section 8.2 below) of the evaluation corpus. The derived RTTM files will contain only 'LEXEME' RTTM records with the speaker's identity expunged, (replaced by <NA>), and the LEXEME subtypes 'alpha', 'acronym', 'interjection', 'propername', and 'other' mapped into the 'lex' subtype.

8 PARTICIPATION INSTRUCTIONS

Participation is encouraged for all those who are interested in one or more of the RT-03 Fall tasks. All participants must, however, agree to completely process all of the data for at least one task. This means that, at a minimum, the speech-input-only processing condition must be implemented. Participants have the freedom to implement systems for either or both domains, Broadcast News or Conversational Telephone Speech.

As a condition of participation, all sites must agree to make their submissions (system output, system description, and ancillary files) available for experimental use by other research sites. Further, submission of system output to NIST constitutes permission on the part of the site for NIST to publish scores and analyses for that data including explicit identification of the submitting site and system.

8.1 PROCESSING RULES

8.1.1 RULES THAT APPLY TO ALL EVALUATIONS

All developed systems must be fully automatic requiring no manual intervention to influence the system's decision-making infrastructure when generating the system output. Manual intervention is allowed to shepherd system processes but not to change any parameter settings or processing steps in response to knowledge or intuition gained from processing the evaluation data.

The only exemption from the automatic processing restriction is for the reference text condition. Participants who use the reference text condition can manually add pronunciations to their dictionaries to enable forced alignment of the out-of-vocabulary terms. Participants cannot use the lexical knowledge gained from the reference+speech-input system to modify their speech-input only system.

Systems will be provided with recorded SPHERE formatted waveform files and a UEM file specifying the speech files and regions within them to be processed. Each conversational telephone speech test waveform will be provided in 2-channel files, and both channels must be processed. Broadcast news speech test data will be presented in single channel files, one per broadcast.

While entire broadcast and conversation files will be distributed, only the material specified in the UEM test index file for the experiment to be run is to be processed. Material outside of the times specified in the UEM test index file is not to be used in any way (e.g., for adaptation).

8.1.2 ADDITIONAL RULES FOR PROCESSING BROADCAST NEWS

News-oriented material (audio, textual, etc.) generated during or after the test epoch beginning February 01, 2001 **may not be used in any way for system development or training.** Broadcast news material must be processed in the chronological order of the date/time of the original broadcast. Although automatic adaptation may be performed using previously-processed material, systems may not "look ahead" in time at later recordings. Hence, processing must be complete on a particular broadcast news test file before moving on to the next file. Any form of within-file adaptation, however, is permitted and systems may look backwards in time at previously-processed files. The show identity and original broadcast date are allowable side information that systems may use. Therefore, systems may make use of show-dependent models.

8.1.3 ADDITIONAL RULES FOR PROCESSING CONVERSATIONAL TELEPHONE SPEECH

Conversational telephone speech may be processed in any order and any form of automatic within-conversation and

cross-conversation adaptation may be employed. No side information is provided for telephone conversations (e.g., corpus collection name, recording time, etc.). No manual or automatic segmentation will be provided, although systems may make use of segmentation outputs donated from other sites.

8.2 DATA FORMATS

8.2.1 TEST DATA

For practicality, the recorded waveform files to be processed will be distributed on CD-ROM and the corresponding indices, annotations, and transcripts will be made available via the Web or FTP using an identical directory structure. After the evaluation, system outputs will be released in this structure as well.

Directory	Description
indices/	index files containing the list of files and times to be processed for particular experiments
audio/	audio files
input/<EXP-ID>/	ancillary data including reference annotations for various experiments – must be used in accordance with instructions for that experiment
output/<EXP-ID>/	system output submissions – will be made available as received for integration tests
reference/	reference transcripts, annotations, and MS-wav files for post-evaluation scoring and analyses

Note: EXP-ID specifies a unique identifier for each experiment and is defined in 8.3.1.

For clarity, the "audio/" and "reference/" directories are subdivided into <DATA>/<LANG>/<TYPE> subdirectories:

where,

<DATA> is either [dev03f|eval03f]

<LANG> [english]

<TYPE> is either [bnews|cts]

The "indices/" directory contains a set of UEM test index files specifying the waveform data to be evaluated for each EXP-ID condition supported in this evaluation as described in 8.3.1 and these files are named <EXP-ID>.uem with the special site code "expt". A separate UEM file, defined in Section 5.3, will be provided for each experiment for each supported <DATA>, <LANG>, and <TYPE>. Corresponding ancillary data for some control conditions is given in the "input/" directory under subdirectories with the same EXP-ID.

8.2.2 MDE OUTPUT FORMAT

8.2.2.1 RTTM FORMAT

See Appendix A for a description of the RTTM format. Each RTTM file corresponds to a single source file in the test.

8.2.2.2 RT XML FORMAT

See Appendix B for a description of the RT XML format. Each RT XML file corresponds to a single source file in the test.

8.2.3 SYSTEM DESCRIPTION

For each test run (for each unique EXP-ID), a brief description of the system (algorithms, data, configuration) used to produce the system output must be provided along with your system output. If multiple system runs are submitted for a particular experiment with different systems/configurations, explicitly designate one run as the primary system and the others as contrastive systems in the system description. This information is to be recorded in a file named:

<EXP-ID>.txt

(where EXP-ID is defined in Section 7.3.1)

and placed in the "output" directory alongside the similarly-named directories containing your system output. This file is to be formatted as follows:

1. EXP-ID = <EXP-ID>
2. Primary: yes | no
3. System Description:

   [brief technical description of your system; if a contrastive test, contrast with primary system description]

4. Training:

   [list of resources used for training; for STT, be sure to address acoustic and LM training, and lexicon]

5. References:

   [any pertinent references]

8.3 SUBMISSION INSTRUCTIONS

8.3.1 SUBMISSION EXPERIMENT CODES

The output of each submitted experiment must be identified by the following code as specified above.

EXP-ID =
<SITE>_<YEAR>_<TASK>_<DATA>_<LANG>_
<TYPE>_<COND >_<SYSID>_<RUN>

where,

SITE ::= expt | bbn | bbnplus | cu | elisa | clips | sri | sriplus | ibm | mitll | ms | pan | ...

(The special SITE code "expt" is used in the EXP-ID-based filename of the UEM test index files under the "indices/" directory to list the test material for a particular experiment and in the EXP-ID-based subdirectory name under the "input/" directory to indicate ancillary data to be used in certain control condition experiments.)

YEAR ::= 03f

For the RT-03 Fall Evaluation, these are:

TASK ::= ewd | fwd | ipd | subd | sastt | 03rt | data

where,

ewd = edit word detection

fwd = filler word detection

ipd = IP detection

subd = SU boundary detection

sastt = Speaker attributed STT

03rt = RT-03 rich transcription

data = a special TASK code used to provide a directory for ancillary data such as common CTM files used over many MDE experiments. Please make sure to use increasing run numbers for this special experiment ID when making multiple submissions so that your ancillary data from earlier submissions is not over-written here at NIST

DATA ::= dev03f | eval03f

LANG ::= eng

RT-03F only includes English (eng) material.

TYPE ::= bnews | cts

CONDITION ::= spch | ref

where,

spch = audio input only

ref = audio input + reference transcript

[The "spch" (speech) condition is the primary condition of interest. The "ref" (reference) condition is provided as a control for perfect speech recognition and includes both the speech and reference transcript as input. The MDE tasks for this condition may make use of only the LEXEME entries in the supplied RTTM as defined in Section 7 "Evaluation Conditions".]

SYSID ::= site-named string designating the system used

[This is intended so that we can differentiate between contrastive runs for the same condition. Therefore, a different SYSID should be created for runs where any manual changes were made to a particular system]

RUN ::= 1..n (with values greater than 1 indicating multiple runs of the same experiment/system)

[An incremental run number MUST be used for multiple submissions of any particular experiment with an identical configuration (due to a bug or runtime problem.) This should NOT be used to indicate contrastive runs. Instead, a different SYSID should be used. However, please note that ONLY the first run will be considered "official" and will be scored by NIST unless special arrangements are made with NIST. Please also note that submissions which reuse identical experiment IDs/run numbers from previous submissions will be automatically rejected.]

For examples:

bbn_03f_ip_eval03f_eng_cts_spch_superreco1_1

sri_03f_sastt_eval03f_eng_bnews_ref_speakerid2_1

8.3.2 SUBMISSION DIRECTORY STRUCTURE

All system output submissions must be formatted according to the following directory structure:

output/<SYSTEM-DESCRIPTION-FILES>

output/<EXP-ID>/ <OUTPUT-FILES>

where,

<SYSTEM-DESCRIPTION-FILES> one per <EXP-ID> as specified in 8.2.3

<EXP-ID> is as defined in Section 7.3.1

<OUTPUT-FILES> are as defined in Section 7.2

Note: one output file must be generated for EACH input file as specified in the test index for the experiment being run. The output files are to be named so as to be identical to the input file basenames with the appropriate .ctm or .rttm filetype extension. For example, an STT output file for the speech waveform file sw_47620.sph must be named sw_47620.ctm and an MDE output file must be named sw_47620.rttm or sw_47620.xml for RTTM and RT XML formats respectively. When generated, these output files are to be placed under the appropriately-named EXP-ID directory on your system identifying the experiment run.

8.3.3 SUBMISSION PACKAGING AND UPLOADING

To prepare your submission, first create the previously-described file/directory structure. This structure may contain the output of multiple experiments, although you are free to submit one experiment at a time if you like. The following instructions assume that you are using the UNIX operating system. If you do not have access to UNIX utilities or ftp, please contact NIST to make alternate arrangements.

First change directory to the parent directory of your "output/" directory. Next, type the following command:

tar -cvf - ./output | gzip > <SITE>_<SUB-NUM>.tgz
where,

<SITE> is the ID for your site as given in Section 7.3.1

<SUB-NUM> is an integer 1 – n where 1 identifies your first submission, 2 your second, and so forth.

This command creates a single tar file containing all of your results. Next, ftp to jaguar.ncsl.nist.gov giving the username 'anonymous' and your e-mail address as the password. After you are logged in, issue the following set of commands, (the prompt will be 'ftp>'):

ftp> cd incoming
ftp> binary
ftp> put <SITE>_<SUB-NUM>.tgz
ftp> quit

You've now submitted your recognition results to NIST. The last thing you need to do is send an e-mail message to Audrey Le at audrey.le@nist.gov to notify NIST of your submission. The following information should be included in your email:

1) The name of your submission file

2) A listing of each of your submitted experiment IDs

3) e.g.,

```
Submission: bbnplus_1 <NL>
Experiments: <NL>
bbnplus_03f_subd_eval03f_eng_cts_spch
_superreco1_1<NL>
bbnplus_03f_ipd_eval03f_eng_cts_spch_
superreco2_1 <NL>
```

Note that submissions received after the stated due dates FOR ANY REASON will be marked late. So, please submit your files in time for us to deal with any transmission/formatting problems that might occur -- well before the due date if possible.

9 SCHEDULE

The evaluation schedule below is accurate as of September 17, 2003. Please consult the live version of the schedule at http://macears.ll.mit.edu/macears_docs/ears-schedule.txt for any late-breaking changes.

1 Oct - NIST releases eval data to sites

20 Oct - System output due at NIST

22 Oct - NIST releases scored results

5 Nov - Slides for notebooks due

13-14 Nov - RT03F Workshop

Please note that the stated dates are hard deadlines. All late submissions will be marked as such and given the tight schedule, severely late submissions may not be scored at all prior to the workshops.

10 WORKSHOPS

The evaluation will be followed by the Rich Transcription 2003 Fall (RT-03F) Workshop. The workshop is open to all participants. Information regarding workshop logistics and registration will be posted at a later date in email.

Appendix A: RTTM File Format Specification

This description has been excerpted from RTTM-format-v12.doc[14]. There are four general object categories to be represented. They are STT objects, MDE objects, source (speaker) objects, and structural objects.[15] Each of these general categories may be represented by one or more types and subtypes, as shown in table 1.

Table 1 Rich Text object types and subtypes

Type	Subtypes
Structural types:	
SEGMENT	**eval**, or (none)
NOSCORE	(none)
NO_RT_METADATA	(none)
STT types:	
LEXEME	**lex**, **fp**, **frag**, **un-lex**[16], **for-lex**, **alpha**[17], **acronym**[17], **interjection**[17], **propername**[17], and **other**
NON-LEX	**laugh**, **breath**, **lip-smack**, **cough**, **sneeze**, and **other**
NON-SPEECH	**noise**, **music**, and **other**
MDE types:	
FILLER	**filled_pause**, **discourse_marker**, **explicit_editing_term**, and **other**
EDIT	**repetition**, **restart**, **revision**, **simple**, **complex**, and **other**
IP	**edit**, **filler**, **edit&filler**, and **other**
SU	**statement**, **backchannel**, **question**, **incomplete**, **unannotated**, and **other**
CB	**coordinating**, **clausal**, and **other**
A/P	(none)
SPEAKER	(none)
Source information:	
SPKR-INFO	**adult_male**, **adult_female**, **child**, and **unknown**

The STT, MDE and Source information objects are potential research target. And, except for the static speaker information object [**SPKR-INFO**], each object exhibits a temporal extent with a beginning time and a duration. (The duration of interruption points [**IP**] and clausal boundaries [**CB**] is zero by definition.)

These objects are represented individually, one object per record, using a flat record format with object attributes stored in white-space separated fields. The format is shown in table 2.

[14] The latest RTTM format document can be found at the URL 'http://www.nist.gov/speech/tests/rt/rt2003/fall/index.htm'.

[15] Structural objects are important because they are produced by LDC to provide a modicum of temporal organization in the annotation and identify non-evaluable regions.

[16] Un-lex is also used to tag words that are infected with or affected by laughter.

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

Table 2 Object record format for EARS objects

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

where

`file` is the waveform file base name (i.e., without path names or extensions).

`chnl` is the waveform channel (e.g., "**1**" or "**2**").

`tbeg` is the beginning time of the object, in seconds, measured from the start time of the file.[18] If there is no beginning time, use `tbeg` = "**<NA>**".

`tdur` is the duration of the object, in seconds.[4] If there is no duration, use `tdur` = "**<NA>**".

`stype` is the subtype of the object. If there is no subtype, use `stype` = "**<NA>**".

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

`name` is 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>**".

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

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

Table 3 Format specialization for specific object types

Field 1	2	3	4	5	6	7	8	9
Type	*file*	*chnl*	*tbeg*	*tdur*	*ortho*	*stype*	*name*	*conf*
SEGMENT	file	chnl	tbeg	tdur	**<NA>**	**eval** or **<NA>**	name or **<NA>**	conf or **<NA>**
NOSCORE	file	chnl	tbeg	tdur	**<NA>**	**<NA>**	**<NA>**	**<NA>**
NO_RT_METADATA	file	chnl	tbeg	tdur	**<NA>**	**<NA>**	**<NA>**	**<NA>**
LEXEME **NON-LEX**	file	chnl	tbeg	tdur	ortho or **<NA>**	stype	name	conf or **<NA>**
NON-SPEECH	file	chnl	tbeg	tdur	**<NA>**	stype	**<NA>**	conf or **<NA>**
FILLER **EDIT** **SU**	file	chnl	tbeg	tdur	**<NA>**	stype	name	conf or **<NA>**
IP **CB**	file	chnl	tbeg	**<NA>**	**<NA>**	stype	name	conf or **<NA>**
A/P **SPEAKER**	file	chnl	tbeg	tdur	**<NA>**	**<NA>**	name	conf or **<NA>**
SPKR-INFO	file	chnl	**<NA>**	**<NA>**	**<NA>**	stype	name	conf or **<NA>**

[18] 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 B: RT XML File Format Specification

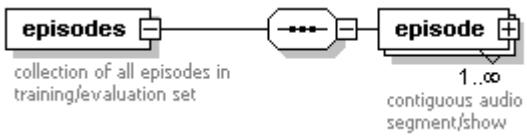
RT XML SCHEMA DEFINITION – VERSION 2.3

Schema: **rtxml_v2.3.xsd**
Schema Location: http://www.speech.bbn.com/ears/rtxml_v2.3.xsd

Elements
episodes

element **episodes**

diagram

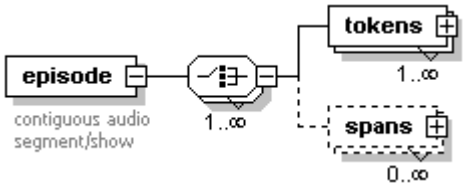


children **episode**

annotation documentation collection of all episodes in training/evaluation set

element **episodes/episode**

diagram



children **tokens spans**

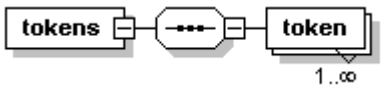
attributes

Name	Type	Use	Enumerations
episode_name	xs:string	required	
media_file	xs:string	optional	

annotation documentation contiguous audio segment/show

element **episodes/episode/tokens**

diagram



children **token**

attributes

Name	Type	Use	Enumerations
token_sequence_index	xs:integer	required	
token_sequence_type	xs:string	optional	

element **episodes/episode/tokens/token**

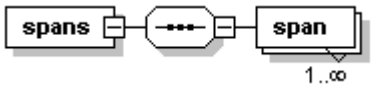
diagram

attributes

Name	Type	Use	Enumerations
token_index	xs:integer	required	
token_symbol	xs:string	required	
token_type	xs:string	required	lexeme fragment nonlexeme environment foreign
start_time	xs:decimal	optional	
duration	xs:decimal	optional	
confidence	xs:decimal	optional	
nonlexeme_type	xs:string	optional	laugh breath lipsmack cough sneeze other
environment_type	xs:string	optional	noise music other
speaker_boundary	xs:string	optional	beg end begend
speaker_id	xs:string	optional	
speaker_type	xs:string	optional	male female child other
sentence_boundary	xs:string	optional	beg end begend
sentence_type	xs:string	optional	statement question incomplete backchannel other
filler_boundary	xs:string	optional	beg end begend
filler_type	xs:string	optional	filled_pause discourse_marker explicit_editing_term other
edit_boundary	xs:string	optional	beg end begend
edit_type	xs:string	optional	repetition revision restart simple complex other
ip_label	xs:string	optional	1R 1L 2L 2R 1L1R 2L1R 1L2R 2L2R
channel_id	xs:string	optional	

element **episodes/episode/spans**

diagram



children **span**

attributes

Name	Type	Use	Enumerations
span_sequence_index	xs:integer	required	
token_sequence_idref	xs:integer	required	

element **episodes/episode/spans/span**

diagram

attributes

Name	Type	Use	Enumerations
index	xs:integer	required	
start_idref	xs:integer	required	
end_idref	xs:integer	required	
type	xs:string	required	segment sentence speaker filler edit
subtype	xs:string	optional	
start_time	xs:decimal	optional	
duration	xs:decimal	optional	
label	xs:string	optional	
confidence	xs:decimal	optional	

XML Schema documentation generated with **XMLSPY** Schema Editor **http://www.altova.com/xmlspy**

Appendix C: GLM Processing

Prior to scoring, both the reference and system output token strings will be transformed using a global map file (GLM). The GLM is intended to ensure that reference and hypothesis tokens which do not differ semantically are scored as correct. This is accomplished by transforming the token strings in both the reference and system output via a set of mapping rules. The GLM applies a set of rules to the system output which expands contractions to all possible expanded forms.

Note that GLM processing may result in the generation of several alternative token strings in the system output. It may also result in token strings being split into two or more strings. For example, contractions are mapped to their expanded form and compound words are split into their constituents. After GLM filtering, hyphens in both the system output and reference are transformed into token separators.