SParseval: Evaluation Metrics for Parsing Speech

Brian Roark[a], Mary Harper[b,c], Eugene Charniak[d], Bonnie Dorr[c], Mark Johnson[d], Jeremy G. Kahn[e], Yang Liu[f,g], Mari Ostendorf[e], John Hale[h], Anna Krasnyanskaya[i], Matthew Lease[d], Izhak Shafran[j], Matthew Snover[c], Robin Stewart[k], Lisa Yung[j]

[a]Oregon Health & Science University; [b]Purdue University; [c]University of Maryland; [d]Brown University; [e]University of Washington; [f]ICSI, Berkeley; [g]University of Texas at Dallas; [h]Michigan State; [i]UCLA; [j]Johns Hopkins University; [k]Williams College

Abstract

While both spoken and written language processing stand to benefit from parsing, the standard Parseval metrics (Black et al., 1991) and their canonical implementation (Sekine and Collins, 1997) are only useful for text. The Parseval metrics are undefined when the words input to the parser do not match the words in the gold standard parse tree exactly, and word errors are unavoidable with automatic speech recognition (ASR) systems. To fill this gap, we have developed a publicly available tool for scoring parses that implements a variety of metrics which can handle mismatches in words and segmentations, including: alignment-based bracket evaluation, alignment-based dependency evaluation, and a dependency evaluation that does not require alignment. We describe the different metrics, how to use the tool, and the outcome of an extensive set of experiments on the sensitivity of the metrics.

1. Motivation for SParseval

Natural language parsing technology was originally evaluated on textual corpora (Marcus et al., 1993), for which the punctuated sentences matched the tokens in the yields of the gold-standard parse trees. Under these conditions it is appropriate to perform sentence-level parse scoring (Sekine and Collins, 1997; Black et al., 1991). However, parsers are now being applied in spoken domains such as Switchboard conversational telephone speech (CTS) (Godfrey et al., 1992), for which words are recognized and sentence boundaries detected by fully automated systems. Although parsers have been evaluated on Switchboard, they initially were applied to gold-standard transcripts, with either manual (Charniak and Johnson, 2001) or automatic (Kahn et al., 2004) sentence segmentations.

As the NLP and speech processing communities are converging to work on spoken language processing, parsing techniques are now being applied to automatic speech recognition (ASR) output with both automatic (errorful) transcripts and automatic sentence segmentations. This creates the need to develop and evaluate new methods for determining spoken parse accuracy that support evaluation when the yields of gold-standard parse trees differ from parser output due to both transcription errors (wrong words) and sentence segmentation errors (wrong boundaries).

This paper describes the SParseval scoring tool[1] that was developed by the *Parsing and Spoken Structural Event Detection* team at the 2005 CLSP Johns Hopkins Summer Workshop in order to evaluate spoken language parsing performance. The tool builds on the insights from the parsing metrics literature (e.g., Carroll (ed.) (1998), Carroll et al. (2002), Sekine and Collins (1997), and Black et al. (1991)), and implements both a bracket scoring procedure similar to Parseval and a head-dependency scoring procedure that evaluates matches of `(dependent word, relation, head word)`. The latter procedure maps each tree to a dependency graph and then evaluates precision and recall on the edges of the graph.

To illustrate why a new approach is needed, consider the example in Figure 1, in which the first line above the alignment file represents the gold-standard transcription and sentence segmentation for a span of speech (segmentation boundaries marked as ||). The second line represents the errorful ASR system output that the parser would be given to produce parses, containing words produced by a speech recognizer and the sentence segmentations provided by an automatic system. An alignment for these two spans is depicted in the box. Given the fact that the words and sentences do not directly line up, it is difficult to score the test parses against the gold parses on a sentence-by-sentence basis. The word insertions and deletions resulting from ASR errors, together with different sentence segmentations, make the span-based measures proposed in Black et al. (1991) difficult to apply. However scoring can proceed if we create a super tree for the gold and test inputs over an entire speech transcript chunk (e.g., a conversation side) as in Kahn et al. (2004), so that the parse relations produced by the parser on test input can be compared to the gold relations to obtain recall, precision, and F-measure scores. Alignments are used to establish comparable constituent spans for labeled bracketing scoring.

In Section 2, we describe the tool and illustrate its use for scoring parses under a variety of conditions. Section 3 summarizes results of a set of experiments on the sensitivity of the metrics when parsing speech transcripts.

2. SParseval

2.1. Overview

The SParseval tool was implemented in C and was designed to support both speech-based bracket and head dependency scoring at the level of a demarcated chunk of speech such as a conversation side. It also supports more traditional text-based scoring methods that require the input to the parser to match perfectly in words and sentence segments to the gold standard. To calculate the bracket scores

```
Gold:  I like Baltimore         in August ||  I think everyone does     ||  do n't  you
Test:  I like baldies more || in August      I think very   wonton   's      donut  you
```

Alignment File:

```
I           I         000    // 000 is a match
like        like      000
Baltimore   baldies   001    // 001 is a substitution
            more      010    // 010 is an insertion
in          in        000
August      August    000
I           I         000
think       think     000
everyone    very      001
does        wonton    001
            's        010
do          donut     001
n't                   100    // 100 is a deletion
you         you       000
```

Figure 1: An example of the alignment of a gold-standard transcript with segmentation to a system-produced transcript with segmentation that illustrates the concepts of match, substitution, insertion, and deletion.

[1]http://www.clsp.jhu.edu/ws2005/groups/eventdetect/files/SParseval.tgz

in the face of word and segmentation errors, the tool is designed to utilize information from a word-level alignment between the yields of the test parses and gold parses in a speech transcript chunk (e.g., a conversation side or broadcast news story), as shown in Figure 1, in order to assign constituent spans for calculation of bracket matches. The tool also provides scores based on all of the head dependencies extracted from the test and gold trees, as well as a more focused set of open class dependencies, which omit closed-class function words. Dependency scoring requires the user to provide a head percolation table in a format specified for the tool, which will be discussed later in the section. While bracketing accuracy requires an alignment between the yields of the gold and test parses to establish constituent spans, head-dependency scoring can be run without an externally provided alignment. Note that labeled or unlabeled bracket or dependency metrics can be reported.

We had several other design constraints that we sought to satisfy with this tool. First, we wanted to provide the ability to evaluate parsing accuracy without an externally provided alignment file. Requiring the use of an user-provided alignment carries the risk that it could be chosen to optimize parser evaluation performance. In the absence of an alignment, dependency-based evaluation has obvious advantages over bracketing evaluation, to the extent that no span information is required. To evaluate the quality of dependency evaluation without alignment, we chose to provide a contrastive metric with alignment. This allows for controlled experimentation regarding the alignment-free methods of evaluation, as well as their validation. In addition, the use of an alignment allows the comparison of dependency and bracketing metrics.

A second design constraint was that we wanted users to be able to configure the tool using simple parameter files, similar to those used in the widely used *evalb* scoring tool (Sekine and Collins, 1997). Because dependency evaluation depends on head-percolation, we extended this flexibility to include the ability to specify the head-percolation table in a standard format. These parameterizations allow the tool to be used for various annotation standards.

Finally, we wanted the tool to require no special preprocessing of the trees for scoring. For that reason, the tool handles phenomena such as disfluency constituents in a way that is consistent with past practice (Charniak and Johnson, 2001), without taxing the user with anything more than indicating disfluency non-terminals (e.g., `EDITED`) in the parameter file.

SParseval was designed to be flexibly configurable to support a wide variety of scoring options. The scoring tool runs on the command line in Unix by invoking the *sparseval* executable with flags to control the scoring functionality. To use the tool, there are several input files that can be used to control the behavior of the evaluation.

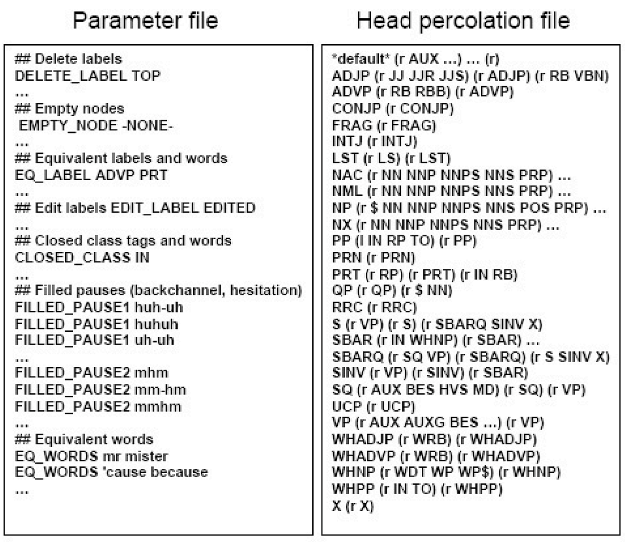
2.2. Input files

2.2.1. Gold and Test Parse files

Like *evalb*, *sparseval* expects one labeled bracketing per line for both the file of gold-standard reference trees and the file of parser-output test trees. There is a command line option to allow the gold and test parse files to be lists of files containing trees, each of which can be scored. In that case, each line is taken to be a filename, and gold trees are read from the files listed in the gold parse file, while test trees are read from the files listed in the test parse file. Without that command line option, lines in the files are expected to represent complete labeled bracketings.

2.2.2. Parameter file

As with *evalb*, a parameter file can be provided to parameterize the evaluation by dictating the behavior of non-



Parameter file

```
## Delete labels
DELETE_LABEL TOP
...
## Empty nodes
 EMPTY_NODE -NONE-
...
## Equivalent labels and words
EQ_LABEL ADVP PRT
...
## Edit labels EDIT_LABEL EDITED
...
## Closed class tags and words
CLOSED_CLASS IN
...
## Filled pauses (backchannel, hesitation)
FILLED_PAUSE1 huh-uh
FILLED_PAUSE1 huhuh
FILLED_PAUSE1 uh-uh
...
FILLED_PAUSE2 mhm
FILLED_PAUSE2 mm-hm
FILLED_PAUSE2 mmhm
...
## Equivalent words
EQ_WORDS mr mister
EQ_WORDS 'cause because
...
```

Head percolation file

```
*default* (r AUX ...) ... (r)
ADJP (r JJ JJR JJS) (r ADJP) (r RB VBN)
ADVP (r RB RBB) (r ADVP)
CONJP (r CONJP)
FRAG (r FRAG)
INTJ (r INTJ)
LST (r LS) (r LST)
NAC (r NN NNP NNPS NNS PRP) ...
NML (r NN NNP NNPS NNS PRP) ...
NP (r $ NN NNP NNPS NNS POS PRP) ...
NX (r NN NNP NNPS NNS PRP) ...
PP (l IN RP TO) (r PP)
PRN (r PRN)
PRT (r RP) (r PRT) (r IN RB)
QP (r QP) (r $ NN)
RRC (r RRC)
S (r VP) (r S) (r SBARQ SINV X)
SBAR (r IN WHNP) (r SBAR) ...
SBARQ (r SQ VP) (r SBARQ) (r S SINV X)
SINV (r VP) (r SINV) (r SBAR)
SQ (r AUX BES HVS MD) (r SQ) (r VP)
UCP (r UCP)
VP (r AUX AUXG BES ...) (r VP)
WHADJP (r WRB) (r WHADJP)
WHADVP (r WRB) (r WHADVP)
WHNP (r WDT WP WP$) (r WHNP)
WHPP (r IN TO) (r WHPP)
X (r X)
```

Figure 2: Example parameter and head table files for scoring parses based on non-terminals from the CTS Penn Treebank.

terminals and terminals in the trees. A skeletal parameter file appears in Figure 2 and a sample parameter file (named SPEECHPAR.prm) that is based on the terminal and non-terminal conventions of the CTS Penn Treebank is distributed with the tool. The file is used to provide several types of information to the scoring tool, following *evalb* conventions whenever possible.

`DELETE_LABEL`: The labels to be ignored need to be specified (e.g., `DELETE_LABEL TOP`). If the label is a pre-terminal, then the tool deletes the word along with the brackets. If the label is a non-terminal, it deletes the brackets but not the children. For scoring purposes, conventionally root non-terminals (e.g., TOP, S1), and punctuation pre-terminals are ignored using `DELETE_LABEL`.

`EMPTY_NODE`: Empty nodes are often removed from trees prior to evaluation. If empty nodes are to be removed, their labels should be indicated in the parameter file (e.g., `EMPTY_NODE -NONE-`).

`EQ_WORDS`, `EQ_LABEL`, `FILLED_PAUSE`: An optional list of equivalent words (e.g., `EQ_WORDS mr. mister`), non-terminal labels (e.g., `EQ_LABEL ADVP PRT`), and filled pause forms (e.g., `FILLED_PAUSE1 huh-uh`) can be specified. For filled pauses (e.g., backchannels and hesitations), the equivalency of the ith group of filled pauses is specified by using a unique label `FILLED_PAUSE`i. These equivalencies support different transcription methods, and in all cases are non-directional. For example, the letter "A" in an acronym may appear with a period in the gold standard transcript but without it in the ASR transcript.

`CLOSED_CLASS`: An optional list of closed class tags (e.g., `CLOSED_CLASS IN`) or words (e.g., `CLOSED_CLASS of`) can be specified for omission from the open class dependency metric.

`EDIT_LABEL`: An optional list of edit labels can be specified (e.g., `EDIT_LABEL EDITED`). This option is available to support parsing utterances that contain speech repairs (e.g., *I went I mean I left the store*, where *I went* is the edit or reparandum, *I mean* is an editing phrase, and *I left* is the alteration in a content replacement speech repair).

When scoring trees with edit labels, the internal structure of edit labeled constituents is removed and the corresponding spans are ignored for span calculations of other constituents, following (Charniak and Johnson, 2001). These edit labeled spans are ignored when creating head

```
Usage: sparseval [-opts] goldfile parsefile

Options:
 -p file   evaluation parameter file
 -h file   head percolation file
 -a file   string alignment file
 -F file   output file
 -l        goldfile and parsefile are lists
                     of files to evaluate
 -b        no alignment
              (bag of head dependencies)
 -c        conversation side
 -u        unlabeled evaluation
 -v        verbose
 -z        show info
 -?        info/options
```

Figure 3: Usage information from command line

dependencies for the dependency scoring. Errors in identifying edit spans have a different impact on dependency scores than on bracketing scores. In the bracketing score, the edit labeled span either matches or does not match. Since no dependencies are created for words in edit spans, no credit is given in the dependency score when spans perfectly match. However, dependency precision is negatively impacted for each word not in an edit span in the test parse that is in an edit span in the gold-standard. Conversely, each word placed inside of an edit span in the test parse that is outside of an edit span in the gold-standard negatively impacts dependency recall.

2.2.3. Head percolation file

For dependency scoring, a head percolation rule file must be provided. An abbreviated example is provided in Figure 2. The file indicates, for specific non-terminals plus a default, how to choose a head from among the children of a labeled constituent. A parenthesis delimits an equivalence class of non-terminal labels, and whether to choose the right-most (*r*) or left-most (*l*) if there are multiple children from the same equivalence class. The head-finding algorithm proceeds by moving in the listed order through equivalence classes, only moving to the next listed class if nothing from the previous classes has been found. If nothing has been found after all equivalence classes are tried, the default is pursued. For example,

```
PP (l IN RP TO) (r PP)
```

indicates that, to find the head child of a PP first the left-most IN, RP, or TO child is selected; if none of these categories are children of the PP, then the right-most PP child is selected; and if there are no PP children, the default rules are invoked. An empty equivalence class – e.g., `(r)` or `(l)` – matches every category. These rules are used recursively to define lexical heads for each non-terminal in each tree. We provide several example head tables that are configured based on the non-terminal conventions of the CTS Penn Treebank with the tool distribution, taken from Charniak (2000), Collins (1997), and Hwa et al. (2005).

2.2.4. Alignment file

To determine bracket scores when there are word errors in the input to the parser, the tool requires an alignment file to establish common span indices. For our purposes, we produced alignment files using SCLite (Fiscus, 2001) and a simple Perl formatting script. An example alignment file appears in Figure 1. We have added comments to indicate the meaning of the three-digit numbers used to indicate matches, substitutions, insertions, and deletions. Alignment files would also be required for bracket scores when parsing inputs that are automatically segmented into words (e.g., Mandarin), because there could be a mismatch in the tokenization of the input to the parser and the yield of the corresponding gold tree.

2.3. Command line options

The ease with which parameter and head percolation files can be created and updated makes the tool flexible enough to be applied under a wide variety of conditions. For example, we have used the tool to score test parses given a training-test split of the Mandarin treebank released by LDC. It was quite simple to create appropriate parameter and head table files to support scoring of test parses. The tool's flexibility also comes from the fact that it is invoked at the command line with a variety of flag options to control the scoring functionality. The way the tool is used depends on the type of data being parsed (speech transcripts with word errors or text that corresponds exactly to the gold standard text), the type of metric or metrics selected, and the availability of alignments. Figure 3 presents the Usage information for *sparseval*. Below, we first enumerate the switch options used with the *sparseval* command, and then provide a variety of examples of how the tool can be used to score parse trees.

- `-p` The parameter file discussed in section 2.2.2. is specified using the *-p file* switch.
- `-h` The head percolation file discussed in section 2.2.3. is specified using the *-h file* switch.
- `-a` The alignment file discussed in section 2.2.4. is specified using the *-a file* switch.
- `-F` Sometimes it is convenient to specify the output file in the command line. This is done with the *-F file* switch. Output defaults to *stdout*.
- `-l` To indicate that the gold and test files discussed in section 2.2.1. specify lists of files rather than labeled bracketings, the *-l* option is used; otherwise, the files input to the tool must contain labeled bracketings.
- `-b` If no alignment information is available and there is some mismatch between the yields of the test and gold parses, then the *-b* option should be used. This indicates that a bracketing score will not be calculated, and only a bag of head dependencies score will be produced. Note that there are temporal no-cross-over constraints on matching dependencies that prevents dependencies that are not temporally near each other from matching.
- `-c` If the evaluation is to be done on a speech chunk basis rather than at the sentence level, the *-c* switch must be used. If this switch is not included, the parser assumes that the evaluation should perform the comparison on a line-by-line basis. When this switch is set, it is assumed that all of the gold parses associated with the speech chunk appear together in a single file, and similarly for the test parses.
- `-u` To provide *unlabeled* scores, the *-u* switch should be used.
- `-v` To produce a verbose scoring report from the scoring tool (i.e., one that provides scores for each speech chunk to be evaluated, in addition to the summary over all speech chunks), the *-v* switch should be used. An example of a verbose output file over five conversation sides is shown in Figure 5.
- `-z` To show additional configuration information in the output, the *-z* switch should be used.

The way the tool is used depends on whether it is being applied to parse trees such that each tree's yield perfectly aligns the words in the corresponding gold standard or not. If the tool is applied to parses of sentences with "perfect" alignment, which would be the case when scoring parses in the test set of the Wall Street Journal Penn Treebank (Marcus et al., 1993), then the tool would be invoked similarly to *evalb*, as shown in figure 4(a), where `gold` is a file containing gold parses and `test` is a file containing test

```
(a)   sparseval -p SPEECHPAR.prm gold test -F output
(b)   sparseval -l -p SPEECHPAR.prm -h headPerc -c -b gold-files test-files -F output
(c)   sparseval -v -l -p SPEECHPAR.prm -h headPerc -c -a align-files gold-files test-files -F output
```

Figure 4: Three command lines for using *sparseval* with (a) standard text parse evaluation; (b) evaluation of parsing errorful ASR system output, with no alignment; and (c) evaluation of parsing errorful ASR system output, with alignment.

```
                   Labeled                      Labeled              Labeled
Sent.  ---------- Brackets ------  ---- Head-depen ---  -- Open-class hh --
 ID   match   gold   test  cross  match   gold   test  match   gold   test
================================  ===================  ===================
    1   255    356    349     41    251    342    338    215    293    289
    2   499    795    825    162    449    701    740    354    565    578
    3   313    529    500    112    278    468    447    202    349    334
    4   193    346    337     67    173    308    310    131    242    221
    5   282    571    507    111    228    484    446    167    382    341
================================  ===================  ===================
Total:  1542   2597   2518    493   1379   2303   2281   1069   1831   1763

Summary
-----------------------------------------
Number of sentences:                           5
Labeled Bracketing Recall:                         59.38
Labeled Bracketing Precision:                      61.24
Labeled Bracketing F-measure:                      60.29
Complete match:                                0
-----------------------------------------
Average Crossing Brackets:                 98.60
No Crossing Brackets:                       0.00
Two or less Crossing:                       0.00
-----------------------------------------
Labeled Head-dependency Recall:                    59.88
Labeled Head-dependency Precision:                 60.46
Labeled Head-dependency F-measure:                 60.17
-----------------------------------------
Labeled Open-class head-dependency Recall:         58.38
Labeled Open-class head-dependency Precision:      60.64
Labeled Open-class head-dependency F-measure:      59.49
```

Figure 5: Verbose output from scoring five conversation sides.

parses. We can also use the tool to evaluate parse quality given ASR transcripts. The command that produces a bag-of-dependencies score for the files in `test-files` given the gold standard files specified in `gold-files` is shown in figure 4(b). This does not require an alignment file. To perform bracket based scoring, it is necessary to supply a list of alignment files as shown in figure 4(c). Figure 5 displays the verbose output from the command in figure 4(c). Because of the specified options, this command uses word alignments to provide labeled bracket spans, head dependency, and open-class head dependency counts for each speech chunk, together with a summary reporting a variety of scores over all speech chunks. If the `-v` flag were omitted, only the summary would have been produced.

3. Metric Evaluation

Since the SParseval tool was developed to cope with word and sentence segmentation mismatch that arises when parsing speech, we examine the impact of these factors on the metrics. Due to space limitations, we will only summarize the findings reported in full in Harper et al. (2005), in which we report more fully on our experience of using the SParseval metrics. Our goal was to investigate the impact of metadata and transcription quality on the parse metrics when applied to conversational speech; hence, we utilized the RT'04F treebank (LDC2005E79) that was carefully transcribed, annotated with metadata, including sentence units (called SUs) and speech repair reparanda (called edits) according to the V6.2 specification (Strassel, 2004), and then annotated for syntactic structure using existing CTS treebanking guidelines (Bies et al., 2005).[2]

We have conducted a series of empirical studies to investigate the sensitivity of the SParseval parsing metrics to a variety of factors that potentially impact parse accuracy on speech. This study was carried out by applying our parse scoring tool to parses generated by three different parsers; the Charniak (2000) and Roark (2001) parsers were trained on the entire Switchboard corpus with dev1 as a development set; whereas, the Bikel (2004) parser was trained on the combination of the two sets. We chose to investigate parse metrics across parsers to avoid the potential bias that could be introduced by investigating only one. Each of the metrics were then extracted from parses produced by the parsers on the RT'04 dev2 set under a variety of conditions: the input to the parser was either a human transcript or a transcript output by a state-of-the-art speech recognizer; it either had human transcribed metadata or system produced (Liu et al., 2005) metadata; and the metadata indicating the location and extent of the edited regions was used to remove that material prior to parsing or not (and so the parsers process the edits together with the rest). We examined the impact of the above data quality and processing factors on the F-measure scores produced by the three parsers on the dev2 conversation sides. The F-measure scores varied along a number of dimensions: bracket versus head dependency, all dependencies versus open class only, with versus without labels, and with versus without alignment. To determine the dependency scores, we utilized the three head percolation tables mentioned in Section 2.

In general, we found that the dependency F-measure scores are on average quite similar to the bracket F-measure scores, and correlate highly with them, i.e. $r = .88$, as do the open class and overall head dependency F-measure scores, $r = .99$. Despite the fact that the correlations between the metrics are quite high, we have found that they differ in their sensitivity to word and sentence segmentation errors. For example, the dependency metrics appear to be less sensitive to sentence boundary placement than the bracket scores, as can be observed in Figure 6. The figure presents SU error along with bracket and head dependency F-measure accuracy scores (using the Charniak head percolation table) across a range of SU detection thresholds.[3] The figure highlights quite clearly that the impact of varying the threshold on bracket scores differs substantially from the impact on dependency scores, on which the impact is somewhat limited except at extreme values. It also highlights the fact that minimizing sentence error does not always lead to the highest parse accuracies, in particular, shorter sentences tend to produce larger parse scores, especially for bracket scores.

We have conducted two analyses of variance to better understand the impact of data quality on the metrics. The first was based on F-measure scores obtained with alignment on the 72 conversation sides of the dev2 set collapsing over head percolation table: 3(Parser: Bikel, Charniak, or Roark) $\times$ 2(Transcript Quality: Reference or ASR) $\times$ 2(Metadata Quality: Reference or System) $\times$ 2(Use of Edit Metadata: use it or not) $\times$ 3(Parse Match Representation: bracket, overall head dependency, or open-class head dependency) $\times$ 2(Labeling: yes or no) analysis of variance (ANOVA). The second was focused on dependency F-measure scores alone in order to investigate the impact of alignment: 3(Parser) $\times$ 2(Transcript Quality) $\times$ 2(Meta-

[2]Three subsets were released: eval is the RT'04 evaluation data set (with 36 conversations, 5K SUs, 34K words), dev1 is a combination of the RT'03 MDE development and evaluation sets used as a development set for RT'04 (72 conversations, 11K SUs, and 71K words), and dev2 is a new development set created for RT'04 (36 conversations, 5K SUs, and 35K words).

[3]The basic SU detection system places an sentence boundary (SU) at an inter-word boundary if the posterior probability is greater than or equal to a threshold of 0.5. The higher the threshold, the fewer boundaries are placed, hence the longer the sentences.

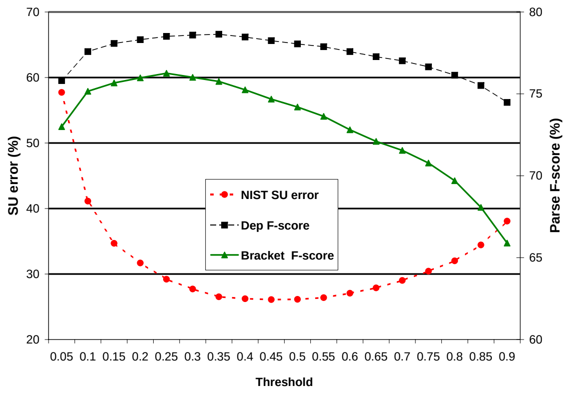


Figure 6: The impact of sentence detection threshold on sentence boundary and parse accuracy.

data Quality) × 2(Use of Edit Metadata) × 2(Parse Match Representations: overall head dependency or open-class head dependency) × 2(Labeling) × 2(Alignment: yes or no) × 3(Head Percolation Table: Charniak (2000), Collins (1997), or Hwa et al. (2005)) ANOVA of the dependency parse scores. We report selected findings of these analyses, starting with some of the significant main effects:

- Parse scores are, on average, significantly greater when the input to the parser is based on hand transcripts rather than ASR transcripts; there was a significant main effect of Transcript Quality in each ANOVA, $F(1, 78) = 19,127.6, p < .0001$ and $F(1, 157) = 47,641.6, p < .0001$, respectively. In the former analysis, parses from reference transcripts had a significantly greater F-measure (81.05) than those based on ASR transcripts (68.95), $p < .0001$, confirming our intuitions that word errors degrade parsing performance. We also investigated the impact of word errors on parse accuracy by using ASR systems with different error rates, and found in general, the greater the WER, the lower the parse scores.
- Parse scores are, on average, significantly greater when using human annotated sentence boundaries and edit information than when using what is produced by a system; there was a significant main effect in each ANOVA, $F(1, 78) = 7,507.85, p < .0001$ and $F(1, 157) = 10,199.9, p < .0001$, respectively. In the former analysis, parse scores obtained based on reference annotations had a significantly greater F-measure (78.20) than those produced by the metadata system (71.80), $p < .0001$. By using metadata detection systems with different error rates, we also investigated the impact of metadata error on the the parse scores, and found that the greater the system error, the lower the parse scores.
- Parse scores are, on average, significantly greater when removing edits prior to parsing the input sentence; there was a significant main effect in each ANOVA, $F(1, 78) = 1,335.89, p < .0001$ and $F(1, 157) = 2,419.35, p < .0001$, respectively. In the former analysis, parse scores obtained by using the edit annotations to simplify the input to the parse resulted in significantly greater F-measure (76.49) than those from parsing the sentences containing the edits (73.51), $p < .0001$.
- In each ANOVA, there was a significant main effect of the use of the parse match representation, $F(2, 78) = 5.61, p < .005$ and $F(1, 157) = 20.16, p < .0001$, respectively. In the former ANOVA, we found that the open class dependency F-measure score (75.14) is slightly, though significantly, larger than the overall head dependency F-measure score (74.88), $p < .005$. Bracket scores (74.93) do not differ significantly from the other two scores. A similar trend is preserved in the second dependency-only ANOVA.
- In the dependency-only ANOVA, there was a significant main effect of the Head Percolation Table, $F(2, 157) = 195.44, p < .0001$, with Charniak's table producing significantly larger scores (75.91) than Collins' table (75.14), which were larger than those produced using Hwa's table (74.54), $p < .0001$. Based on additional analysis, not only does the Charniak table produce higher scores in general across all three parsers, the table also shows a greater robustness to ASR transcript word error. Dependency parses produced with Charniak's table also produced relatively larger unlabeled scores than the other two tables.
- In the dependency-only ANOVA, the main effect of Alignment was also significant, $F(1, 157) = 43.14, p < .0001$, with scores obtained without the alignment constraint being slightly, although significantly, greater (75.38) than those obtained with alignment (75.01), $p < .0001$. Alignment adds an extra match constraint and so reduces dependency scores slightly compared with scores calculated without this constraint. Based on additional analysis, the relative improvement from relaxing the alignment constraint is greater when using ASR transcripts and when not removing edits prior to parsing. Despite this, alignment does not appear to play a major role for dependency metrics, even though it is required in order to calculate the bracket scores.

An important question we sought to answer in these studies was how effective dependency scoring is in the absence of an externally provided alignment. Recall that the dependencies that are scored are `(dependent word, relation, head word)`, where the relation is determined using a provided head percolation table. The relation is the non-head non-terminal label and the head non-terminal label. We include a special dependency for the head of the whole sentence, with the root category as the relation. Note that in this formulation each word is the dependent word in exactly one dependency. The dependency score in the absence of an alignment takes ordered sequences of dependency relations – ordered temporally by the dependent word – and finds the standard Levenshtein alignment, from which precision and recall can be calculated. Since this alignment maximizes the number of matches over ordered alignments, any user provided alignment will necessarily decrease the score. The results above demonstrate that omitting the alignment causes a very small over-estimation of the dependency scores.

There were also significant interactions in the ANOVAs involving data quality and data use, but as our focus is on the sensitivity of the metrics, we focus here on interactions involving the parse metrics in the first ANOVA: Labeling × Parse Match Representation, $F(2, 78) = 13.36, p < .0001$; Transcript Quality × Parse Match Representation, $F(2, 78) = 66.24, p < .0001$; Labeling × Transcript Quality × Parse Match Representation, $F(2, 78) = 8.23, p < .0005$; Metadata Quality × Parse Match Representation, $F(2, 78) = 246.17, p < .0001$; and Use of Edit Metadata × Parse Match Representation, $F(2, 78) = 3.53, p < .05$.

To get a better sense of some of these interactions, consider Figure 7. Ignoring labels during scoring benefits the dependency scores much more than the bracket-based scores. Although all of the scores, regardless of representation, are relatively lower on ASR transcripts than on reference transcripts, the dependency scores are more

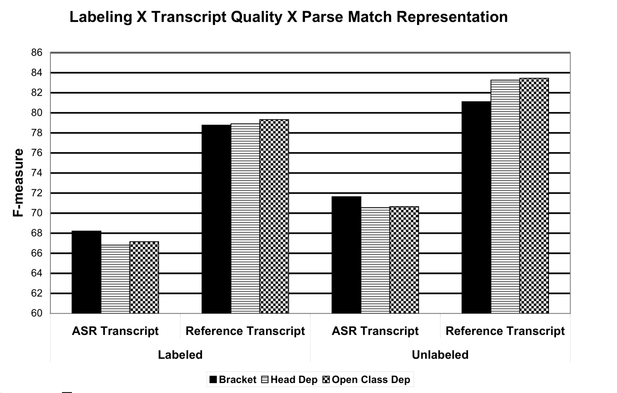


Figure 7: Average F-measure scores given labeling, transcript quality, and parse match representation.

negatively impacted than bracket scores. They were significantly larger than the bracket scores on reference transcripts, but significantly smaller than the bracket scores on ASR transcripts, $p < .0001$. The degradation caused by using ASR transcripts is comparable for all of the labeled and unlabeled dependency scores (around 15.3% for labeled and unlabeled head and open class dependencies), but is less for the labeled and unlabeled bracket scores (13.4% and 11.7%, respectively).

As can be seen in Figure 8, bracket scores are more sensitive to sentence segmentation errors than their dependency counterparts. Bracket scores are significantly greater than both the overall and open class dependency scores given reference metadata ($p < .0001$); however, when system metadata is used, the bracket scores become relatively lower than the dependency scores ($p < .0001$). A similar trend was found for the interaction between use of edit markups and parse match representation; bracket scores are hurt more by leaving the edited material in the word stream than the dependency scores.

4. Summary

We have presented a parsing evaluation tool that allows for scoring when the parser is given errorful ASR system output with system sentence segmentations. The tool provides a lot of flexibility in configuring the evaluation for a range of parsing scenarios.

The metric evaluation studies suggest all of the parse metric factors are not strictly orthogonal to each other given the data quality factors, e.g., ignoring labels tends to improve dependency scores more than bracket scores on ASR transcripts. Metadata errors have a greater negative impact on bracket scores than dependency scores; whereas, word errors have a greater impact on dependency scores, which use word identity as a match criterion, than bracket scores, which simply use alignment. Dependency scoring without alignments was shown to be an effective evaluation option.

Acknowledgments: The authors would like to thank the Johns Hopkins CLSP faculty and staff, Dustin Hillard, Elizabeth Shriberg, Andreas Stolcke, Wen Wang, Stephanie Strassel, Ann Bies, and the LDC treebanking team. This report is based upon work supported by DARPA under contract number MDA972-02-C-0038 and HR0011-06-2-0001, by the National Science Foundation under grant numbers 0121285, 0326276, and 0447214, and by ARDA under contract number MDA904-03-C-1788. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, DARPA, or ARDA.

5. References

A. Bies, J. Mott, and C. Warner, 2005. *Addendum to the Switchboard Treebank Guidelines*. Linguistic Data Consortium.

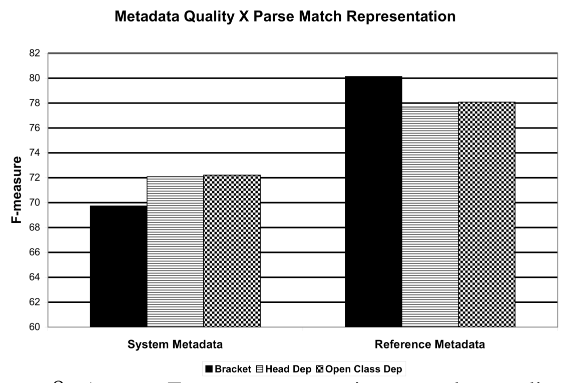


Figure 8: Average F-measure scores given metadata quality and the parse match representation.

D. M. Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing syntactic coverage of English grammars. In *Proceedings 4th DARPA Speech & Natural Lang. Workshop*, pages 306–311.

J. Carroll, A. Frank, D. Lin, D. Prescher, and H. Uszkoreit (eds.). 2002. Proceedings of the LREC workshop 'Beyond PARSEVAL — Towards improved evaluation measures for parsing systems'. http://www.cogs.susx.ac.uk/lab/nlp/carroll/papers/beyond-proceedings.pdf.

J. Carroll (ed.). 1998. Proceedings of the LREC workshop 'The evaluation of parsing systems'. http://www.informatics.susx.ac.uk/research/nlp/carroll/abs/98c.html.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of NAACL*, pages 118–126.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*.

J. Fiscus. 2001. SClite- score speech recognition system output. http://computing.ee.ethz.ch/sepp/sctk-1.2c-be/sclite.htm.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, volume I, pages 517–520.

M. Harper, B. Dorr, J. Hale, B. Roark, I. Shafran, M. Lease, Y. Liu, M. Snover, L. Yung, R. Stewart, and A. Krasnyanskaya. 2005. 2005 Johns Hopkins Summer Workshop Final Report on Parsing and Spoken Structural Event Detection. http://www.clsp.jhu.edu/ws2005/groups/eventdetect/documents/finalreport.pdf, November.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*.

J. G. Kahn, M. Ostendorf, and C. Chelba. 2004. Parsing conversational speech using enhanced segmentation. In *HLT-NAACL 2004*, pages 125–128.

Y. Liu, E. Shriberg, A. Stolcke, B. Peskin, J. Ang, D. Hillard, M. Ostendorf, M. Tomalin, P. Woodland, and M. Harper. 2005. Sructural metadata research in the EARS program. In *Proceedings of ICASSP*.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

S. Sekine and M. J. Collins. 1997. The evalb software. http://cs.nyu.edu/cs/projects/proteus/evalb.

S. Strassel, 2004. *Simple Metadata Annotation Specification V6.2*.