2005 Johns Hopkins Summer Workshop Final Report on Parsing and Spoken Structural Event Detection

Mary Harper[a,b], Bonnie Dorr[b], John Hale[c], Brian Roark[d], Izhak Shafran[e], Matthew Lease[f], Yang Liu [g,h], Matthew Snover[b], Lisa Yung[e], Anna Krasnyanskaya[i], Robin Stewart[j]

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

Abstract

This report describes research conducted by the Parsing and Spoken Structural Event Detection (PaSSED) team as part of the 2005 Johns Hopkins Summer Workshop on Language Engineering. This project investigated the interaction between parsing and the detection of structural metadata in conversational speech, including sentence boundaries, edits (the reparandum portion of speech repairs), and fillers. In terms of parsing, we explored alternative methods of exploiting metadata information in parsing models and measured how varying accuracy in transcription and metadata information affects parsing accuracy. In the other direction, we similarly considered how syntactic and prosodic knowledge could be leveraged in metadata detection, measuring how this knowledge impacts metadata detection accuracy.

As part of this work, we investigated metrics for evaluating parse accuracy in the presence of transcription and metadata detection errors, and we report on our experience using these metrics with several parsers and across varying experimental conditions. A range of methods for handling edits during parsing were evaluated in this research (excision, addition of markups to the input string, and grammar modification). We also developed a ToBI (a prosodic structure annotation scheme [SBP+92]) prosodic event classifier and describe its evaluation. Finally, we present methods for effective n-best sentence boundary candidate generation and reranking using syntactic, prosodic, and other features. These studies are complemented by a second set of reranking investigations wherein we optimize sentence boundary detection explicitly to improve parse accuracy.

The PaSSED project has:

- investigated various techniques to enhance parsing of speech given metadata detection on conversational speech;
- defined metrics for evaluating speech parsing accuracy, implemented them in the publically available SParseval software package, and evaluated them under a wide variety of conditions;
- recast SU detection as an n-best reranking problem with a relatively small n. Using this approach, we demonstrated significant improvements over a very strong baseline SU detection system.
- reported on the interaction between parsing and metadata detection and their synergy;
- fostered new collaborations and identified a number of interesting avenues for future work.

1 Motivation

In order to apply language processing techniques to speech that have been traditionally applied to text, it is important to address the inherent differences between these two types of inputs. Textual input typically involves words that are broken into sentences and clauses using punctuation. Sentences are further organized into chunks such as paragraphs, sections, chapters, articles, books, and so on. Although speech is similar in many ways to text (e.g., it is comprised of words that have the same meaning as in text), it also has many differences, some stemming from the fact that people use different modalities/cognitive processes when processing/producing these inputs/outputs, and others stemming from the different ways in which these two methods of communication are conventionally expressed. State-of-the-art automatic speech recognition (ASR) systems tend to focus on getting the words correct given that word error rate (WER) has been the metric minimized by such systems. Although WER is decreasing, ASR systems do not currently generate/model structural information of the kinds that are available to someone who is reading a text. In fact, spoken language is typically not as highly organized as textual material (e.g., spontaneous speech) and often contains phenomena such as speech repairs that do not appear in text. These aspects of spoken language present a challenge for systems attempting to bridge the gap between speech processing and natural language processing systems. In this workshop effort, the Parsing and Spoken Structural Event Detection (PaSSED) team has focused on the parsing of speech as a first step in building this bridge.

ASR system quality is typically measured in terms of the accuracy of the word sequence. However, automated speech processing applications can benefit from (or sometimes even require) system output that is richer than an undelimited sequence of recognized words. For example, sentence breaks and disfluency annotations are critical for legibility [JWG$^+$03], as well as for downstream processing algorithms that have a complexity polynomial in the length of the string, such as parsing. Because automatic detection of sentence breaks and speech repairs is important for bridging between speech and text processing systems, there has been a growing interest in automatically enriching speech recognition output with structural information, further spurred by the structural metadata effort in the DARPA EARS program [DAR03]. The structural events investigated under EARS include sentence boundaries (SUs), fillers, and speech repairs (reparanda and interruption points) [Str04]. Most of the systems that automatically detect these structural events in this program (see [LSS$^+$05] for an overview) combine prosodic and textual information, using only a limited amount of syntactic knowledge (i.e., POS or simple chunk tags) [1]. Initial efforts have focused on English, but are also expanding to include other languages (e.g., Mandarin and Arabic).

Most current parsers assume that sentence boundaries are given and parse at the sentence level; however, speech recognizers produce only words as output. Although recognizers process segments of speech, these rarely correspond to a sentence in text. This can cause serious problems for downstream parsing technology and would also tend to negatively impact the quality of a parser-based language model. There are in fact two issues. First, there needs to be a match between parser training materials and conditions for applying the parser (or structured LM). Second, parser input needs to be effectively exploitable by the parser model (suggesting that the input segments need to be consistently produced and be a chunk of words that the parser's grammar can reasonably account for). Sentence like units, if they can be automatically produced with high accuracy, would provide better conditions for a parser or structured language model. Kahn, Ostendorf, and Chelba [KOC04] have achieved significant error reductions in parsing performance for conversational English when using sentence boundary information from a state-of-the-art EARS structural event detection system, suggesting that the automatic boundary information helps, although perfect boundaries provide further improvement.

Speech repairs also pose a serious challenge for accurate parsing (e.g., *I went I mean I left the store*, where *I went* is the reparandum, *I mean* is an editing phrase, and *I left* is the alteration in a content replacement speech repair). Effective automatic identification of these kinds of disfluencies and their structure would provide a mechanism for cleaning up transcripts for the downstream text processing modules. Charniak and Johnson [CJ01] evaluated parsing performance on human generated transcripts of Switchboard conversational speech given perfect knowledge of sentence boundaries but without knowledge of the location of speech repairs. Because their parser is context free, they used a preprocessing step

[1] One exception is [JCL04]

to automatically detect speech repairs to remove them prior to parsing. This is largely due to the fact that CFGs are unable to model cross serial dependencies that characterize the correspondence between a reparandum and alteration of a repetition or content replacement speech repair. Recent efforts by [JC04] used a Tree Adjoining Grammar (TAG) based noisy channel model for disfluency detection to model the cross-serial dependencies in the most common subset of speech repairs for accurate detection and removal of those elements prior to parsing. Although their system decouples the detection of speech repairs from parsing, based on their most recent work (i.e., [JCL04]), it appears that information from the parser and the noisy channel model are mutually informative. Wang [Wan03] achieves WER reductions when using an almost-parsing Constraint Dependency Grammar (CDG) LM (the SuperARV LM) trained on the Hub5 2001 (also Switchboard) training set after application of a sanitize and resegmentation method, which removes the obvious disfluencies and breaks complex speech segments into simple sentences that are more amenable to parsing, adding evidence that modeling sentence boundaries and speech repairs is beneficial for processing speech. CDG can also be used to directly represent the correspondences between the elements of a reparandum and alteration based not only on word identity but also on syntactic similarity. Recent experiments by Wang and Harper have shown that, by modeling these correspondences in a SuperARV LM, perplexity is reduced from 98.9 to 86.55 on the Switchboard Penn Treebank corpus.

Clearly parsers should benefit from more accurate information about sentence boundaries and speech repairs when processing transcripts of spontaneous speech; however, there is a great deal that remains to be investigated. Much of the work on spoken parsing accuracy has been done with perfect knowledge of sentence boundaries, largely due to the fact that existing parsing metrics are applied at the sentence level. For our efforts, it was essential to develop and evaluate parsing metrics that are applicable in the face of both word and sentence boundary errors; hence, we have developed SParseval, which is defined in Section 3.1. Using SParseval, we have performed a variety of experiments on the impact of automatic versus reference metadata and ASR versus reference transcripts on parsing accuracy. We have also performed experiments on the impact of prosodic and syntactic knowledge sources on metadata quality.

This report is organized into several topical sections. In Section 2, we provide information about the baseline metadata systems used at the workshop. In Section 3, we describe the SParseval scoring package and a series of studies investigating the SParseval metrics over a variety of data quality conditions. These studies not only attempt to tease out the factors that affect parsing accuracy on speech, but also the sensitivity of the various metrics to the information loss caused by the impact of transcript and metadata accuracy on parse accuracy. They also assess the impact of more accurate sentence boundaries and disfluency annotations on parsing accuracy, while factoring out the effects of the metadata errors and ASR errors individually. Also discussed in this section is the impact of fillers on parse accuracy. Section 4 investigates acoustic cues related to prosodic events in ToBI, a popular scheme for representing the prosodic structure of a spoken utterance [SBP$^+$92], in order to construct classifiers to detect a subset of these events. Given this classifier, we have investigated a variety of ways of incorporating its predictions into metadata detection and parsing systems. An issue that spans many sections (i.e., Sections 3, 4, 5, and 6) is the way in which the interaction between metadata and parsing is modeled. This can be done in a number of ways:

- Parsing using metadata to segment and "clean up" the input to the parser: For edits, this would imply detecting and then excising the edited regions prior to parsing (e.g., [JC04]), which is the method used in Section 3.

- Enriching the input by augmenting the words with special symbols: Section 4 investigates the use of labels generated by the ToBI event classifier to enrich the input to the parser so that it can identify and better cope with edited regions of speech. Section 5.2 investigates the use of EDIT and FILLER markups for parsing transcripts of spoken SUs using the Minipar parser [Lin01], which is able to take this annotated text directly as input, without requiring training on the annotations or deletion and reinsertion of the annotations.

- Enriching the grammar, i.e., modeling edits implicitly: Section 5.3 explores several ways of enriching the grammar specifically for disfluency detection.
- Re-ranking an initial set of hypothesis, possibly using additional cues that cannot be easily incorporated in a grammar: In Section 6, we have investigated the impact of incorporating syntactic, prosodic, and other knowledge sources into metadata detection of SU by using a reranking approach.

2 The Baseline Structural Metadata System

In order to provide background information, in this section, we describe the metadata extraction (MDE) tasks and briefly summarize the ICSI/SRI/UW MDE baseline system [LSS$^+$04b, LSS$^+$04a], which most of the research summarized in this report builds upon.

2.1 MDE Tasks

The MDE research effort within the DARPA EARS program [DAR03] aims to enrich speech recognition output by adding automatically tagged information on the location of sentence boundaries, speech disfluencies, and other important phenomena. The data is annotated according to annotation guidelines [Str04]. There are four tasks for structural MDE in EARS in the most recent evaluations:

- Sentence-like unit (SU) detection aims to find the end point of an SU. Detection of subtype (statement, backchannel, question, and incomplete) for each SU boundary was also a task evaluated in RT'04. However, in this report we focus only on SU boundary detection, which provides important segmentation information for downstream language processing.
- Filler word detection aims to identify words used as filled pauses (e.g., *uh*, *um*), discourse markers (e.g., *you know*, *so*, *like*), and explicit editing terms inside speech repairs.
- Edit word detection aims to find all words within the reparandum region of an edit disfluency. Edit disfluencies generally follow the template:

  (reparandum) * ⟨editing term⟩ correction

  By detecting the reparandum and removing it, more fluent utterances can be generated.
- Interruption point (IP) detection aims to find the interword location at which point fluent speech becomes disfluent. In addition to the IPs inside an edit disfluency, the starting point of a filler word string is considered an IP.

Evaluation is conducted using both the human (or reference) transcripts and speech recognition output, the latter to assess the impact of recognition errors. Two corpora with different speaking styles are used for MDE evaluation: conversational telephone speech (CTS) and broadcast news. Each MDE task is evaluated separately, using its own performance measure. Scoring tools, created for these tasks by NIST, first align the reference and hypothesis words. This is straightforward when evaluating on human transcripts since the words used by the system match the reference exactly; however, when speech recognition output is used, the system words typically do not align perfectly with those in the reference transcripts. In this case, an alignment that minimizes the word error rate is used. After alignment, the hypothesized structural events are mapped to the reference events using the word alignment information, and then unmatched structural events (i.e., insertions and deletions) are counted. For SU and IP detection, the error rate is the average number of misclassified boundaries per reference event:

$$\text{SU error rate} = \frac{\text{number of incorrect boundaries}}{\text{total number of SU boundaries}} \tag{1}$$

$$\text{IP error rate} = \frac{\text{number of incorrect boundaries}}{\text{total number of IP boundaries}} \tag{2}$$

For edit and filler word detection, the error rate is the average number of misclassified reference tokens per reference edit or filler word token:

$$\text{Edit word error rate} = \frac{\text{number of misclassified words}}{\text{total number of edit words}} \tag{3}$$

$$\text{Filler word error rate} = \frac{\text{number of misclassified words}}{\text{total number of filler words}} \tag{4}$$

A detailed description of the scoring tool is provided in [Fal04]. Note that the NIST metric error rates can exceed 100%. The following example shows a system SU hypothesis aligned with the SU reference:

```
Reference:  w1       w2   w3  /   w4
System:     w1  /    w2   w3      w4
               ins           del
```

where w_i is a word and '/' indicates an SU boundary. This example has two misclassified boundaries: one insertion error and one deletion error (indicated by 'ins' and 'del'). Since there is only one reference SU boundary, the NIST SU error rate for this system output is 200%.

In addition to the previous measures, other performance metrics such as recall/precision/F-measure and word-level classification error rate [Liu04] can be used. The word-level (also called boundary-based) classification error rate differs from the NIST performance metrics above in that the denominator is the total number of all the words:

$$\text{word-level error rate} = \frac{\text{number of incorrect boundaries}}{\text{total number of words}} \tag{5}$$

This metric can be applied to any of the MDE tasks. If the metadata events are the classes to be detected, then F-measure can be used as a metric to balance precision and recall:

$$F\textit{-measure} = \frac{(1+\beta^2) * \textit{recall} * \textit{precision}}{\beta^2 * \textit{recall} + \textit{precision}} \tag{6}$$

where $\textit{precision} = \frac{TP}{TP+FP}$, $\textit{recall} = \frac{TP}{TP+FN}$, and TP and FP denote the number of true positives and false positives, respectively. FN represents the number of false negatives, and β corresponds to the relative importance of *precision* versus *recall*. β is set to 1 if false alarms and misses are considered equally costly.

Detection-error tradeoff (DET) curves have also been used to compare the performance of systems that output posterior probabilities of SU events at each word boundary [HO04]. A DET curve displays the missed detection rate versus the false detection rate on a log-log scale [MDK+97]. For assessing the significance of performance differences, a matched pair test is used, which compares scores on segments of speech from a single speaker, bounded by a pause of at least 3 seconds [HO04].

2.2 The Baseline MDE System

This section provides information about the baseline metadata systems used at the workshop and their performance.

2.2.1 General Approaches

The MDE tasks can be seen as classification tasks that determine whether an interword boundary is an event boundary (e.g., SU or IP) or whether a word belongs to an event of interest (filler word or edit word). To detect metadata events, prosodic and textual information is utilized. Typically, at each interword boundary, prosodic features are extracted to reflect pause length, duration of words and phones, pitch contours, and energy contours. These prosodic features are modeled by a classifier (e.g., a decision tree), which generates a posterior probability of an event given the feature set associated with a boundary. Textual cues include contextual information of the words, their corresponding classes, and some syntactic information. We have examined three different modeling approaches for the MDE tasks: HMMs, maximum entropy (MaxEnt), and conditional random fields (CRF).

In the HMM framework, the transition probabilities are modeled generally by a hidden event n-gram LM to capture the structural events associated with the word sequences. Additional LMs (involving words or classes) can be interpolated with the basic hidden event LM. The observation (with features F) probability $P(F|E)$ is obtained from a prosody model implemented as a decision tree that generates the posterior probability of the events given the features, $P(E|F)$. Forward-backward decoding [SSHTT00] is used to find the most likely event at each point.

The MaxEnt and CRF approaches address some of the weakness of the generative HMM approach [LSSH05]. These approaches directly estimate the posterior probability of an event given observations and better match the performance metrics. Additionally, they provide more freedom for incorporating various knowledge sources, especially overlapping features. The posterior probability estimation of the event (E) given the features (O) in the MaxEnt and CRF follows the exponential form, i.e., $p(E|O) = \frac{1}{Z(O)} e^{\sum \lambda_k g_k(E,O)}$, where Z is a normalization term, and $g(E,O)$ are indicator functions. The MaxEnt and CRF approaches differ in that the latter uses the sequence information, that is, the O in the formula above is the entire sequence; whereas, the former approach uses "features" associated with one point. We use a first-order CRF that captures the dependency between a state and its previous state along with efficient Viterbi decoding for sequence decoding.

2.2.2 Setup for SU and Edit Detection

Since SU detection and edit detection tasks are most relevant to the research conducted in the workshop, we provide modeling details and present baseline results for these two tasks.

For SU detection, in addition to the word identities, we make use of various automatic taggers that map the word sequence to other representations, including POS tags and automatically induced classes, as well as prosodic information. One of the three modeling approaches (described above) is first used to detect the SU boundary, and then a second step is used to determine the subtype of the SUs (statement, question, backchannel, or incomplete SU) using a MaxEnt classifier [Liu04].

For HMM SU detection, the state transition probability is obtained by combining three LMs: word-based LMs trained from the LDC data and an extra textual corpus, and an automatically induced class-based LM. The observation probability is calculated based on a decision tree prosody model, which uses bagging for more robust probability estimation. Since POS tags cannot be obtained on the fly, we adopt a loosely coupled approach: for each word boundary, the POS-based LM is applied via the HMM approach (without using the prosody model) to generate posterior probabilities of events, which are then combined with the posteriors from the other models (i.e., the various LMs and the prosody model).

For the MaxEnt and CRF approaches, the following features are used for SU detection:

- N-grams of words, POS tags, and automatically induced classes. These features use various values of N and starting positions.

- A binary turn feature is used to indicate whether there is a speaker change.
- Prosodic information is incorporated by cumulatively binning the posterior probabilities from the prosody model into several binary features through thresholding in a cumulative fashion: $p > 0.1$, $p > 0.3$, $p > 0.5$, $p > 0.7$, $p > 0.9$, with heuristically chosen thresholds.
- It is convenient to include posterior event probabilities from the additional textual LM (obtained using the HMM framework) as features, in a way similar to the prosody model.
- A compound feature involving the decision tree's hypothesis and POS tags is used.

For edit word detection, we used a CRF classifier, for which each word has an associated tag, representing whether or not it is an edit word. The classes used for CRF edit word detection are the beginning of an edit (B-E), inside of an edit (I-E), each of which has a possible IP associated with it (B-E+IP or I-E+IP), and outside of an edit (O). Note that IPs are included in the target class in order to identify the internal IPs inside complex edit disfluencies. Features used in the CRF classifier for edit word detection are:

- N-grams of words and POS tags.
- Repetition information. At each word boundary, this feature represents whether there is a repeated word sequence (as many as three words) that ends at that point, with optional filler words allowed starting from that point.
- Fragment information. This feature represents whether a word is a fragment. Only in the reference transcription condition can this feature be triggered. In the speech recognition output condition, no word fragment information is available.
- Filler words. This feature represents whether there is a predefined filler phrase after a word boundary. This is not from the filler word detection results; rather, a list of cue words is used.
- Prosody posterior probabilities. This is obtained from a prosody model trained for IP detection.

Additional details about the models and features used for the various MDE tasks are provided in [Liu04].

2.2.3 Baseline Results for SU and Edit Detection

For this workshop our focus is on CTS data; therefore, we present system performance for that corpus only. The models were trained using the RT'04F training set (roughly 400,000 words of MDE annotated Switchboard transcripts) and evaluated on three additional sections of largely Fisher (with some Switchboard) data that were RT'04F SimpleMDE annotated by the LDC: a 75,000 word development set (dev1); a second 35,000 word development set (dev2); and a 35,000 word evaluation set (eval). See Section 3.2.1 for more details on these three metadata annotated sets and their corresponding treebanks.

Table 1 shows the results for SU and edit detection. All the results were obtained using the NIST MDE scoring tool mdeval-v20 [Fal04]. For SU detection, we report results using CRF (which in most cases performs the best among the three approaches when used alone), and also our baseline system involving the combination of HMM and MaxEnt. Since posterior probability estimation is unavailable (from the Viterbi decoding) using CRF, we generate posterior probabilities using this combination model, which at times outperforms the single best CRF approach. Additionally, this model has the advantage of being likely to generate more accurate probability estimations, since HMM and MaxEnt models have

RT'04 Set	dev1		dev2		eval	
Transcript	REF	ASR	REF	ASR	REF	ASR
Data Size	6 hours		3 hours		3 hours	
ASR WER	0	18.8	0	11.7	0	14.9
SU Detection:						
CRF	26.69 (35.70)	40.41 (50.88)	25.88 (35.45)	35.80 (45.66)	26.55 (37.64)	37.25 (47.77)
HMM+MaxEnt	26.48 (37.28)	40.62 (51.35)	26.14 (37.52)	35.06 (46.61)	26.96 (38.55)	36.68 (48.36)
Edit Detection:						
CRF	53.01	82.84	53.39	76.03	50.07	79.77

Table 1: MDE baseline SU and edit detection results. For SU detection, we report the boundary detection results (insertion and deletion errors) along with the total error rate including substitution errors in parenthesis for both reference (REF) and speech recognition (ASR) transcripts.

different error patterns. This is critical to our SU reranking experiments presented in Section 6; posterior probabilities from the MDE baseline system are needed to extract alternative hypotheses for reranking and as an important feature for the reranker. Since we do not perform reranking experiments on edits, results on edit detection are obtained using the CRF edit detection model.

3 Parsing Metrics and Impacting Factors

In this section, we report on a number of experiments evaluating the impact of structural metadata, in particular edit, SU detection, and filler word detection on parsing accuracy. These experiments are coupled with the development and evaluation of the SParseval evaluation suite[2]. This suite is described first, followed by several experiments that investigate the factors impacting the accuracy of parse scores, not to mention the sensitivity of the various scoring methods to word error and metadata error. We also discuss a set of preliminary experiments on the impact of fillers on parsing.

3.1 SParseval Scoring Tool

3.1.1 Motivation for SParseval

Natural language parsing technology was originally developed for textual sources. Initially corpora used for training statistical parsers were based on a punctuated text with correct words and case information [MSM93]. Not surprisingly, all of the methods developed for evaluating these statistical parsers have used sentence-level parse scoring (e.g., [SC97, BAF$^+$91]). Now that parsers are being applied in spoken domains such as Switchboard conversational telephone speech [GHM92], the assumption that the parser has access to "perfect" word transcripts and sentence boundaries is called into question. Although parsers have been evaluated on Switchboard, they initially utilized human transcripts and reference sentence segmentations (e.g., [CJ01]). Although parsing has been done in the face of potentially imperfect sentence boundaries in [KOC04], their use of reference transcripts enabled alignment with the gold standard words for bracket-based scoring.

As the NLP and speech processing communities are converging to work on spoken language processing, parsing techniques are now being applied to speech recognition transcripts with word errors and potentially imperfect sentence boundaries provided by automatic methods (e.g., [LSS$^+$05]). This creates the need to develop and evaluate new methods for determining spoken parse accuracy that support evaluation under these challenging conditions. Standard methods, such as Parseval metrics, for evaluating parse accuracy

[2]This work has benefited from collaboration with Jeremy Kahn, Mari Ostendorf, Mark Johnson, and Eugene Charniak.

assume that the yield of a tree output by a parser is identical to that of a gold-standard reference tree; however, this assumption will often be violated when parsing ASR outputs due to both word and sentence boundary errors.

This section describes the SParseval scoring tool that was developed by the *Parsing and Spoken Structural Event Detection* team at the 2005 CLSP Johns Hopkins Summer Workshop in order to cope with the uncertainties of speech when evaluating spoken language parsing performance. It 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 dependency-based procedure maps each tree to a dependency graph and then evaluates precision and recall on the edges of the graph. This effort builds on the insights from the parsing metrics literature (e.g., [Car98, CFL$^+$02, SC97, BAF$^+$91]).

```
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.

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 for a span of speech and the sentence segmentation that would be used to create the gold standard parses (marked as ||), and the second line represents the system input that the parser would be given to produce parses. This input contains 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 (as a result of automatic speech recognition errors), together with different sentence segmentations, make the span-based measures proposed in [BAF$^+$91] difficult to apply. However scoring can proceed if we create a super tree for the gold and test inputs over the entire speech chunk as in [KOC04], 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.

The SParseval tool was implemented in C and supports both speech-based bracket and head dependency scoring at the level of a demarcated chunk of speech such as a conversation side, as well as 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. We chose to support both bracket and head dependency scoring in order to investigate whether they are affected differently by word and segmentation errors. The tool 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 involve open class content words. In this initial effort, we specified the tags associated with the closed class words in the tool's parameter file so that dependencies involving these words could be ignored when calculating the open class dependency score. Since a parser may be embedded in many different spoken language applications, it is important to explore the relationship between a variety of metrics and task specific performance goals.

3.1.2 SParseval

The scoring tool runs on the command line in Unix by invoking the *sparseval* executable with several flags to control the scoring functionality:

- *-p file* is used to provide the evaluation parameter file, which is similar to that used by *evalb* [SC97]. Note that SPEECHPAR.prm is the parameter file that comes with the tool. It specifies the list of labels to be ignored (e.g., S1, TOP), the list of labels corresponding to empty nodes to be removed from trees prior to evaluation, an indication of words, tags, and filled pause forms that are considered equivalent, and the list of closed class tags. It also contains a list of edit labels which are ignored for span calculations of other constituents following [CJ01].

- *-c* indicates that evaluation is to be done on a speech chunk basis rather than at the sentence level. If this flag is not included, the parser assumes that the yields of the gold and test parses are identical. When this flag is set, it is assumed that all of the gold parses associated with the chunk appear together in a single file, and similarly for the test parses.

- *-h file* specifies the head percolation rule file used to determine the head dependency score. The parses (gold and test) are converted to a head dependency format by using the heuristic rules in the table to identify constituent heads to support head word percolation and extraction of head dependencies. We have evaluated three different sets of percolation rules: two of the tables, Charniak's and Collins' head percolation tables, were developed for parsing the Penn Treebank (Charniak and Collins), but Hwa's table was focused on generating more semantically rich head dependencies. Hwa's table was developed for studies reported in [HRW$^+$05], for which the goal was to project syntactic dependencies from a resource rich language to a resource poor language to bootstrap a parser for the latter language when there is parallel text in the two languages.

- *-a file* specifies the string alignment file used. For the workshop, we generated alignment files using SCLite [Fis01] and a simple Perl formatting script.

- *-b* indicates that no alignment will be used. This means that a bracketing score will not be calculated, and only a bag of head dependencies score will be produced over the span of speech. Note that there are temporal no-cross-over constraints on matching dependencies that prevents dependencies that are not temporally near each other from matching.

- *-u* is a flag that indicates that the bracket or dependency labels should be ignored when scoring.

- *-v* is used to request verbose output from the scoring tool, i.e., it provides scores for each chunk of speech evaluated in addition to the summary over all chunks.

- *-l* is used to indicate that input files will be used to specify lists of the gold and test files to be scored.

- *-F file* specifies the output file.

For example, the following command produces a bag-of-dependencies score for the files in `test`.

```
sparseval -l -p SPEECHPAR.prm -h headPerc -c -b gold test -F output
```

To perform bracket based scoring, it is necessary to supply a list of alignment files as shown in the following command, which produces both a bracket-based and an aligned dependency score.

```
sparseval -l -p SPEECHPAR.prm -h headPerc -c -a align gold test -F output
```

3.2 Experimental Evaluation of SParseval and Impacting Factors

We have evaluated several parsers under a variety of conditions in order to develop a deeper understanding of how the factors impacting parsing performance on speech affect our metrics. We have investigated these factors using two conversational speech tree banks. The first is the Switchboard Treebank corpus and the second is a new resource that was developed together with LDC for the workshop, a treebank for the RT'04F SimpleMDE dev1, dev2, and eval sets.

3.2.1 Data Resources

Due to our dual goals of investigating factors impacting the parse accuracy of conversational speech and the effect of syntactic and other knowledge sources on improving MDE detection accuracy, it was very important for the team to have access to a unified conversational speech resource with consistent metadata markups and parse trees. Although the Switchboard Penn Treebank is a very useful resource for our experiments (especially for parser training), there were a number of reasons for developing a new resource. First, the experiments required access to state-of-the-art speech recognition output, and we had access to scored transcripts from state-of-the-art recognizers for the EARS dev1, dev2, and eval sets. Second, the RT'04F SimpleMDE data is annotated with metadata that was used in the RT'04 benchmark tests. Using this new data allowed us to evaluate our MDE systems against a very strong baseline, a state-of-the-art RT'04 MDE system described in Section 2. Third, the Switchboard Penn Treebank is not consistent with the RT'04F SimpleMDE specification. Although we could have attempted to map the trees to this specification, the results of our experiments would have been impacted by the bias introduced by the mapping step.

Data Set	# Conversations	# SUs	# Words
dev1	72	11k	71k
dev2	36	5k	35k
eval	36	5k	34k

Table 2: Attributes of the RT'04 data sets.

The data used to develop the treebank was drawn from transcribed English conversational telephone speech originally developed for the EARS Program. As part of EARS, this data was carefully transcribed and annotated for structural metadata extraction (MDE). LDC defined several versions of an MDE annotation task for EARS; this corpus was annotated according to SimpleMDE V6.2 [Str04], which contains the following elements: Fillers (including filled pauses and discourse markers), Edit Disfluencies (repetitions, revisions and restarts), and SUs (syntactic/semantic units).

Table 2 presents information on the size of three EARS RT'04F data sets that were treebanked (i.e., number of conversations, number of sentences (i.e., SUs), and number of words). Note that eval is the RT'04 evaluation data set, dev1 is a combination of the RT'03 MDE development and evaluation sets used as a development set for RT'04, and dev2 is a new development set created for RT'04. The data comprised a total of 144 conversations or 140,000 words, representing 21,000 SUs. Using the existing MDE annotations as a starting point, the data was then annotated for syntactic structure. Treebank annotation was performed in accordance with existing guidelines for treebanking conversational telephone speech [BFKM95, Tay95].

Manual treebanking was preceded by generation of automatic parse trees[3]. The first challenge was to convert the existing MDE annotations in RTTM format (a format developed by NIST for the EARS Program that labels each token in the reference transcript according to the properties it displays (e.g., lexeme versus non-lexeme, edit, filler, SU)) into a format appropriate for the parser such that it would generate accurate parses in a form that would require as little hand modification by the treebank team as possible. To establish correspondence between treebank tokens and MDE tokens, unique tokens IDs from the original MDE annotation files were mapped to the treebank files. The two token sequences were aligned using a simple algorithm to obtain a mapping. Using this mapping, each treebank token of the form (part-of-speech word) was extended to (part-of-speech: [MDE_ID] word), thus creating a consistent mapping.

The initial parse trees were produced using the Charniak parser [Cha00], trained on all of Switchboard and supplemented with Wall Street Journal data (with a 4-1 ratio). We cleaned up the Switchboard trees prior to training the parser. By hand, we fixed a couple of errors in the treebank that caused problems for training the parser (e.g., missing top level parentheses, a preterminal used as a non-terminal). We also developed a perl script to clean up the parse trees used for training, which:

- removed CODE lines,
- promoted children of TYPO and then removed the TYPO bracket,
- removed XX constituents and edited constituents,
- removed DFL, IP, RM, and RS constituents,
- for `A|B` and `A^B` non-terminal constructions, we kept the `A` and discarded the `B` alternative,
- removed remaining carets on non-terminals (e.g., `^A` became `A`).

Parse trees were supplied for each side of the conversation, and each tree corresponded to a single SU as defined by the MDE annotation. Prior to parsing, we tokenized the words in the SU using LDC's tokenizer script (`http://www.cis.upenn.edu/~treebank/tokenizer.sed`), and then the edited portion of each disfluency was separated from the rest of the sentence using the MDE annotations. The cleaned up sentence was parsed, edits for that sentence were then parsed separately, and finally the parses for the edits were inserted back into the parse for the rest of the sentence. Special treatment was required for regions unannotated for MDE, complex edits, and SUs that were comprised solely of edited material. The resulting parses were then augmented with function tags using a modification of the Bikel parsing engine [Bik04] to add semantic function information without altering the parse provided in a novel use of the constraint system built into the parser. This improved the parses given to the annotators, and decreased the number of manual corrections necessary.

One additional challenge was the existence of discrepancies between the MDE and treebank treatments of structural metadata. For instance, when tokenization differences emerged it was necessary to follow treebank guidelines so that the correct constituents were produced, but we were also careful to maintain alignment between MDE and the original parses, particularly for the tokenization of punctuation and clitics such as don't. Differences emerged in the treatment of nested disfluencies, which are permitted by treebank but were not annotated as part of the SimpleMDE task, so these nested elements were added during treebank annotation. Differences also emerged in the treatment of filler annotation. This issue

[3]We would like to acknowledge Jeremy Kahn for providing advice and scripts that were adapted to generate the parse trees provided to the LDC treebanking team.

will be discussed in Section 3.3.1. This treebanking effort led to a revision of the RT'04F RTTM data because of the discrepancy resolution (i.e., some edit annotation and tokenization changes were made to the RTTM data). LDC has released the RT'04F treebank and the revised RTTM dataset as a combined dataset, LDC2005E79.

3.2.2 Experimental Setup

We have investigated the sensitivity of the parsing metrics to a variety of factors that potentially impact parse accuracy on speech. The first study was carried out by applying our parse scoring tool to parses generated by three different parsers [Bik04, Cha00, Roa01] under a variety of conditions. We chose to investigate parse metrics across parsers to avoid the potential bias that could be introduced by investigating only one. Bikel's parsing engine [Bik04] was developed to be reconfigurable based on the settings used. In these experiments, we used the settings for Collins' model II [Col99], which is a generative, lexicalized PCFG parsing model with distance constraints and parameters to model probability distributions over subcategorization frames for headwords in order to make complement/adjunct distinctions. Charniak's parser [Cha00] is a lexicalized bottom-up PCFG model that uses of a "maximum-entropy-inspired" model for conditioning and smoothing in order to enable the use of a variety of different conditioning events. Roark's parser uses a beam-search with an incremental (left-to-right), top-down derivation strategy, and relies upon the fully connected left-context of each competing derivation to condition the probabilities of subsequent rule expansions. The resulting model allows for very effective pruning strategies, leading to a fast, robust, and quite accurate parsing of text and speech. Charniak's and Roark's parsers were trained on the entire Switchboard corpus with dev1 as a development set; whereas, Bikel's parser was trained on the combination of the two sets. All three parsers were then evaluated on the RT'04 dev2 set under a variety of conditions.

The gold standard dev2 parses were produced by LDC, as described in Section 3.2.1. The dependent variable used in this investigation is F-measure (although in our tables, we also report recall and precision). The independent variables other than parser type can be placed into three categories: data quality, data use, and metric factors. The two data quality conditions investigated are:

1. **Transcript Quality:** the input to the parser was either a human transcript or a transcript output by a state of the art speech recognizer. In this initial study, we used the speech transcripts generated by a high quality RT'04 ASR system, the IBM+SRI system [KMP$^+$04], which had a WER of 11.7% on the dev2 set.

2. **Metadata Quality:** the structural metadata was based on human metadata annotations or was produced by the state-of-the-art RT'04 metadata system [LSS$^+$05] described in Section 2.2. The performance of this system appears in Table 1.

These two data quality factors produce four conditions: reference transcripts and reference metadata, reference transcripts and system metadata, ASR transcripts and reference metadata, and ASR transcripts and system metadata, which are depicted in Figure 3.

	Reference Transcripts	ASR Transcripts
Reference Metadata	Best Case for Parser	Impact of Word Error
System Metadata	Impact of Metadata Error	Worst Case for Parser

Table 3: The four data quality conditions to be examined in these parsing studies.

In order to generate the reference metadata for the words in ASR transcripts, the following procedure was developed. Note we adopted this procedure due to the availability of existing tools. The scoring tool is run once to perform the mapping, and a second time to evaluate the quality of the mapping.

- Score the metadata output in RTTM format against the reference annotations using mdeval-v19a, which generates ASR word onsets/offsets aligned to the reference transcripts.
- Given reference metadata onset/offset information, we mark each ASR lexeme with the type of any metadata events that cover the lexeme's aligned time span. According to RT'04F evaluation guidelines, a lexeme is covered by a metadata event if its midpoint lies within the event time span.
- We generate metadata events from this, where the event onset/offset time of the metadata event corresponds to the onset of the first word it covers and the offset of the last word it covers. When we score the ASR words and aligned metadata events against the reference, some projected events may not cover any of the ASR words prior to alignment. The scoring tool is implemented to discard such events and then perform the alignment. While such instances do occur, they are infrequent.
- In the final step, we compare the original ASR output to the words in the alignment output (generated by mdeval-v19a in the previous steps) and resolved some of the mismatches (that arise largely due to text normalization issues).

Given the reference tags associated with each word boundary of the ASR output, RTTM files are generated and evaluated against the original reference using mdeval-v19a. Table 4 shows the NIST SU error rate for the three data sets. Most of the errors are the result of word deletion errors, in which case, there is no way to map the reference tags to corresponding words. As the word error rate on the ASR condition improves, this SU error rate would typically be expected to decrease.

	dev1	dev2	eval
SU	10.72	5.37	5.03

Table 4: SU error rate when mapping reference metadata onto the ASR words

In addition to the two data quality conditions, there was one data use condition:

1. **Use of Edit Metadata:** either the parsers use the metadata indicating the location and extent of the edited regions to remove that material prior to parsing, or they do not (and so they parse edits together with the rest). It should be noted that the parsers were trained differently in these two conditions. In the former case, the parsers were trained on parses from which edited subtrees were removed. In the latter case, the parsers were trained with the edited subtrees left in place, so that the structure of edits could be learned to some extent[4]. Parse accuracy was evaluated using "relaxed edited scoring"[CJ01], i.e., we ignore the internal structure, contiguous boundaries, and attachment of EDITED constituents.

The four metric factors are enumerated below:

1. **Parse Match Representation:** The parses are scored based on the bracket-based, governor head dependency, or open class dependency representation supported by SParseval.
2. **Labeling:** The parses are either labeled or unlabeled. For unlabeled brackets, this means that a bracket is scored as correct if it matches a gold bracket without consideration of the label. For dependencies, this implies that the label of the relation in `(dependent word, relation, head word)` is not considered when matching the dependencies to the gold standard. This factor has been investigated for text-based parsing, but not in the presence of transcript and sentence segmentation errors.

[4]CFGs cannot represent cross-serial dependencies. The relative inability of of PCFGs to identify edit words in comparison with more specialized techniques has been shown in [KLC$^+$05]. Note though that PCFGs recover edit words with around a 65% F-measure, so they are able to help with detection despite their lack of power.

3. **Alignment:** Word alignments are needed to score brackets in the ASR transcript condition. On the other hand, the dependencies can be scored either with or without alignment. When alignment is used, it is an added constraint on the match between the gold and test dependencies. Note that SParseval does not allow arbitrary matching in the non-aligned case; it does not allow dependencies that crossover other matched dependencies.

4. **Head Percolation Table:** For the dependencies to be extracted from the test and gold standard parses, a set of head percolation rules is used. We evaluated three head percolation tables and their impact on parse accuracy, namely the head table used by Charniak's parser, that used by Collins' parser, and a head table developed by Rebecca Hwa to produce relations between semantically more important words in the parse [HRW$^+$05].

3.2.3 Experimental Results

See Section A in the appendix for a set of tables providing the parse scores over all 72 conversation sides for each scoring condition and parser. The results for the Bikel parser appear in Tables 33, 34, 35, and 36. The Charniak parser results appear in Tables 37, 38, 39, 40. Results from Roark's parser appear in Tables 41, 42, 43, and 44.

It should be noted that to score brackets in SParseval, the words in a conversation side being parsed for evaluation must be aligned with the words in the gold standard conversation side prior to scoring; hence, the effect of using alignment or not can only be evaluated for the dependency metrics. Similarly, head percolation is a factor only when evaluating dependency scores. Hence, we have performed two analyses of this data: the first is a factorial analysis of variance involving the scoring of aligned parses only and ignoring head percolation rules in order to examine the significance of the factors impacting dependency versus bracket F-measure scores; the second is a factorial analysis of variance over dependencies only in order to investigate the effects of alignment and head percolation table on dependency F-measure scores.

Analysis I: Scores Obtained with SCLite Alignments. F-measure for the 72 conversation sides of the dev2 set was entered into a 3(Parser) $\times$ 2(Transcript Quality) $\times$ 2(Metadata Quality) $\times$ 2(Use of Edit Metadata) $\times$ 3(Parse Match Representation) $\times$ 2(Labeling) analysis of variance on scores obtained with alignment and collapsing over head percolation table [5]. Enumerated below are the significant main effects:

- **Parser:** There is a significant main effect of parser, $F(2,78) = 252.76, p < .0001$. Using a post hoc Bonferroni t-test, we found that Charniak's parser obtained a significantly greater F-measure (75.91) than Bikel's parser (75.43), $p < .001$, which is significantly greater than Roark's parser (73.67), $p < .0001$. Although there are differences among the parsers, our purpose in investigating these three parsers was to avoid parser bias in this experiment on parse metrics and impacting data quality factors.

- **Transcript Quality:** There is a significant main effect of transcript quality, $F(1,78) = 19,127.6, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses on human transcripts had a significantly greater F-measure (81.05) than those based on ASR transcripts (68.95), $p < .0001$. Clearly, the word error produced by the ASR system causes an overall significant degradation in parse quality relative to reference transcripts.

[5] An analysis of variance (ANOVA) is a statistical technique which compares means by splitting the overall observed variance into different parts. It enables us to investigate not only the effect of individual factors but also their interactions. It should be noted that values for specific conditions reported in figures and in post hoc analyses represent the average over all scores given the specified conditions.

- **Metadata Quality:** There is a significant main effect of structural metadata quality, $F(1,78) = 7,507.85, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses involving human metadata annotations had a significantly greater F-measure (78.20) than those based on our baseline metadata system (71.80), $p < .0001$. The effect of metadata error from our baseline system does cause an overall significant reduction in F-measure relative to the reference metadata.

- **Use of Edit Metadata:** There is a significant main effect of using the edit structural metadata to remove edited material prior to parsing, $F(1,78) = 1,335.89, p < .0001$. A post hoc Bonferroni t-test indicates that parses for which edits were removed prior to parsing had a significantly greater F-measure (76.49) than those that parsed everything together (73.51), $p < .0001$.

- **Parse Match Representation:** There is a significant main effect of the use of parse match representation, $F(2,78) = 5.61, p < .005$. Using a post hoc Bonferroni t-test, we found that the open class dependency F-measure score (75.14) is slightly, though significantly, larger than the head dependency F-measure score (74.88), $p < .005$. Bracket scores (74.93) do not differ significantly from the other two scores.

- **Labeling:** There is a significant main effect of scoring with or without label match, $F(1,78) = 1,809.81, p < .0001$. Using a post hoc Bonferroni t-test, we found that scoring without labels produced a greater F-measure score (76.89) than scoring with them (73.11), $p < .0001$. This result is not surprising since the former condition relaxes the label match required for correctness in the latter case.

There are several interesting interactions involving only data quality and data use factors:

- **Transcript Quality $\times$ Metadata Quality:** There is a significant interaction between transcript and metadata quality, $F(1,78) = 64.36, p < .0001$, which can be seen in Figure 2. Use of reference transcripts and metadata gave a significantly higher score (84.61) than using reference transcripts and system metadata (77.49), $p < .0001$, which gave a significantly higher score than using ASR transcripts and reference metadata (71.79), $p < .0001$, which gave a significantly higher score than using ASR transcripts and system metadata (66.12), $p < .0001$. The use of system metadata reduces the F-measure when parsing reference transcripts slightly more than when parsing ASR transcripts (8.4% versus 7.9% relative reduction).

- **Metadata Quality $\times$ Use of Edit Metadata:** There is a significant interaction between metadata quality and the use of the metadata to remove edits prior to parsing, $F(1,78) = 364.19, p < .0001$, which can be seen in Figure 3. Scores are significantly higher when using edit information from reference metadata (80.44) than scores obtained not using edit information from reference metadata (75.96)[6], $p < .0001$, which are significantly higher than scores obtained using edit information and system metadata (72.53), $p < .0001$, which are significantly higher than scores obtained not using the edit information and system metadata (71.07), $p < .0001$. The increase in F-measure from removing edits prior to parsing is relatively greater when using reference metadata than system metadata (5.91% versus 2.06% relative increase in F-measure).

- **Transcript Quality $\times$ Use of Edit Metadata:** There is a significant interaction between transcript quality and the use of the metadata to remove edits prior to parsing, $F(1,78) = 51.79, p < .0001$, which can be seen in Figure 4. Scores are significantly higher when using edit information from reference transcripts (82.84) than scores obtained not using edit information from reference

[6]Keep in mind that reference SU metadata is still used in this case.

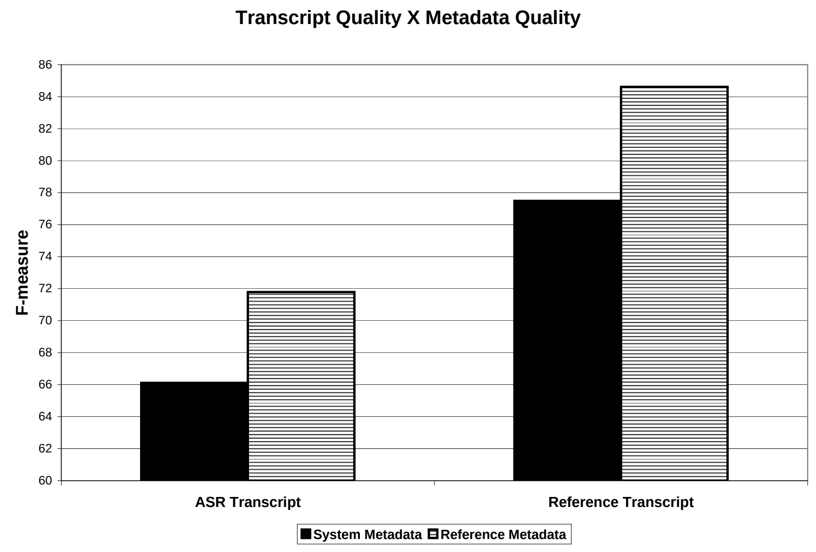

Figure 2: Average F-measure scores given transcript and metadata quality.

transcripts (79.26), $p < .0001$, which are significantly higher than scores obtained using edit information from ASR transcripts (70.13), $p < .0001$, which are significantly higher than scores obtained not using the edit information from ASR transcripts (67.77), $p < .0001$. The increase in F-measure from removing edits prior to parsing is relatively greater when using reference transcripts than when using ASR transcripts (4.5% versus 3.5% relative increase in F-measure).

- **Metadata Quality × Parser:** There is a significant interaction between the metadata quality and parser, $F(2, 78) = 3.33, p < .05$. Although each of the parsers gets significantly higher scores using reference than system metadata, as can be seen in Figure 5, the relative decrease in F-measure from using system metadata is slightly greater for Bikel's parser (8.6%) than Charniak's (8.0%) or Roark's (7.9%) parser. Although the difference between Charniak's (79.09) and Bikel's (78.81) parser is insignificant using reference metadata, Charniak's parser gets a significantly higher score (72.73) than Bikel's (72.05) using system metadata, $p < .0001$. All other differences were significant.

- **Use of Edit Metadata × Parser:** There is a significant interaction between the metadata quality and the parser, $F(2, 78) = 6.92, p < .002$, which can be seen in Figure 6. Although the difference between Charniak's (77.26) and Bikel's (77.16) parser is insignificant when using edit metadata, Charniak's parser gets a significantly higher score (74.55) than Bikel's (73.70), $p < .0001$, when the edit information is not used to remove that material prior to parsing. All other differences were significant. The relative decrease in F-measure from not using the edit metadata to remove edits prior to parsing is slightly greater for Bikel's parser (4.7%) than Charniak's (3.6%) or Roark's (3.8%).

- **Metadata Quality × Use of Edit Metadata × Parser:** There is a significant interaction among these three factors, $F(2, 78) = 4.19, p < .02$, which can be seen in Figure 7. All of the parsers gain more from utilizing edit metadata than not using it with reference metadata, and far more than when using system metadata. However, Bikel's parser benefits more from using rather than ignoring

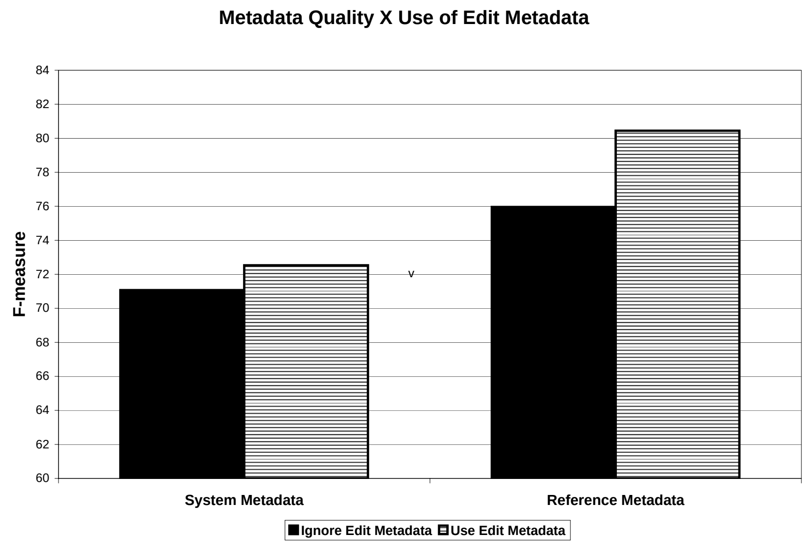

Figure 3: Average F-measure scores given metadata quality and the use of edit metadata to remove edited regions prior to parsing.

edit metadata compared to the other parsers, especially on system metadata. Charniak's parser obtains a significantly greater F-measure when ignoring system edit metadata than Bikel's parser obtains, although they do not differ significantly when using the edit metadata to remove them prior to parsing (for both reference and system metadata) or in the reference metadata condition in which edits are not removed prior to parsing.

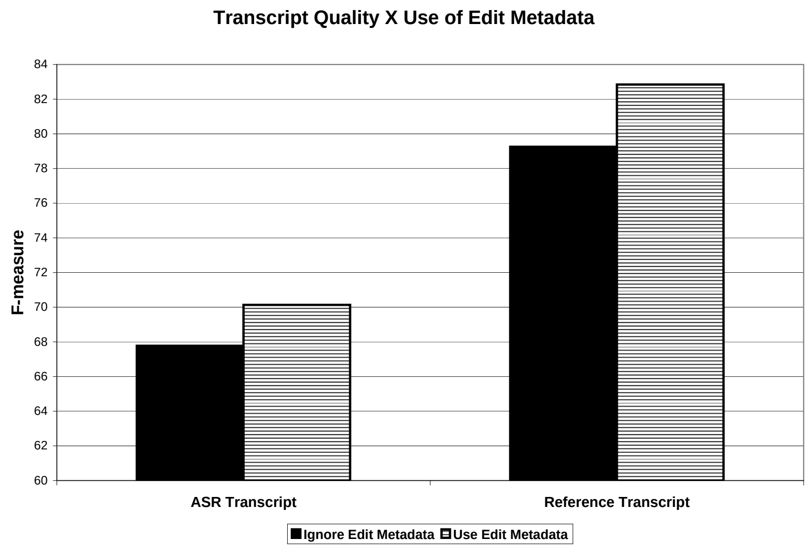

Figure 4: Average F-measure scores given transcript quality and the use of edit metadata to remove edited regions prior to parsing.

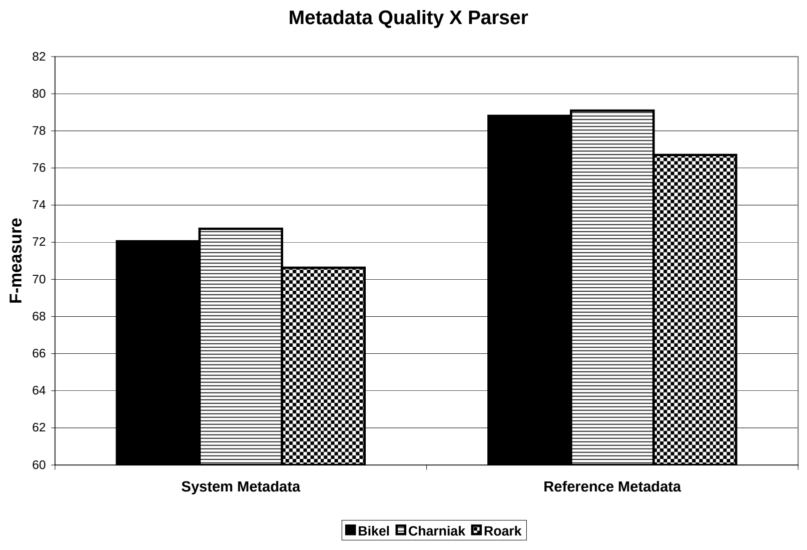

Figure 5: Average F-measure scores given metadata quality and parser.

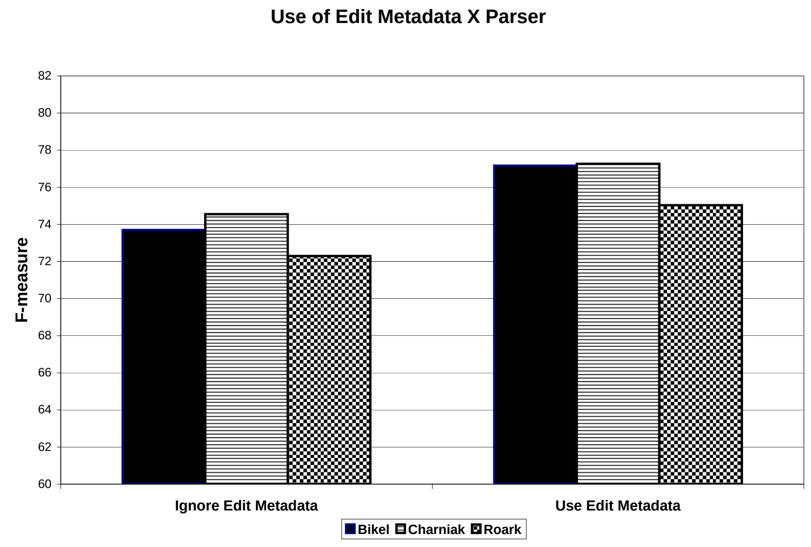

Figure 6: Average F-measure scores given use of edit metadata and parser.

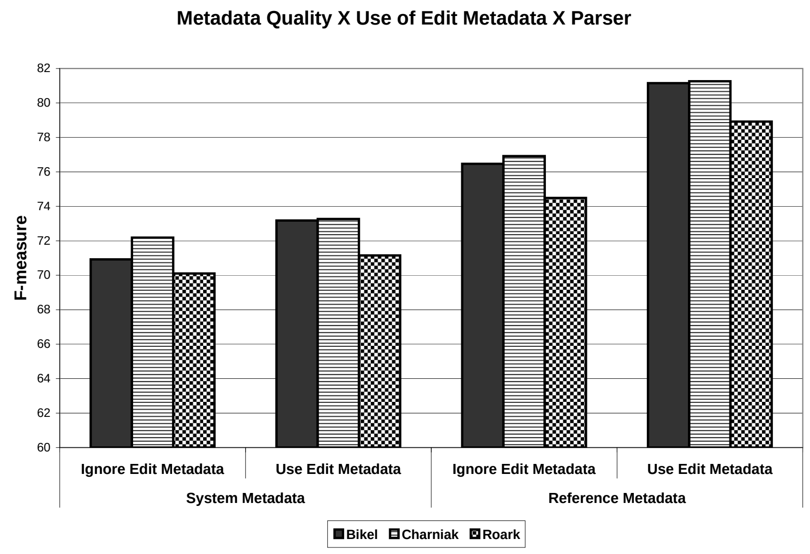

Figure 7: Average F-measure scores given metadata quality, use of edit metadata, and parser.

There are also several interesting interactions involving the parse metric factors:

- **Labeling × Parse Match Representation:** There is a significant interaction between labeling and the parse match representation used for scoring, $F(2, 78) = 13.36, p < .0001$, which can be seen in Figure 8. Ignoring labels benefits the dependency scores much more than the bracket-based scores. It is interesting to note that the dependency scores are slightly greater than the bracket scores when labels are ignored (significant $p < .05$), although the opposite trend is observed when labels are scored (head dependency scores are significantly lower than bracket scores, $p < .01$). There is a significant difference between the head dependency and open class dependency scores when scoring with labeling information; however, the difference is no longer significant when ignoring label information.

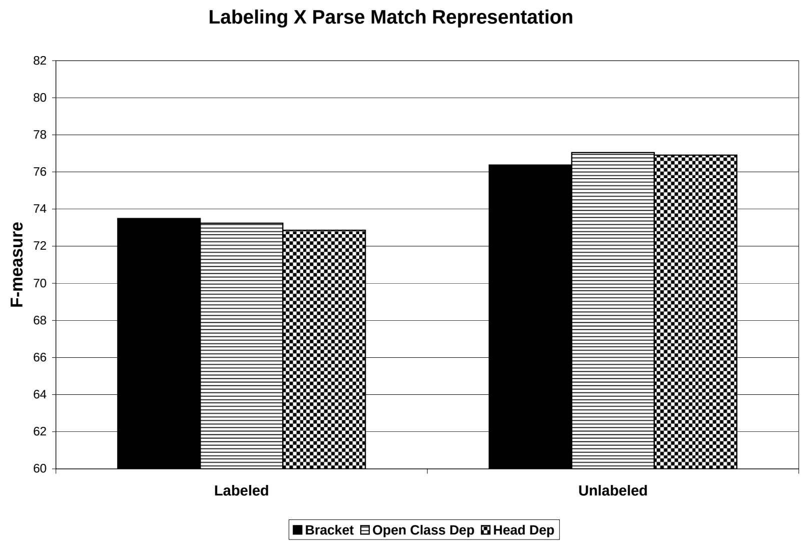

Figure 8: Average F-measure scores either using or ignoring the label during scoring given the parse match representation.

- **Metadata Quality × Parse Match Representation:** There is a significant interaction between metadata quality and parse match representation, $F(2, 78) = 246.17, p < .0001$. As can be seen in Figure 9, bracket scores are significantly greater than both the head 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 ($p < .0001$). This suggests that bracket scores are more negatively impacted by errors in sentence segmentation than their dependency counterparts. Additionally, the open class dependency scores are only significantly larger than the head dependency scores for reference metadata ($p < .01$).

- **Use of Edit Metadata × Parse Match Representation:** There is a significant interaction between parse match representation and the use of the metadata to remove edits prior to parsing, $F(2, 78) = 3.53, p < .05$. As can be seen in Figure 10, each of the parse match scores increases from the use edit metadata ($p < .0001$). However, the bracket-based scores increase slightly more from edit metadata use (4.7%) than the dependency metrics do (4.0% and 3.8%, for the head and open class dependency representations, respectively).

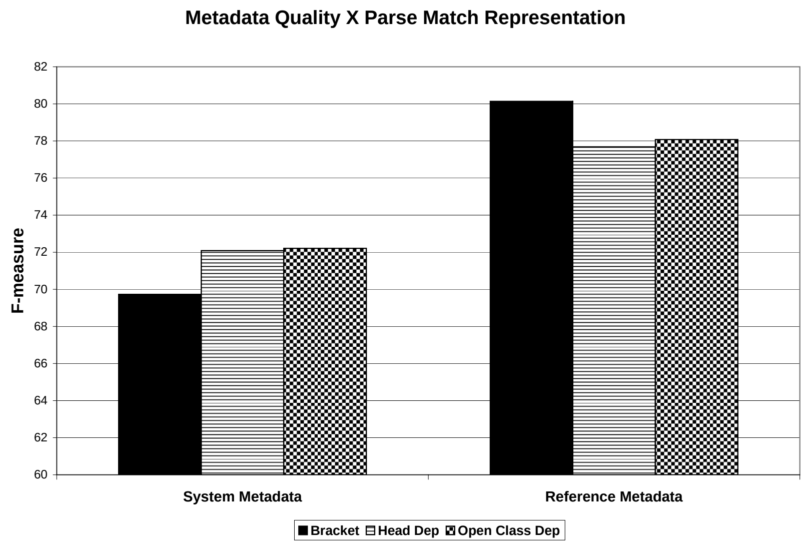

Figure 9: Average F-measure scores given metadata quality and the parse match representation.

- **Transcript Quality $\times$ Parse Match Representation:** There is a significant interaction between transcript quality and parse match representation, $F(2, 78) = 66.24, p < .0001$. As can be seen in Figure 11, all of the scores, regardless of representation, are relatively lower on ASR transcripts than on reference transcripts. The scores for dependencies are significantly larger than the bracket scores on human produced transcripts, but significantly smaller than the bracket scores on ASR transcripts, $p < .0001$. On the other hand, the dependency scores do not differ significantly from each other under either of these conditions.

- **Labeling $\times$ Transcript Quality $\times$ Parse Match Representation:** There is a significant interaction among these three factors, $F(2, 78) = 8.23, p < .0005$, which can be seen in Figure 12. Dependency F-measure scores are more negatively impacted by word errors in the ASR transcripts than the bracket scores. The degradation 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).

The impact of the data quality and use factors were, by and large, as one might predict. Reference transcripts allow a parser to produce larger parse scores than ASR transcripts, and similarly for reference metadata compared to system metadata. Use of edit metadata to remove edits prior to parsing also results in higher parse scores. There are a few interesting interactions among the data quality factors. For example, there is a significant interaction between transcript quality and metadata quality that suggests that reducing transcript quality has more of a negative impact on parse scores than reducing metadata quality. Also, it should be noted that using edit metadata to remove edit regions prior to parsing has a more positive effect on parse scores when using reference metadata than using system metadata. A similar finding was observed in the interaction between the use of edit metadata and transcript quality. It should also be noted that the data quality factors interact with parser type. There is an interaction between

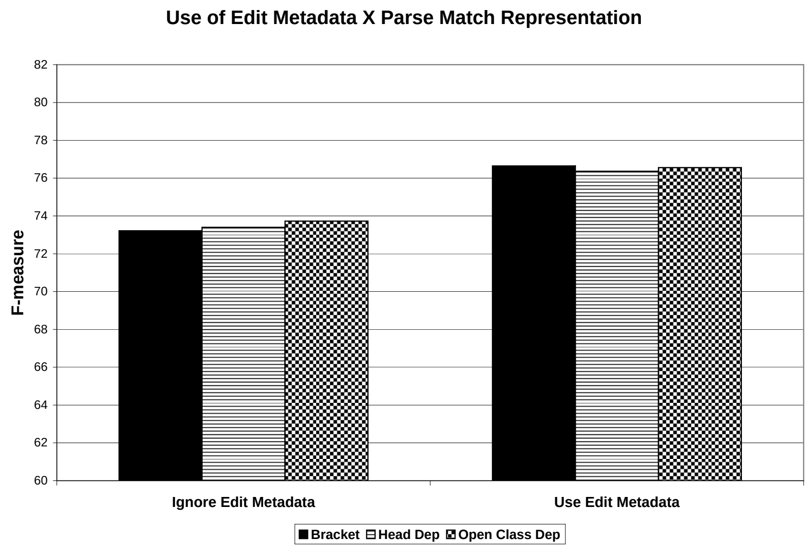

Figure 10: Average F-measure scores given the use of edit metadata to remove edited regions prior to parsing and parse match representation.

parser and metadata quality, parser and use of edit metadata, and parser and both these factors. These interactions are largely the result of the fact that Bikel's parser is relatively more negatively impacted by the use of system metadata to parse but seems to rely on using the metadata to remove edits prior to parsing, especially in the system metadata case since the relative loss from not using this information is much higher than for the other two parsers.

In calculating the correlation between the different parse match representation scores obtained with alignment, the dependency scores are on average quite similar to the bracket scores, and correlate highly with them, as can be seen in Table 5. The correlation between the open class and head dependency scores is even higher, .9926. However, as can be seen in the above analysis of variance, these dependency and bracket scores are affected differently by data quality and use factors. Ignoring labels more positively impacts dependency scores than bracket scores, but the biggest improvement comes from ignoring labels when scoring parses obtained for reference transcripts. Given that dependency scores are more negatively impacted by word error than bracket scores, bracket scores are larger than dependency scores in the ASR transcript case, regardless of whether the labels are ignored during scoring or not. The bracket scores are more sensitive than the dependency scores to the errors in segmentation resulting from the use of system metadata. Furthermore, the bracket scores benefit more from the use of edit metadata than do the dependency scores. It should be noted that parser does not interact significantly with any of the metric factors.

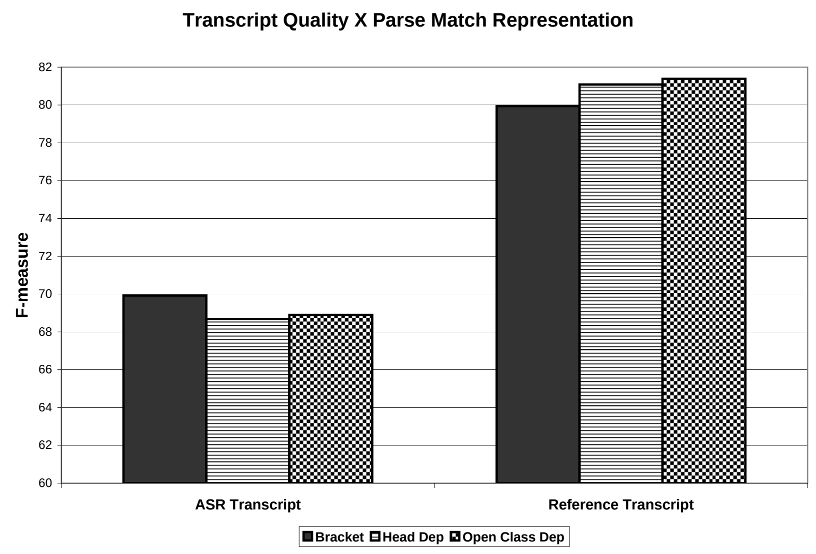

Figure 11: Average F-measure scores given transcript quality and parse match representation.

Correlation of Bracket and:	**Head Percolation Table**		
	Charniak	**Collins**	**Hwa**
Head Dep	0.8882	0.8776	0.8866
Open Class Dep	0.8860	0.8737	0.8865

Table 5: Correlations between bracket scores and aligned dependency scores given head percolation table.

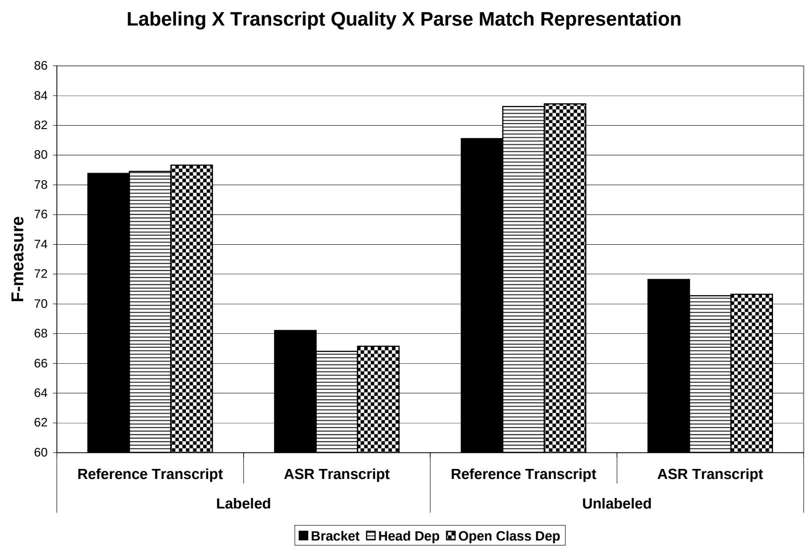

Figure 12: Average F-measure scores given labeling, transcript quality, and parse match representation.

Analysis II: Dependency Scores. F-measure for the 72 conversation sides of the dev2 set was entered into a 3(Parser) × 2(Transcript Quality) × 2(Metadata Quality) × 2(Use of Edit Metadata) × 2(Dependency Parse Match Representations) × 2(Labeling) × 2(Alignment) × 3(Head Percolation Table) analysis of variance of the dependency parse scores (i.e., bracket scores are not included in this analysis). Enumerated below are the significant main effects.

- **Parser:** There is a significant main effect of parser, $F(2, 157) = 601.13, p < .0001$. Using a post hoc Bonferroni t-test, we found that Charniak's parser obtained a significantly greater dependency F-measure scores (76.15) than the scores obtained by Bikel's parser (75.58), $p < .0001$, which were significantly greater than those from Roark's parser (73.85), $p < .0001$.

- **Transcript Quality:** There is a significant main effect of transcript quality, $F(1, 157) = 47,641.6, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses on human transcripts had a significantly greater dependency F-measure score (81.35) than those based on ASR transcripts (69.04), $p < .0001$. Clearly, the word error produced by the ASR system causes a significant degradation in parse quality.

- **Metadata Quality:** There is a significant main effect of structural metadata quality, $F(1, 157) = 10,199.9, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses involving human metadata annotations had a significantly greater F-measure (78.04) than those based on our baseline metadata system (72.35), $p < .0001$. The effect of metadata error from our baseline system does cause a significant reduction in dependency F-measure.

- **Use of Edit Metadata:** There is a significant main effect of using the edit structural metadata to remove edited material prior to parsing, $F(1, 157) = 2,419.35, p < .0001$. A post hoc Bonferroni t-test indicates that parses for which edits were removed prior to parsing had a significantly greater dependency F-measure (76.58) than those that parsed everything together (73.81), $p < .0001$.

- **Parse Match Representation:** There is a significant main effect of the use of parse match representation, $F(1, 157) = 20.16, p < .0001$. Using a post hoc Bonferroni t-test, we found that the open class dependency F-measure score (75.32) is slightly, although significantly, larger than the head dependency F-measure score (75.07), $p < .0001$.

- **Labeling:** There is a significant main effect of scoring with or without label match, $F(1, 157) = 4,825.66, p < .0001$. Using a post hoc Bonferroni t-test, we found that scoring without labels produced a greater dependency F-measure score (77.16) than scoring with them (73.24), $p < .0001$.

- **Head Percolation Table:** There is a significant main effect of the head percolation table, $F(2, 157) = 195.44, p < .0001$. Using a post hoc Bonferroni t-test, we found that Charniak's table produced significantly larger F-measure scores (75.91) than those based on Collin's table (75.14), which were larger than those produced using Hwa's table (74.54), $p < .0001$.

- **Alignment:** There is a significant main effect of alignment, $F(1, 157) = 43.14, p < .0001$. It should be noted that alignments were required to obtain bracketing scores in the first analysis, however, for this second analysis, the dependencies could be matched either with word alignment or not. Using a post hoc Bonferroni t-test, we found that scores obtained without the alignment scoring constraint are slightly, although significantly, greater (75.38) than those obtained with alignment (75.01), $p < .0001$.

There were several interesting interactions involving data quality and data use factors. By and large these trends are quite similar to the analysis carried out with alignment and across all three parse match representations, and so we report the results without displaying figures unless there is some difference.

- **Transcript Quality $\times$ Metadata Quality:** There is a significant interaction between transcript and metadata quality, $F(1, 157) = 188.34, p < .0001$. Use of reference transcripts and metadata gave a significantly higher score (84.59) than using reference transcripts and system metadata (78.12), $p < .0001$, which gave a significantly higher score than using ASR transcripts and reference metadata (71.50), $p < .0001$, which gave a significantly higher score than using ASR transcripts and system metadata (66.58), $p < .0001$. The use of system metadata reduces the dependency F-measure when parsing reference transcripts slightly more than when parsing ASR transcripts (7.6% versus 6.9% relative reduction).

- **Metadata Quality $\times$ Use of Edit Metadata:** There is a significant interaction between metadata quality and the use of the metadata to remove edits prior to parsing, $F(1, 157) = 599.80, p < .0001$. Scores are significantly higher when using edit information from reference metadata (80.12) than scores obtained without using edit information from reference metadata (75.97), $p < .0001$, which are significantly higher than scores obtained using edit information from system metadata (73.05), $p < .0001$, which are significantly higher than scores obtained without using the edit information from system metadata (71.65), $p < .0001$. The increase in dependency F-measure from removing edits prior to parsing is relatively greater when using reference metadata than when using system metadata (5.5% versus 1.9% relative increase in F-measure).

- **Transcript Quality $\times$ Use of Edit Metadata:** There is a significant interaction between transcript quality and the use of the metadata to remove edits prior to parsing, $F(1, 157) = 122.67, p < .0001$. Scores are significantly higher when using edit information from reference transcripts (83.05) than those obtained without using edit information from reference transcripts (79.65), $p < .0001$, which are significantly higher than those obtained using edit information from ASR transcripts (70.12), $p < .0001$, which are significantly higher than scores obtained without using the edit information from ASR transcripts (67.97), $p < .0001$. The increase in F-measure from removing edits prior to parsing is relatively greater when using reference transcripts than ASR transcripts (4.3% versus 3.2% relative increase in F-measure).

- **Metadata Quality $\times$ Parser:** There is a significant interaction between metadata quality and parser, $F(2, 157) = 18.31, p < .0001$. Although each of the parsers gets significantly higher scores using reference than system metadata, the relative decrease in F-measure from using system rather than reference metadata is slightly greater for Bikel's parser (7.8%) than Charniak's (7.0%) or Roark's (7.0%) parser. Although the difference between Charniak's (78.94) and Bikel's (78.67) parser is insignificant using reference metadata, all other differences are significant, in particular, Charniak's parser gets a significantly higher score (73.37) than Bikel's (72.50) using system metadata, $p < .0001$.

- **Use of Edit Metadata $\times$ Parser:** There is a significant interaction between use of edit metadata and parser, $F(2, 157) = 24.28, p < .0001$. The relative decrease in F-measure from not using the edit metadata to remove edits prior to parsing is slightly greater for Bikel's parser (4.3%) than Charniak's (3.2%) or Roark's (3.3%) parsers. Although the difference between Charniak's (77.40) and Bikel's (77.25) parser is insignificant when using edit metadata, Charniak's parser gets a significantly higher score (74.91) than Bikel's (73.92), $p < .0001$, when the edit information is not used to remove that material prior to parsing.

- **Metadata Quality $\times$ Use of Edit Metadata $\times$ Parser:** There is a significant interaction among these three factors, $F(2, 157) = 7.49, p < 0.001$. All of the parsers gain more from utilizing edit metadata than not using it, and gain more from reference metadata than from system metadata. For system metadata, the relative decrease in F-measure from not using the edit metadata to remove edits prior to parsing is greater for Bikel's parser (3.1%) than Charniak's (1.4%) or Roark's (1.3%). For reference metadata, the relative decrease in F-measure from not using the edit metadata to remove edits prior to parsing is slightly greater for Bikel's parser (5.4%) than Charniak's (4.9%) or Roark's (5.2%). Clearly, Bikel's parser benefits relatively more from using rather than ignoring edit metadata compared to the other parsers, especially for the system metadata.

- **Transcript Quality $\times$ Metadata Quality $\times$ Use of Edit Metadata:** There is a significant interaction among these three factors, $F(1, 157) = 4.27, p < .05$, in contrast to the analysis comparing bracket and dependency scores. As can be seen in Figure 13, the largest F-measure scores are obtained when parsing reference transcripts with reference metadata and using that metadata to remove edits prior to parsing. When transcript and metadata quality is reduced by using system outputs, parse accuracy drops. The relative gain from using edit metadata relative to not using it is at the highest level when using reference transcripts and metadata (6%), followed by using ASR transcripts and reference metadata (4.9%). Use of system metadata results in lower improvements from removing edits, with reference transcripts giving a slightly better improvement (2.5%) than ASR transcripts (1.3%). Note that all pairwise differences are significant.

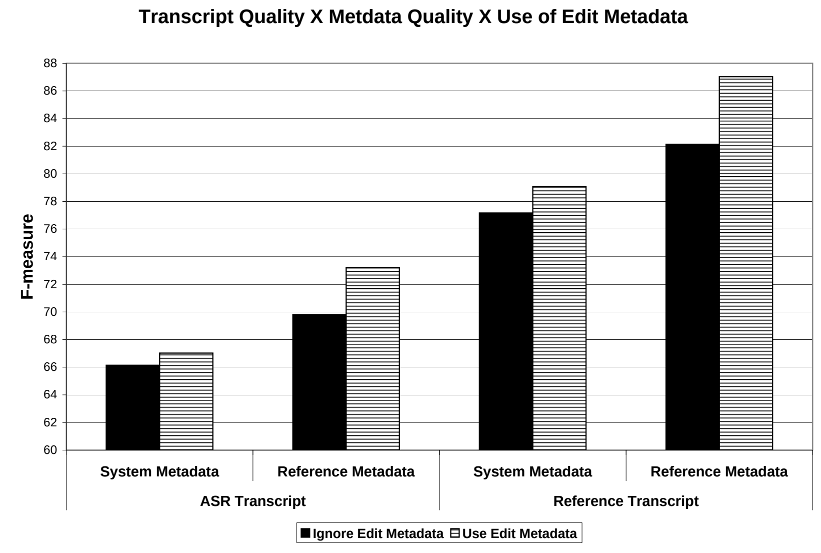

Figure 13: Average F-measure scores given transcript quality, metadata quality, and the use of edit metadata.

There were also several interesting significant interactions involving factors related to our metrics.

- **Transcript Quality $\times$ Alignment:** There is a significant interaction between the transcript quality used and alignment, $F(1, 157) = 5.95, p < .05$, which can be seen in Figure 14. Scoring with a word alignment constraint for matching dependencies results in slightly lower dependency F-measure scores than those obtained without alignment; however, relaxing the alignment scoring constraint benefits the parsing of ASR transcripts (.7%) slightly more than parsing with reference transcripts (.3%). Not aligning in the ASR condition gives a greater opportunity for a match to occur.

- **Use of Edit Metadata $\times$ Alignment:** There is a significant interaction between use of edit metadata and alignment, $F(1, 157) = 4.83, p < .05$, which can be seen in Figure 15. Relaxing the

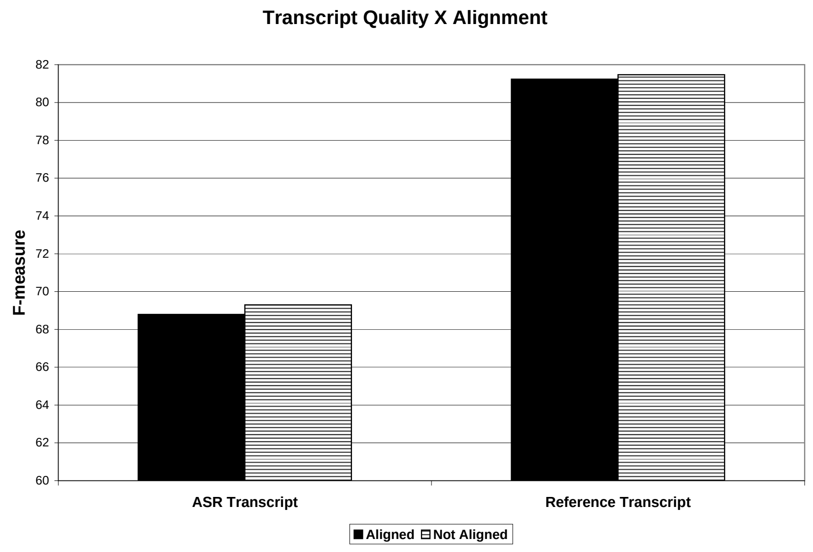

Figure 14: Average F-measure scores given transcript quality and alignment.

alignment constraint benefits parse scores when edits are left in the word string to be parsed (.7%) more than when the edits are removed (.3%). This suggests that the edited region provides an opportunity for obtaining additional dependencies when word alignment is not carried out.

- **Transcript Quality $\times$ Head Percolation Table:** There is a significant interaction between transcript quality and the head percolation table used to extract the dependencies for scoring, $F(2, 157) = 20.46, p < .0001$, which appears in Figure 16. In general, using Charniak's head table gives higher dependency F-measure scores than Collin's table, which results in higher scores than Hwa's table. This trend is clearly found with reference transcripts; however, for ASR transcripts, there is no significant difference between using Collin's and Hwa's table, although using Charniak's table still produces higher scores. Using Collin's table results in a greater relative loss in going from reference transcripts to ASR transcripts (15.71%) than Charniak's (14.80%) and Hwa's (14.87%) tables.

- **Labeling $\times$ Head Percolation Table:** There is a significant interaction between scoring with or without labels and the head percolation table used, $F(2, 157) = 5.00, p < .01$. As can be seen in Figure 17, ignoring labels benefits the dependency scores generated with Charniak's table (5.6% improvement) more than those generated using Collins' table (5.3% improvement) and those generated using Hwa's table (5.1%). Hence, the relative differences among the scores obtained using different head tables is not simply due to label robustness; the scores obtained with the Charniak table improve more when relaxing label match during scoring.

- **Parser $\times$ Head Percolation Table:** There is a significant interaction between parser and the head percolation table used, $F(4, 157) = 2.47, p < .05$. As can be seen in 18, all of the parsers obtained the largest dependency F-measure scores using Charniak's head table, with Collins' table giving second largest scores, and Hwa's the lowest scores; however, the relative decrease in scores from the

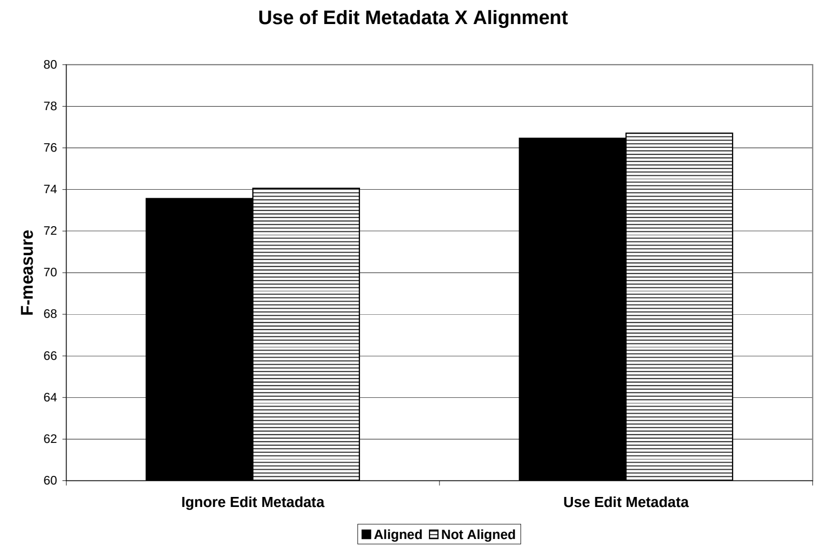

Figure 15: Average F-measure scores given the use of edit metadata and alignment.

use of table differs across parsers. Roark's parser using Collins' table gives a score that is closer to Charniak's table than the other two parsers, and so the greatest relative decrease in F-measure comes from its use of Hwa's table.

- **Labeling × Transcript Quality:** There is a significant interaction between the choice of whether or not to score labels and transcript quality, $F(1, 157) = 30.71, p < .0001$, which is depicted in Figure 19. It should be noted that the results from scoring reference transcripts without labels (83.47) gives a significantly higher score than using reference transcripts and scoring labels (79.24), $p < .0001$, which gives a significantly higher score than using ASR transcripts and not scoring labels (70.85), $p < .0001$, which gives a significantly higher score than using ASR transcripts and scoring labels (67.24), $p < .0001$.

- **Labeling × Metadata Quality:** There is a significant interaction between the choice of whether or not to score labels and metadata quality, $F(1, 157) = 5.35, p < .05$, which is depicted in Figure 20. Results from scoring with reference metadata and without labels (80.07) gives a significantly higher score than using reference metadata and scoring labels (76.02), $p < .0001$, which gives a significantly higher score than using ASR transcripts and not scoring labels (74.24), $p < .0001$, which gives a significantly higher score than using ASR transcripts and scoring labels (70.46), $p < .0001$.

- **Labeling × Parser:** There is a significant interaction between whether labels are scored or not for dependency scoring and parser, $F(2, 157) = 4.41, p < .05$, which is depicted in Figure 21. This effect is the result of the fact that the F-measure scores for each parser increase at different rates from relaxing label match when scoring the dependencies, with Roark's parser gaining relatively more (5.7%) than Charniak's (5.4%) and Bikel's (5.0%).

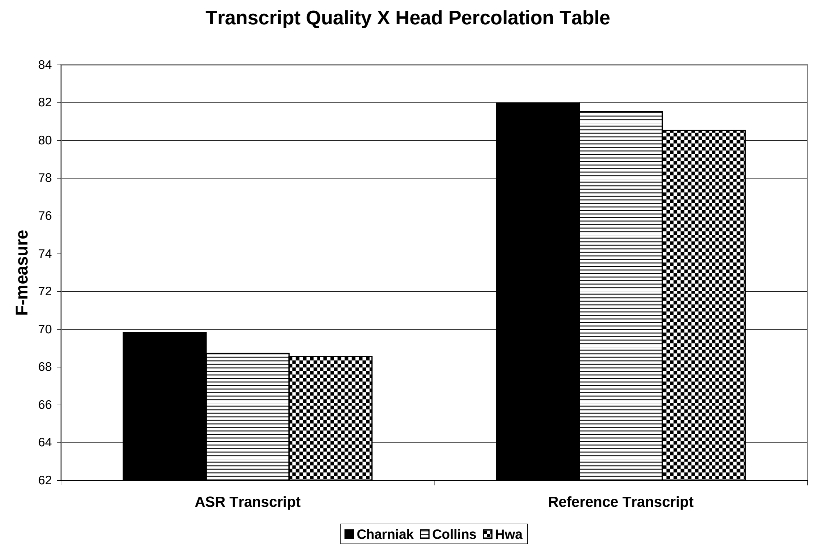

Figure 16: Average F-measure scores given transcript quality and head percolation table used.

- **Labeling $\times$ Parse Match Representation:** There is a significant interaction between whether labels are scored or not for dependency scoring and parse match representation, $F(1, 157) = 4.77, p < .05$, which is depicted in Figure 22. It is interesting to note that, when labels are ignored, the head dependency scores are not statistically significantly different than open class dependency scores; however, they do differ significantly when labels are used, $p < .0001$.

- **Metadata Quality $\times$ Parse Match Representation:** There is a significant interaction between metadata quality and parse match representation, $F(1, 157) = 5.69, p < .05$, as shown in Figure 23. Head dependency scores are not statistically significantly different than open class dependency scores when using system metadata; whereas, they are significantly different when using reference metadata, $p < .0001$.

- **Transcript Quality $\times$ Labeling $\times$ Parser:** There is a significant interaction among these three factors, $F(2, 157) = 3.51, p < 0.05$, as shown in Figure 24. All differences among parser scores are significantly different except for Bikel's and Charniak's parser on reference transcripts when scoring labels. In general, each of the parsers does better when parsing reference rather than ASR transcripts and when labels are ignored for scoring purposes rather than being scored. Charniak's parser is slightly more robust than Bikel's when operating on ASR transcripts, and it also seems to do slightly better on the reference transcripts when labels are not scored.

The impact of the data quality and use factors on dependency scores alone were similar to their impact on bracket and dependency scores. Reference transcripts and reference metadata allow a parser to produce higher parse scores than their system-produced counterparts. Also, use of edit metadata to remove edits prior to parsing results in larger parse scores. The interaction between transcript quality and metadata quality suggests that reducing transcript quality has more of a negative impact on parse scores than reducing metadata quality, and using edit metadata to remove edit regions prior to parsing has a more

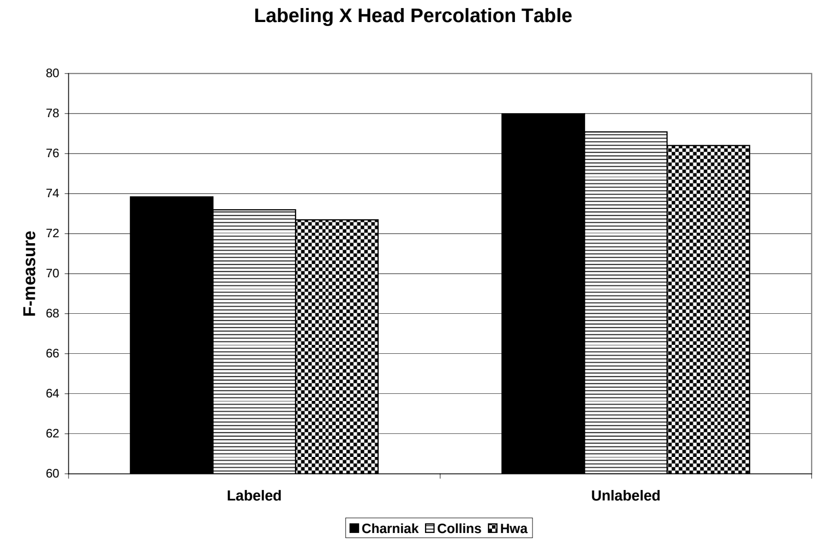

Figure 17: Average F-measure scores given labeling and head percolation table used.

positive effect on parse scores when using reference metadata than using system metadata. A similar finding was observed in the interaction between the use of edit metadata and transcript quality. The interactions between data quality factors and parser gives a similar picture about the impact of metadata and its use on Bikel's parser. There was also a significant interaction that emerged for dependency scores: the interaction between transcript and metadata quality with the use of edit metadata to remove edits prior to parsing.

The additional metric-related factors in the case of the dependency score analysis are alignment and head percolation table. Alignment adds an extra match constraint and so reduces the dependency scores slightly compared with scores calculated without this constraint. The relative improvement from relaxing the alignment constraint is greater when using ASR transcripts and when not removing edits prior to parsing; however, alignment does not appear to play a major role for dependency metrics, even though it is required in order to calculate the bracket scores. The headword percolation rules affect the magnitude of the dependency scores produced, with the Charniak table producing higher scores in general across all three parsers, with greater robustness to ASR transcripts. Dependency parses produced with Charniak's table also produces relatively larger unlabeled scores than the other two tables. The focus on dependency scores made evident a number of interactions between labeling and several other factors, in particular with transcript quality, with metadata quality, and with parser. The interactions between labeling and parse match representation, as well as between metadata quality and parse match representation, are a subset of the interactions observed in the analysis involving brackets.

It is interesting to note that metadata errors have a greater 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. These 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.

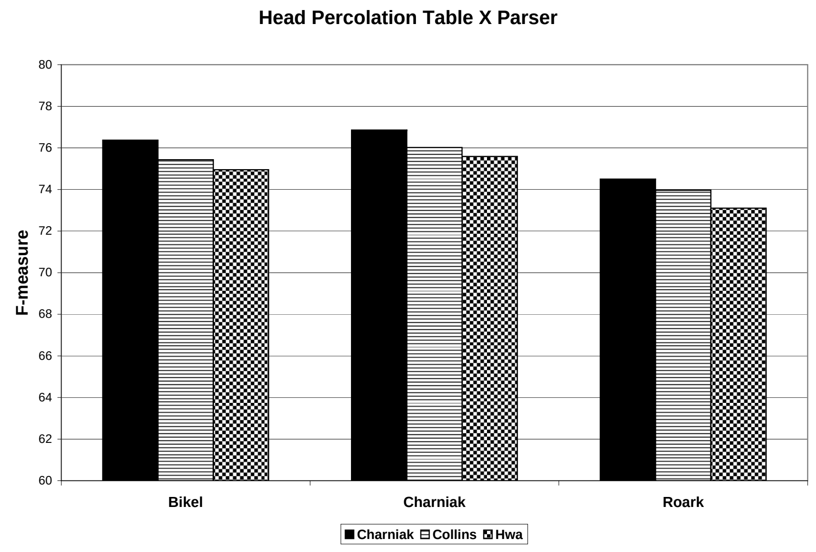

Figure 18: Average F-measure scores given parser and head percolation table used.

In the next section, we will focus on the impact of more subtle gradations in the data quality factors in order to better understand the impact of improvements in ASR transcription and metadata annotation accuracy. We also investigate the individual impact of system SUs or system edits on parse accuracy.

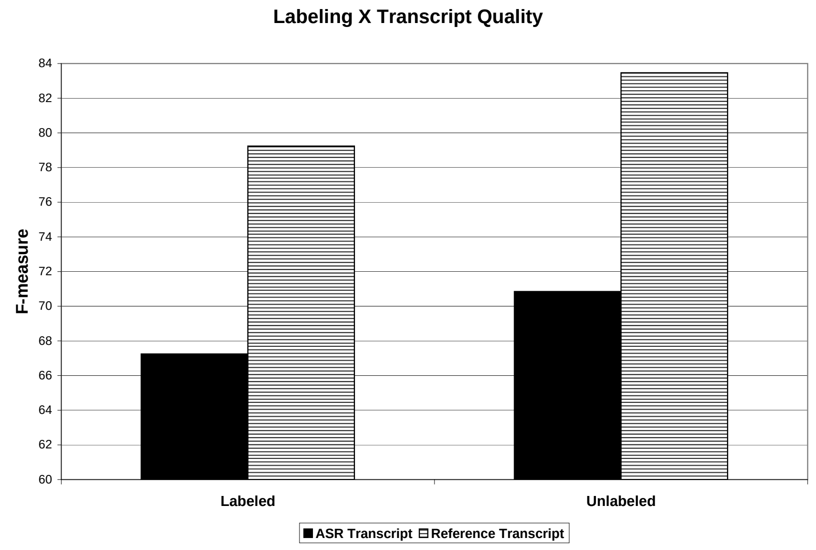

Figure 19: Average F-measure scores given labeling and transcript quality.

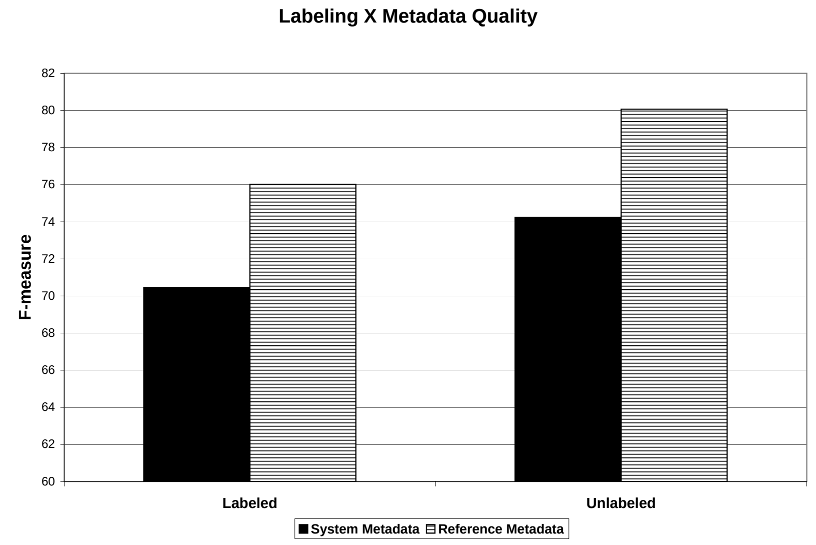

Figure 20: Average F-measure scores given labeling and metadata quality.

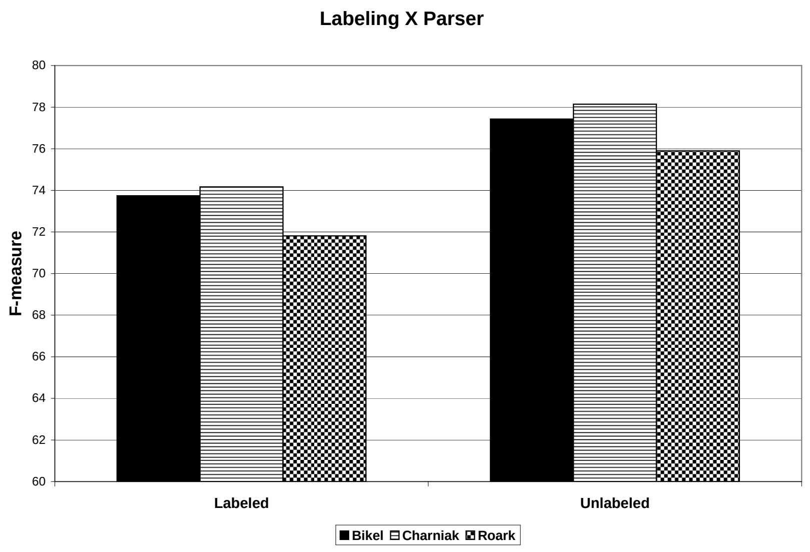

Figure 21: Average F-measure scores given labeling and the parser used.

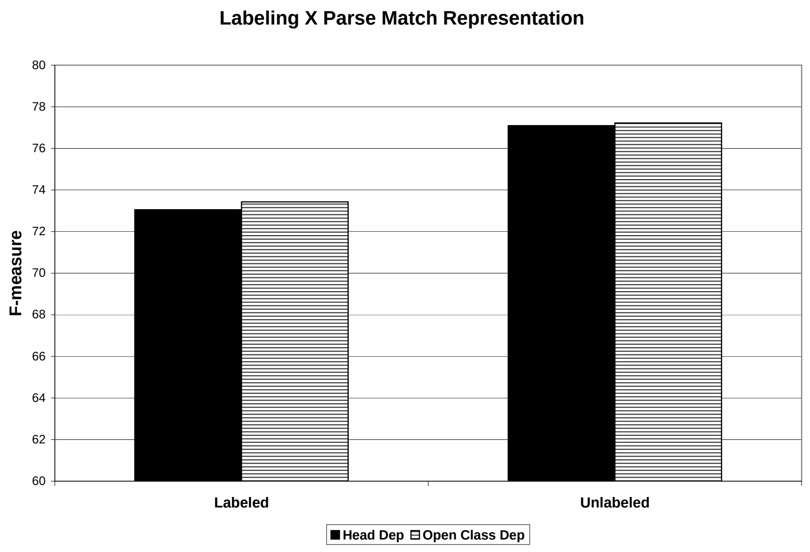

Figure 22: Average F-measure scores given labeling and parse match representation used.

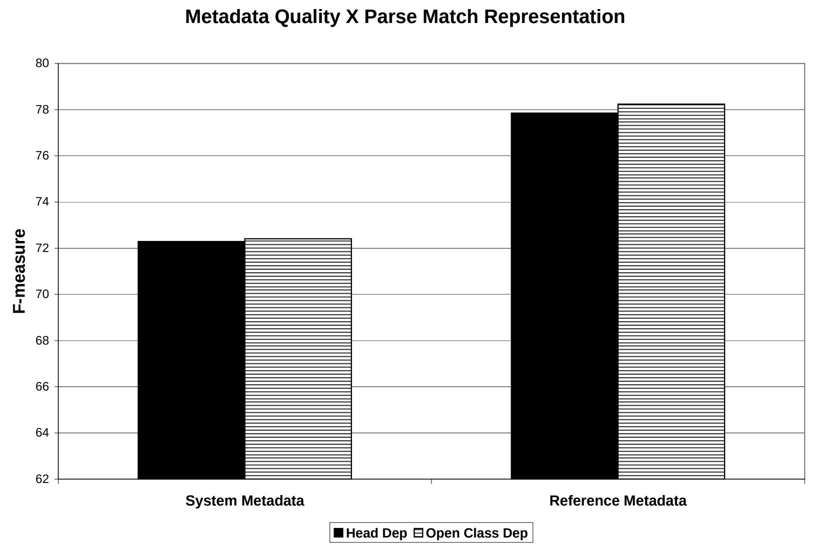

Figure 23: Average F-measure scores given metadata quality and parse match representation used.

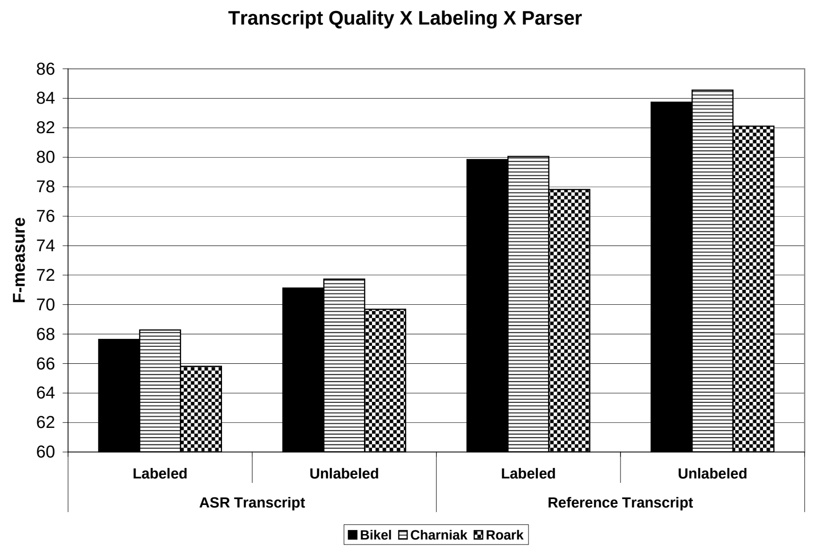

Figure 24: Average F-measure scores given transcript quality, labeling, and parser.

3.2.4 Focus Studies

Transcript Quality: In the previous investigations, we compared parsing of ASR system generated transcripts with parsing of reference transcripts. In this follow-up study, we investigate the effect of ASR transcript quality on parses produced by Charniak's parser using system metadata. Parsing is carried out on dev2 transcripts with three transcript qualities: reference, IBM+SRI [KMP$^+$04] (WER of 11.7% on dev2), and SRI [SBB$^+$00] transcripts (WER of 14.4% on dev2). A summary of parse scores over the 72 conversation sides appears in Table 45 in Section A of the appendix.

We have performed two analyses of this data: the first is a factorial analysis of variance involving the scoring of aligned parses only and ignoring head percolation rules in order to examine the significance of transcription quality on dependency versus bracket F-measure scores; the second is a factorial analysis of variance over dependencies only in order to investigate the effects of transcription quality together with alignment and head percolation table on dependency F-measure scores.

F-measure for the 72 conversation sides of the dev2 set was entered into a 3(Transcript Quality) × 2(Use of Edit Metadata) × 3(Parse Match Representation) × 2(Labeling) analysis of variance using alignment. Note that there is no metadata factor since we used only system metadata. There is a significant main effect of transcript quality, $F(2, 31) = 1,835.37, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses on human transcripts had a significantly greater F-measure (78.45) than those based on the IBM+SRI ASR transcripts (67.00), $p < .0001$, which had a significantly greater F-measure than those based on the SRI ASR transcripts, (64.57), $p < .0001$. Clearly, larger WERs produced by ASR systems cause a significant degradation in parse quality. There were several other significant main effects involving the use of edit metadata to remove edited material prior to parsing, $F(1, 31) = 23.56, p < .0001$, scoring with or without label match, $F(1, 31) = 317.65, p < .0001$, and the type of parse match representation $F(2, 31) = 66.69, p < .0001$. Other than transcript quality, the other main effects are similar to, but more focused than (i.e., only system metadata is used) those reported in Section 3.2.3, and so we do not discuss them further.

There is also a significant interaction between transcript quality and parse match representation for dependencies, $F(2, 31) = 3.66, p < .01$, which appears in Figure 25. In all transcript conditions, the bracket scores are significantly lower than each of the corresponding dependency scores, $p < .0001$; however, the head dependency and open class scores do not differ significantly from each other. This is interesting since these studies use system metadata, which tends to lower bracket scores more than the dependency scores relative to using reference metadata, as we previously have shown. Although the word errors in an ASR transcript tend to lower dependency scores to a greater extent than bracket scores, in this focus study, the bracket scores are always lower than the dependency scores, although the differences are reduced in going from reference transcripts to the IBM+SRI ASR transcripts due to a greater parse accuracy loss for the head (14.94%) and open class (14.82%) dependencies than for brackets (12.81%). Similarly in going from reference transcripts to SRI ASR transcripts, there is a greater parse accuracy loss for the head (17.9%) and open class (17.8%) dependencies than for brackets (16.5%).

F-measure for the 72 conversation sides of the dev2 set was also entered into a 3(Transcript Quality) × 2(Use of Edit Metadata) × 2(Alignment) × 2(Labeling) × 3(Head Percolation Table) × 2(Parse Match Representation) analysis of variance using alignment. There is a significant main effect of transcript quality, $F(2, 78) = 4,336.32, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses on human transcripts had a significantly greater F-measure (79.17) than those based on the IBM+SRI ASR transcripts (67.57), $p < .0001$, which had a significantly greater F-measure than those based on the SRI ASR transcripts, (65.20), $p < .0001$. There were also significant main effects of using or not using the edit structural metadata to remove edited material prior to parsing, $F(1, 78) = 40.36, p < .0001$, scoring with or without label match, $F(1, 78) = 767.21, p < .0001$, head percolation table $F(2, 78) = 33.06, p < .0001$, and alignment, $F(1, 78) = 13.14, p < .0005$. Other than transcript quality, the other main effects are similar

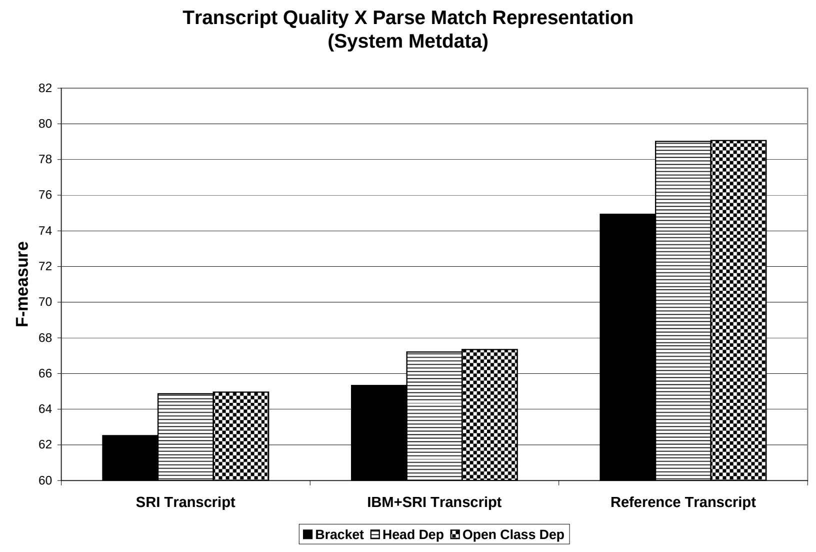

Figure 25: Average F-measure scores given transcript quality and parse match representation.

to, but more focused than (i.e., only system metadata is used) those reported in Section 3.2.3, and so we do not discuss them further.

There is a significant interaction between transcript quality and use of edit metadata for dependency scores, $F(2,78) = 3.66, p < .05$, depicted in Figure 26. The gain from using edit metadata diminishes as we move from parsing reference transcripts (1.68%) to IBM+SRI transcripts (1.02%) to SRI transcripts (.75%). As transcript quality drops, the benefit from removing edit regions prior to parsing is greatly reduced. There is also a significant interaction between transcript quality and labeling, $F(2,78) = 9.78, p < .0001$, depicted in Figure 27. Reference transcripts show the largest relative gain from ignoring labels rather than using them during scoring (5.78%). This gain diminishes as we move to the IBM+SRI transcripts (5.04%) and then to the SRI transcripts (4.92%).

Both of these analyses indicate that even small improvements to ASR transcription quality result in significant improvements in parse accuracy regardless of the scoring metric used, although word errors do have a greater impact on dependency scores than bracket scores. Although word error directly impacts parse accuracy, it also has a secondary impact on the quality of the system metadata, and so would have an indirect impact on parse accuracy, most particularly on bracket scores. In the next focus study, we hold transcript quality constant and vary metadata quality.

Metadata Quality: In this follow-up study, we investigate the effect of metadata quality using Charniak's parser to parse the dev2 reference transcripts with three metadata qualities: reference, prosody model only (SU error of 64.75% on dev2), and the baseline MDE model (SU error rate of 26.14%). For this study, we always use the edit metadata to remove the edited region from the word stream prior to parsing and then reinsert the material prior to scoring. A summary of parse scores over the 72 conversation sides appears in Table 46 in Section A of the appendix.

We have performed two analyses of this data: the first is a factorial analysis of variance involving the

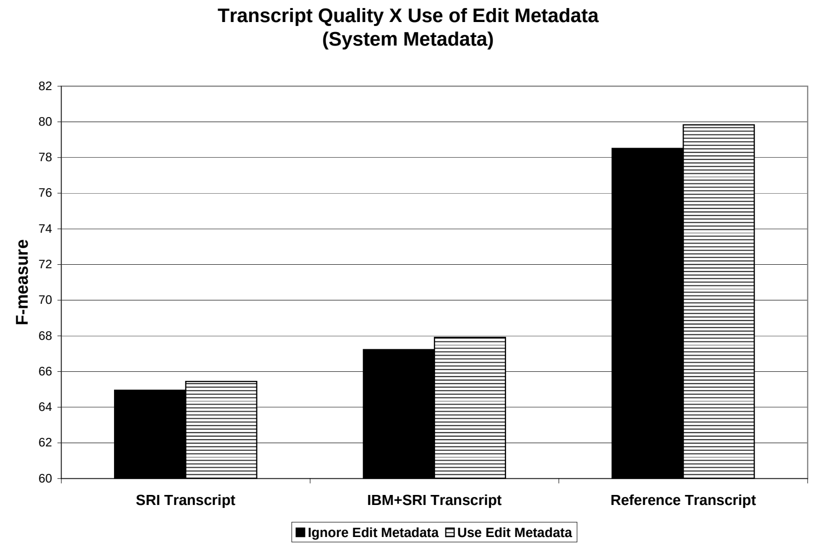

Figure 26: Average F-measure scores given transcript quality and use of edit metadata given system metadata.

scoring of aligned parses only and ignoring head percolation rules in order to examine the significance of metadata quality on dependency versus bracket F-measure scores; the second is a factorial analysis of variance over dependencies only in order to investigate the effects of metadata quality together with alignment and head percolation table on dependency F-measure scores.

F-measure for the 72 conversation sides of the dev2 set was entered into a 3(MDE Quality) $\times$ 3(Parse Match Representation) $\times$ 2(Labeling). There is a significant main effect of metadata quality, $F(2, 17) = 2,187.62, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses on human metadata annotations had a significantly greater F-measure (87.96) than those based on our baseline metadata system (79.14), $p < .0001$, which had a significantly greater F-measure than those based on a simple prosody model (69.69), $p < .0001$. Clearly, increased metadata error produces a significant degradation in parse quality even when parsing involves reference transcripts. There are significant main effects of scoring with or without label match, $F(1, 17) = 235.90, p < .0001$ and the type of parse match representation used, $F(2, 17) = 59.51, p < .001$. Other than metadata quality, the other main effects are similar to, but more focused than (i.e., only reference transcripts are used) those reported in Section 3.2.3, and so we do not discuss them further.

There is also a significant 2-way interaction between metadata quality and parse match representation, $F(4, 17) = 36.43, p < .0001$, which is depicted in Figure 28. It is important to note that as the quality of the metadata drops, all parse scores regardless of match type drop, although the degradation is far more serious for the bracket scores. The relative loss in parse scores when moving from reference to the baseline system metadata is much greater for bracket scores (15.21%) than for head dependency (8.91%) and open class dependency (9.38%) scores. The loss is even greater when using the prosody model metadata system; bracket scores show a greater loss (28.91%) than head dependency (18.51%) and open class dependency (20.28%) scores. Clearly use of the prosody model metadata results in serious parse accuracy loss relative to even the baseline metadata system. It should be noted that the open class dependencies have larger

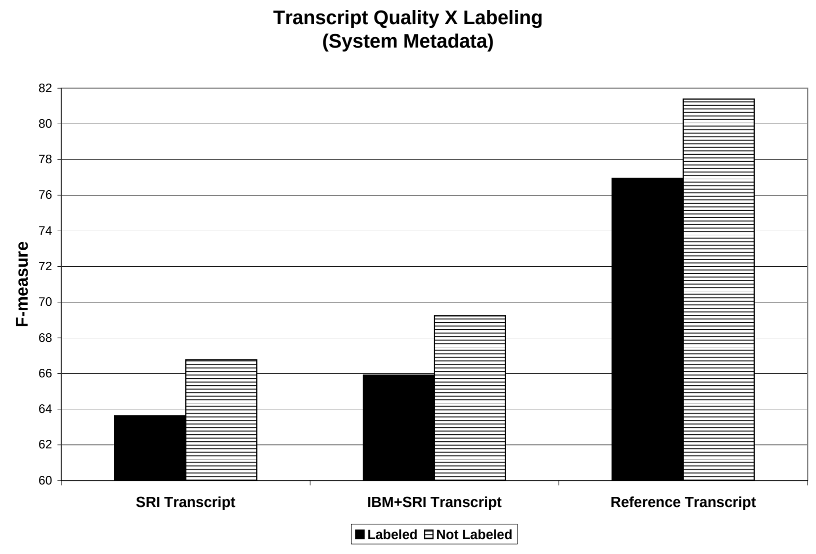

Figure 27: Average F-measure scores given transcript quality and labeling with system metadata.

scores than head dependencies when using reference metadata, but relatively lower scores as the metadata accuracy drops. There is also a significant interaction between labeling and parse match representation, $F(2, 17) = 5.89, p < .005$, which due to our focus here on reference transcripts is somewhat less informative than the results reported in the larger analysis reported in Section 3.2.3, and so we do not discuss it here.

F-measure for the 72 conversation sides of the dev2 set was entered into a 3(MDE Quality) $\times$ 2(Alignment) $\times$ 2(Parse Match Representation) $\times$ 3(Head Percolation Table) $\times$ 2(Labeling). There is a significant main effect of metadata quality, $F(2, 51) = 3,537.66, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses on human metadata annotations had a significantly greater F-measure (87.78) than those based on our baseline metadata system (79.83), $p < .0001$, which had a significantly greater F-measure than those based on a simple prosody model (70.84), $p < .0001$. There were also significant main effects of labeling, $F(1, 51) = 740.02, p < .0001$, and head percolation table, $F(2, 51) = 22.26, p < .0001$. Other than metadata quality, the other main effects are similar to, but more focused than (i.e., only reference transcripts are used) those reported in Section 3.2.3, and so we do not discuss them further. There is also a significant interaction between metadata quality and match type representation, $F(2, 51) = 8.66, p < .0002$, which looks similar to interaction depicted in Figure 28 with the bracket scores omitted, and so we do not include the figure here.

Both of these analyses indicate that improving metadata accuracy results in significant improvements to parse accuracy. These results also show that bracket scores are more sensitive to metadata errors. The greater sensitivity of bracket scores to SU choice can also be observed in Figure 29. The baseline metadata system places an SU at an interword boundary if the posterior probability is greater than or equal to a threshold of .5. The figure presents parse accuracy using bracket and head dependency scores (using the Charniak head percolation table) across a range of SU thresholds. This figure highlights quite clearly that the impact of varying the threshold on bracket scores differs substantially from the impact on dependency scores. This study is interesting also in that it highlights the fact that minimizing SU error does not

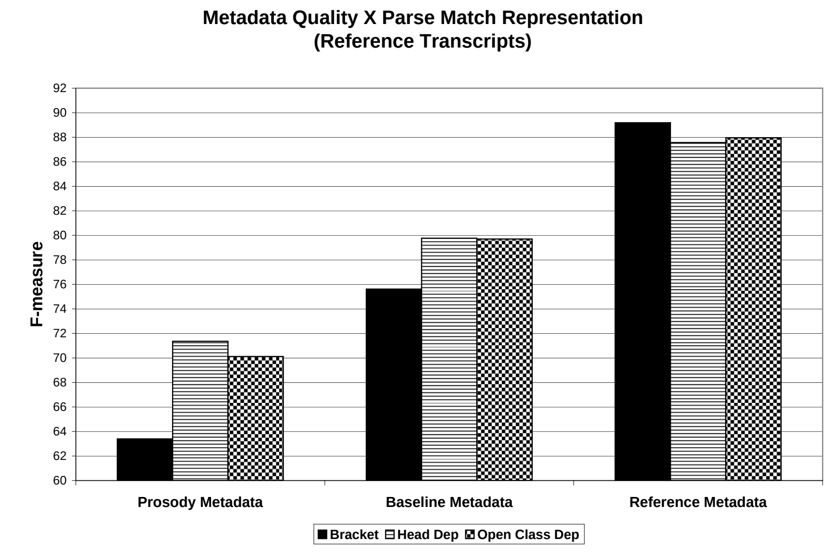

Figure 28: Average F-measure scores given metadata quality and parse match representation used.

always lead to the highest parse accuracies, in particular, shorter sentences tend to produce larger parse scores. This will be investigated further in Section 6, where we perform reranking experiments in which we individually investigate the objective of increasing SU boundary detection accuracy and increasing parse accuracy.

Impact of SU and Edit Metadata Quality: In the third and final follow-up study, we investigate the effect of SU and edit metadata quality using Charniak's parser to parse the dev2 reference transcripts with four metadata conditions: reference SU and reference edit metadata (Reference All), reference SU and system edit metadata (Reference SU), system SU and reference edit metadata (Reference Edit), system SU and system edit metadata (System All). For this study, we always use the edit metadata to remove the edited region from the word stream prior to parsing and then reinsert the material prior to scoring. A summary of parse scores over the 72 conversation sides appears in Table 47 in Section A of the appendix.

We have performed two analyses of this data: the first is a factorial analysis of variance involving the scoring of aligned parses only and ignoring head percolation rules in order to examine the significance of metadata quality of SUs and edits on dependency versus bracket F-measure scores; the second is a factorial analysis of variance over dependencies only in order to investigate the effects of this metadata together with alignment and head percolation table on dependency F-measure scores.

F-measure for the 72 conversation sides of the dev2 set was entered into a 4(MDE Quality Conditions) $\times$ 3(Parse Match Representation) $\times$ 2(Labeling). There is a significant main effect of metadata condition, $F(3, 23) = 985.95, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses based on reference edit and reference SU metadata had significantly larger F-measure scores (87.96) than those based on reference SU and system edit metadata (84.06), $p < .0001$, which had significantly larger scores than those based on system SU and reference edit metadata (82.12), $p < .0001$, which had significantly larger scores than those involving system metadata entirely (79.14), $p < .0001$. This suggests that SU errors more

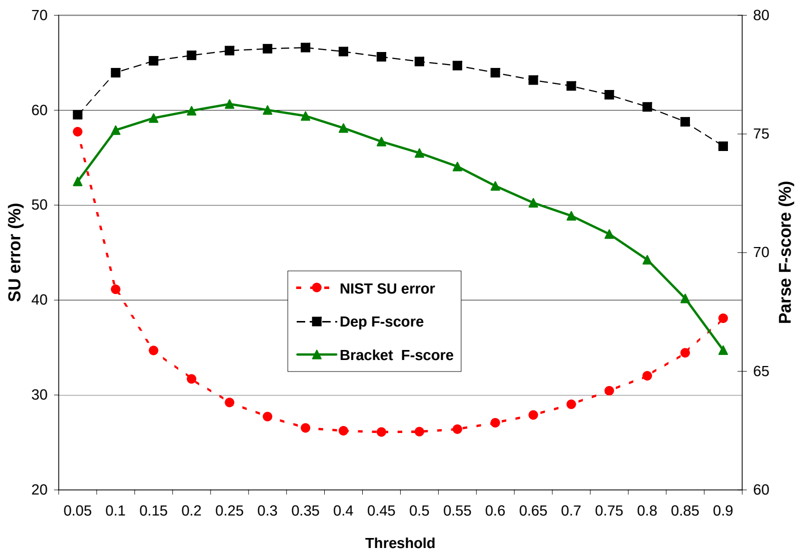

Figure 29: The impact of SU threshold on SU and parse accuracy.

negatively impact parsing accuracy than edit errors. There is a significant main effect of scoring with or without label match, $F(1, 23) = 831.37, p < .0001$, as well as parse match representation, $F(2, 23) = 36.90, p < .0001$. These main effects are similar to, but more focused than (i.e., only reference transcripts are used) those reported in Section 3.2.3, and so we do not discuss them further.

There is also a significant interaction between labeling and parse match representation, $F(2, 23) = 20.89, p < .0001$, as well as between metadata quality and parse match representation, $F(6, 23) = 38.08, p < .0001$. The former interaction is better understood with respect to the broader study in section 3.2.3. The later interaction is depicted in Figure 30. As can be observed, the impact of using system edit metadata instead of reference edit metadata while maintaining reference SUs is fairly small, with the percent accuracy loss for bracket scores (5.02%) being somewhat comparable to the impact on the head (4.24%) and open class dependency scores (4.42%). By contrast, the impact of using system SUs instead of reference SUs while maintaining the use of reference edits presents a very different pattern; the loss for bracket scores (11.54%) is much more dramatic than for the head (5.60%) and open class dependency scores (6.02%). The greatest loss to all scores comes from using all system generated metadata; the loss for bracket scores (15.21%) is much more dramatic than for the head (8.91%) and open class dependency scores (9.38%). When using reference edits and reference SUs, the bracket scores are significantly greater than the head dependency scores, $p < .001$, as well as the open class dependency scores, $p < .05$. When using reference SUs and system generated edit metadata, there is no significant difference among the scores, suggesting that the bracket scores are more negatively impacted than the dependency scores by the use of system edits. When using system SUs and reference edits, the bracket scores are significantly smaller than the dependency scores, $p < .0001$, but the dependency scores are not significantly different. The same pattern is observed when using system generated SUs and edits.

F-measure for the 72 conversation sides of the dev2 set was entered into a 4(MDE Quality Conditions) $\times$ 2(Alignment) $\times$ 2(Parse Match Representation) $\times$ 3(Head Percolation Table) $\times$ 2(Labeling) for dependency

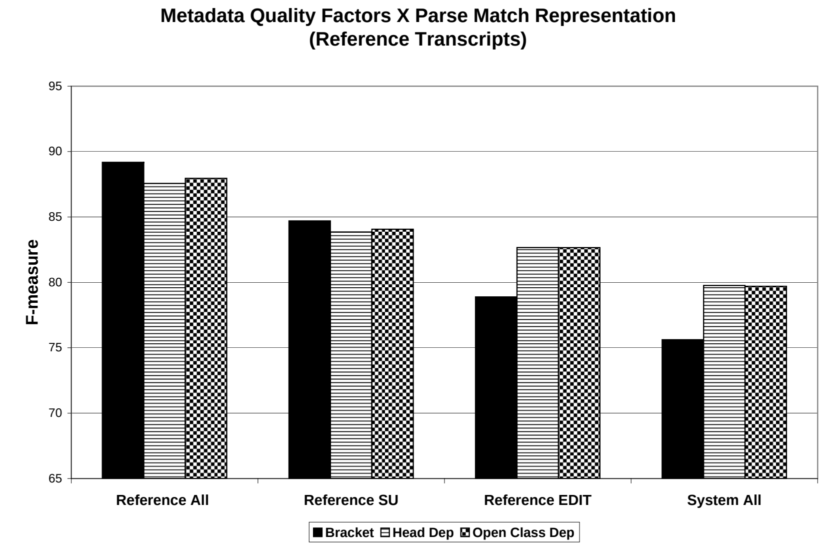

Figure 30: Average F-measure scores given SU and edit metadata quality and parse match representation used.

scores. There is a significant main effect of metadata condition, $F(3,66) = 1,430.69, p < .0001$. Using a post hoc Bonferroni t-test, we found that parses based on reference edit and SU metadata had significantly larger F-measure scores (87.78) than those based on reference SU and system edit metadata (84.06), $p < .0001$, which had significantly larger scores than those based on system SU and reference edit metadata (82.68), $p < .0001$, which had significantly larger scores than those involving system metadata entirely (79.83), $p < .0001$. This suggests that SU errors more negatively impact parsing accuracy than edit errors (even for dependency scores). There is a significant main effect of scoring with or without label match, $F(1,66) = 2739.19, p < .0001$, as well as head percolation table used, $F(2,66) = 79.17, p < .0001$. These main effects are similar to, but more focused than (i.e., only reference transcripts are used) those reported in Section 3.2.3, and so we do not discuss them further. There is also a significant interaction between labeling and head percolation table used, $F(2,66) = 3.09, p < .05$. Again since this study focused on reference transcripts, the results presented in Section 3.2.3 provide a deeper understanding of this interaction.

Overall this focus study demonstrates that higher quality SUs would have the most immediate impact on parse accuracy, so we have chosen to focus on improving SU accuracy in studies reported in Section 6.

3.3 Filler Recognition and Parsing

In addition to measuring the impact of reference versus recognized words, SU boundaries, and edit regions on parse accuracy, we were similarly interested in measuring the impact of reference versus recognized filler regions. Whereas previous work investigated methods for measuring the effect on parsing of incorrectly recognized SU boundaries [KOC04] and edit regions [CJ01], no such study has been performed to date with regard to recognized words and filler regions. In terms of fillers, the closest related work was a study which showed that parsing accuracy on conversational speech is degraded in the presence of interjections and parentheticals (i.e., INTJ and PRN constituents) [ECJ02]. However, the relevance of this study to our work was limited in the following ways:

- It made no attempt to automatically detect the phenomena.
- No method was proposed for how such detection could be integrated with a parsing model.
- There was no discussion of whether the standard Parseval metrics could be used in such a scenario or if some modification was needed, as we have seen to be true in all three cases of evaluating the impact of recognized words (see Section 3), SU boundaries [KOC04], and edit regions [CJ01].
- It remains unclear precisely what the relationship is between INTJs and PRNs (the phenomena it studied) and the sort of fillers annotated under SimpleMDE.

In regard to the last point, it is perhaps more to the point to say it remains unclear exactly how SimpleMDE filler annotations integrate with tree syntax in general.

Due to these challenges, as well as the fact that work on this topic began late relative to others pursued in the workshop, the study of the interaction between filler recognition and parsing is more preliminary than the experiments measuring the impact of word, SU boundary, and edit region recognition errors on parsing. Nevertheless, some in-roads were made, and the remainder of this section describes the work that was performed. This includes:

- suggestions for revisions to tree annotation guidelines to support a unified treatment of fillers across SimpleMDE and syntactic annotation,
- a simple experiment showing that a state-of-the-art parser often misanalyzes fillers,
- an experiment that treats filler annotations like POS tags and uses a state-of-the-art POS-tagger to detect them.

Future work involving how filler recognition can integrated with a parsing model and its impact evaluated appears in Section 7.

3.3.1 Unifying SimpleMDE and Treebank Filler Annotations

We describe here issues involved in achieving consistent annotation of fillers for data dually annotated under SimpleMDE and treebank guidelines [Str04, BFKM95, Tay96, BMW05]. In addition to enabling a study of how filler recognition and parsing interact, such a consistent definition would be valuable in terms of establishing both standard vocabulary for the community and consistent resources for training and evaluating speech processing systems. In terms of vocabulary, we follow RT'04F evaluation guidelines [Fal04] in using the term *filler* to encapsulate the SimpleMDE notions of filled pauses, discourse markers, and explicit editing terms; this differs from treebanking guidelines' use of *filler* in reference to filled pauses only [Tay96].

In seeking consistent annotation across standards, we begin by assuming SimpleMDE annotation is performed prior to treebanking (as was the case with LDC2005E15, the *JHU Speech Parsing Workshop English TB Data*), and that because SimpleMDE annotators listen to the audio, whereas the treebank annotators do not, SimpleMDE annotations should by default be presumed correct and preserved. This is consistent with the most recent treebanking guidelines: *try to follow the disfluency annotation as much as possible, even if this sacrifices a more intelligible reading* [BMW05]. Because work on LDC2005E15 was focused on SUs and edits, rather than on fillers, there remains a need to determine how SimpleMDE-annotated fillers should be treated during treebanking. A preliminary inspection of the filler treebank annotations produced suggests that inconsistencies between the two annotations are not infrequent, with the consequence that the conflicting annotations suggest alternative readings of various utterances.

SimpleMDE v6.2 [Str04] identifies four[7] types of fillers (emphasized text and examples below are excised from the annotation guidelines):

1. **Filled Pauses (FPs)**: *hesitation sounds that speakers employ to indicate uncertainty or to maintain control of a conversation while thinking of what to say next.* Examples include `ah, eh, er, uh, um`, etc.

2. **Discourse Markers (DMs)**: *words or phrases that function primarily as structuring units of spoken language.* Examples include `you know, i mean, like, so, well`, etc.

3. **Explicit Editing Terms (EETs)**: *overt statements from the speaker recognizing the existence of an edit disfluency.* Examples include `I mean, or rather`, etc.

4. **Asides and parentheticals (A/Ps)**: *asides occur when the speaker utters a short side comment on a new topic and then returns to the main topic being discussed. Parentheticals are similar to asides in that they are brief remarks that break the flow of the larger utterance, but unlike asides the remark is on the same topic as the larger utterance.* Asides and parentheticals are not distinguished from one another in annotation. Examples include `oh I don't know, how long has it been, his words`, etc.

Annotation guidelines specify that words which can occur as fillers can generally also occur as non-fillers; hence, annotators must be conscious of surrounding context in determining whether or not a given word or phrase is acting as a filler. A particular note is that words acting as backchannels, *words or phrases that provide feedback to the dominant speaker by indicating that the non-dominant speaker is still engaged in the conversation*, are not marked as fillers. Guidelines specify that in cases of uncertainty the word or phrase should not be annotated as a filler. Finally, in the event of multiple, contiguous fillers, annotators must decide if multiple words comprise a phrase filler or if each word is acting as an individual filler. Guidelines stipulate that *when in doubt, annotators should err on the side of separating the fillers into separate units.*

In terms of tree annotation, fillers were first specifically addressed in Section 2.2 of the Switchboard addendum [Tay96] to the original bracketing guidelines [BFKM95]. Emphasized text and examples below are excised from the addendum.

1. **Fillers**: *fillers are bracketed* INTJ. Examples include `uh, um`, etc.

2. **Discourse Markers**: *most discourse markers, like fillers, are labeled* INTJ. Examples include `well, like, now, see, say, actually, anyway`. *The discourse marker* `you know` *is labeled* PRN.

3. **Editing terms**: *Editing terms are labeled* PRN. Examples include `I mean, excuse me`, etc.

[7]SimpleMDE also identifies Discourse Response (DR) as a subtype of Discourse Marker but states the DR concept is experimental and subject to revision. No DRs appear in the RT'04F corpora.

4. **Asides**: *Asides which interrupt the flow of a sentence are fully bracketed inside* PRN.

5. *Continuers and assessors of all types are labeled* INTJ *and constitute a sentence on their own.* Examples include `uh-huh, huh, really, exactly, right, yeah, oh, okay`, etc.

6. *Combinations of* INTJ*s like* `oh really, oh yeah`, *etc. are each labeled individually as* INTJ *and the highest label is also* INTJ.

In addition to the Switchboard addendum, an additional addendum [BMW05] was created during treebanking of LDC2005E15 (the workshop corpus). Relevant details to mention include:

1. *all daughters of an INTJ node should have the UH POS tag*

2. *long renaming is enclosed in PRN*

For completeness, we note that the original treebank guidelines [BFKM95] and POS guidelines [San90] make no explicit mention of fillers. The treebank guidelines state of the (INTJ) tag only that it *corresponds approximately to the part-of-speech (POS) tag* UH, and PRN is only loosely defined in terms of surrounding punctuation and annotator intuition, although the final example in Section 2 of the guidelines does show disfluent speech without any delimiting punctuation bracketed by PRN. The POS guidelines regarding UH are limited to one list of examples, without any definition of the term. We also note that the Switchboard corpus serves as a useful reference as to the resulting product of these rules. The top 97% of the INTJ word-phrase distribution and top 65% of the PRN distribution are conveniently listed by Engel et al. [ECJ02]. We assume phrasal INTJs shown in the listing there correspond to top-most INTJs, which, in turn, parent an additional INTJ for each of the terminals (per the last rule in the Switchboard addendum [Tay96] cited above).

After comparing SimpleMDE and treebank guidelines with regard to filler annotation, as well as preliminary inspection of the data to that end, we believe a consistent, unified treatment of filler annotation is possible. As stated earlier, our assumption is that SimpleMDE annotation will be performed prior to treebanking, thus the primary goal is identifying how SimpleMDE filler annotations can be unambiguously mapped to tree annotation. This leads to a preliminary proposal of a few additional rules for treebanking:

1. a word should descend from INTJ in the tree if it is marked as a complete filler phrase in the SimpleMDE annotation

2. a phrase (group of multiple, contiguous words) should descend from PRN in the tree if it is marked as a complete filler in the SimpleMDE annotation

3. a short list of common single-word fillers, such as `like, so, well, actually`, and `now`, etc., should descend from INTJ only if they are marked as complete fillers in the SimpleMDE annotation.

4. a short list of common phrasal fillers, such as `you know` and `i mean`, etc., should descend from PRN only if they are marked as complete fillers in the SimpleMDE annotation.

The first two rules give a simple unidirectional mapping from SimpleMDE to tree annotation. Such a simple mapping is not possible in the other direction due to INTJ and PRN categories conflating fillers with other phenomena (i.e., continuers, assessors, renaming, etc.). As such, the latter two rules provide an admittedly weak safety net for catching common filler words being treated as fillers in syntactic annotation when the SimpleMDE annotation does not support this. An initial version of these rules were submitted to LDC during the workshop for review, and since then we have continued to study the data, revise the rules accordingly, and stay in communication with LDC about the issue. The question remains open whether the rules described here are the best ones to adopt. To maximize consistency with existing treebank guidelines, we have preserved the ambiguity with regard to use of INTJ and PRN categories and their correspondence to SimpleMDE filler annotation guidelines.

A final note is merited in regard to the SimpleMDE annotation guideline that in cases of uncertainty, the word or phrase in question should not be annotated as a filler. As SimpleMDE annotations were specified in terms of transcript cleanup (i.e., marking fillers in transcripts to improve readability), this decision to not annotate a word or phrase as a filler in case of doubt may be viewed as deferring the decision to the eventual reader. Unfortunately, the situation is somewhat different when we consider SimpleMDE as an annotation step preceding treebanking. Unlike SimpleMDE annotation, treebank annotation must commit to a specific analysis for the given utterance, hence a SimpleMDE deferral in case of uncertainty amounts to a final decision when SimpleMDE filler annotations are mapped to syntax as described above. As such, we also suggest that at least in the usage scenario we describe here, it would be best to encourage SimpleMDE annotators to take their best guess in cases of uncertainty rather than always leaving such cases unannotated.

3.3.2 A Simple Filler Experiment

This section reports a simple experiment showing that a modern parser [Cha00] often misanalyzes filler constructions when parsing reference transcripts with reference SUs. In this experiment, a parse tree is input to a trivial rule-based filler recognition system which deterministically recognizes fillers as follows:

- Every occurrence of `like, so, well, actually`, and `now` with POS tag UH is labeled as a discourse marker[8]
- Every occurrence of `you know` and `i mean` found within a PRN constituent is labeled as a discourse marker
- Following a previous rule-based system [JCL04], every occurrence of `uh, um, eh`, and `ah` is labeled as a filled pause, and `oh` is labeled as a discourse marker if it is the first word in the sentence or the sentence is longer than four words.

Note that the last rule ignores the syntax of the input tree and merely serves to boost the absolute recognition accuracy of the system for all inputs. In contrast, the first two rules heavily depend on the syntactic accuracy of the input trees.

Given this rule-based system, we evaluated filler recognition accuracy on dev2 given input trees of varying syntactic accuracy. In all conditions we used human reference words and SU boundaries; only the syntactic accuracy of the input trees was varied. Given human annotated trees as input, the system achieved 11.6% NIST error, and this represents a lowerbound on achievable error rate. While this lowerbound may seem high, recall that there is no direct mapping from treebank annotations to SimpleMDE annotations, and that correlation which does exist is somewhat obscured by current annotation inconsistencies in the data. Relative to the state of the art, this lowerbound error is roughly half the error achieved on the eval test set by the best performing system in the RT'04F evaluation, although it should be noted the RT'04F system used recognized rather than reference SU boundaries [JCL04].

Table 6 shows the performance of our rule-based system given parser output arising from three different training conditions. The first row corresponds to the training condition described earlier in Section 3.2.2. The second row shows just what you would expect: training on more (in-domain) data helps. The last row is somewhat surprising in that the training configuration used simultaneously results in a 2% improvement in filler recognition accuracy while at the same time parsing accuracy degrades by 1%. This bears further

[8]This is roughly equivalent to checking whether the given word is part of an INTJ constituent due to the annotation guideline that *all daughters of an INTJ node should have the UH POS tag* [BFKM95].

Train	**Tune**	**Parser F**	**Filler Err**
Swbd-train	dev1	88.1	24.7
Swbd-train+dev1	eval	88.5	23.7
Swbd-train+dev1+eval+Swbd-dev-2K	Swbd-dev-rest	87.6	21.5

Table 6: Parse accuracy of trees under various parser training conditions, and the filler recognition error achieved from using these trees as input to the rule-based system. Given gold trees as input, filler recognition error falls to 11.6%

Type distinguished	**End word distinguished**	**Nist Err**
yes	yes	31.8
yes	no	30.6
no	yes	32.2
no	no	21.9

Table 7: MXPOST filler recognition on dev2 when trained on dev1.

investigation beyond what time was available in the workshop. The key point for this discussion is that the best filler recognition error achieved is roughly double that obtainable given gold trees as input. This shows that the parser is often mistaking filler words as their non-filler counterpart homophones. In short, its syntactic analysis of such words is often incorrect. As such, we see that our parser stands to benefit from integrating knowledge of fillers into its model, and this is a topic of proposed future work in Section 7.

3.3.3 Filler Detection with MXPOST

In this section, we investigate the similarity between filler detection and part-of-speech (POS) tagging. Treating filler detection as a sequential labeling task akin to POS-tagging, we apply a state-of-the-art POS-tagger, MXPOST [Rat96], training it on filler tags rather than POS tags. As experimental variables, we vary the tag set used for training: (1) whether the tags distinguish filler-ending-words from other filler words (e.g., in `you know`, whether `know` is tagged differently than `you`), and (2) whether the tags distinguish filler types (i.e., filled pause, discourse marker, explicit editing term) or merely distinguish fillers from non-fillers. Performance is evaluated solely in terms of distinguishing between filler and non-filler words, reflecting our ultimate interest in eventually applying filler detection in support of parsing. We envision parsing (at least initially) will benefit from recognition of filler words but not filler types (see future work described in Section 7). Experiments were carried out on dev2 using reference words and SU boundaries, and MXPOST was run with its default settings.

Results are presented in Tables 7 and 8. We see that for the task of distinguishing filler from non-filler words, the best result of 21.9% NIST error does not model filler end tags or filler types. While a direct comparison to the best RT'04F result (23.7%) [JCL04] is not appropriate here since we have initially used reference rather than detected SU boundaries, we do see that performance is in the same general ballpark[9].

[9]The filler detection task in RT'04F also required detecting the *type* of each filler, but since type substitution errors accounted for less than 0.2% error in all measured systems, this seems to be largely a non-issue.

Type distinguished	**End word distinguished**	**Nist Err**
yes	yes	23.7
yes	no	25.1
no	yes	23.5
no	no	24.1

Table 8: MXPOST filler recognition on dev2 when trained on train, dev1, and eval.

We were concerned to see that in this best performing case of training on the simplest tag set, training on Swbd-train and eval as well as dev1 lowered performance from 21.9% error to 24.1% error. We suspect this can be attributed to variable annotation quality of the data. In particular, whereas dev1, eval, and 10% of Swbd-train had their annotations adjudicated between annotators, annotations in dev2 and the remainder of Swbd-train represent second-pass, non-adjudicated annotation. This suggests a tradeoff between quantity and quality of data for training, and reliability of test data. In experiments conducted since the conclusion of the workshop (not reported in detail here), we have seen that training on dev1 and eval (both adjudicated datasets) still performs worse than training on dev1 alone when evaluating on dev2, likely due to dev2's not being adjudicated. In contrast, when evaluated instead on the 10% of Swbd-train that was adjudicated, we do see a small improvement from the larger training set. Significance testing and further probing is merited here.

4 Using Prosodic Structure

This section focuses on harnessing acoustic cues related to meaning, intent and emphasis to inform downstream applications such as parsing and detection of other structural events. Already, the baseline MDE system, described in Section 2, incorporates acoustic measurements involving duration, energy, and pitch to predict sentence boundaries. Going beyond sentence boundary, here we develop techniques to detect certain commonly perceived prosodic events in spoken utterances. Section 4.2 reports development of a classifier for detecting prosodic events[10], and three applications for utilizing their predictions. The first application tackles the problem of disfluency detection, wherein prosodic predictions are incorporated into a parsing framework via two simple generative models. This is described in Section 4.3. The second application, described in Section 4.4, examines the impact of prosodic predictions on the sentence boundary detection task through a series of experiments on ASR transcripts with different word error rates. The third application, in Section 4.5, describes a set of syntactic-prosodic features that capture potentially useful interactions between the two structures. These features are shown to be useful in the sentence boundary reranking experiments described in Section 6.

4.1 Motivation

Speech conveys more than a sequence of words to a listener. Apart from age, gender, dialect and emotional state of the speaker, it also conveys meaning, intent, and emphasis of the utterance. The acoustic cues that convey information about the content are often described by linguists and phoneticians in terms of prosody. The most apparent effect of prosody is in drawing a listener's attention to important information through contrasts. For example, certain contrastive pitch or duration patterns can highlight the associated word or phrase. In addition, prosodic cues also segment speech into groups of syllables, that are hypothesized to have a hierarchical structure, although not necessarily identical to that of syntax. For a detailed review of the role of prosody in processing spoken language, see [CDD97, OSB03].

4.2 Automatic Detection of Prosodic Events

A popular scheme for representing the prosodic structure of a spoken utterance is ToBI [SBP$^+$92]. Briefly, ToBI has three streams of information, namely, tones (To), break indices (BI), and prominence. Tones are marked at the word level, break indices at word endings, and prominence at syllable level. The tone tier represents an intonational pattern as a sequence of high (H) and low (L) tones marked with diacritics to indicate their function as a pitch accent or as a phrase tone. Break indices which indicate the degree

[10]We thank Dustin Hillard for providing software tools for prosodic feature extraction and for providing the automatic word- and phone-level time alignments of the subset of Switchboard corpus used in this work.

of juncture between the words, are marked as labels assigned after each word in the transcription. Break indices take on values from 0 to 4, where increasing numerical value corresponds to a higher degree of disjuncture. The break index tier is responsible for the prosodic grouping of utterances. The two highest break index values, namely, 3 and 4, correspond to intermediate and intonational phrases respectively. In addition, a suffix'p' is used to denote a disfluent break. Prominence is marked at the syllable level with a single indicator variable (*). Although originally ToBI was intended for English, it has been adopted for languages such as Greek and Korean. For illustrative examples in English, see [BE93]. For the purpose of this work, it is sufficient to view ToBI as a system that provides a symbolic representation of certain commonly perceived prosodic events in a spoken utterance.

4.2.1 A ToBI Corpus

During the workshop, an automatic system was developed for detecting ToBI-style prosodic events in English conversational speech. The system was trained on manually transcribed data, which consisted of a subset of the Switchboard corpus [GHM92]. The annotated corpus is described in [OSSH$^+$01] and contains about 5 hours of speech from 64 telephone conversations. The consistency of annotations has not been measured, since only one transcription was available (a trained linguist is needed to perform the annotation). For previous studies on inter-labeler reliability with ToBI, see [PBH94]. The annotations consisted of the following set of labels.

1. *Tones*: H-L%, H-H%, L-H%, L-L%.
2. *Break Indices*: 0-4, 1-, 2-, 3-, 4-, 1p, 2p, 3p
3. *Prominence*: *

4.2.2 Acoustic Features

Prosodic events are cued by variations in pitch, duration and energy. These variations are often associated with word, syllable or sub-syllable constituents. The set of features described in [SFK$^+$05, SSHW98] were adopted for this work. Similarly to the prosody model used in the baseline metadata system described in Section 2, the features are computed from a time-aligned phone and word-level output from an ASR system. The syllabic constituents of the input are identified using a syllabified lexicon, generated with Fisher's implementation (*tsylb2*) of Kahn's principles of English syllabification [Fis96, Kah76]. The duration of the constituents are derived from the ASR alignment, while energy and pitch are computed every 10ms with *snack*, a public-domain sound toolkit from Royal Institute of Technology in Stockholm (KTH) [Sj01]. The duration, energy, and pitch are post-processed according to stylization procedures outlined in [SSHW98] and normalized to account for variability across speakers. In addition, statistics such as min, max, mean, and variance of each feature are computed. Note that components in the input feature vector related to pitch cannot be computed for unvoiced regions of speech. All in all, 102 types of features are computed for each word in the transcript, which can be grouped into three categories as illustrated in Figure 31. Similar features were used in the baseline MDE system reported in Section 2.

To compute the acoustic feature vectors, phone and word-level time alignments are needed. They were generated automatically by an ASR system at University of Washington after applying standard text normalization procedures. The ToBI corpus was created based on a modified version of Switchboard transcripts that were generated at Mississippi State University using modified transcription rules and additional manual correction. The break index values were marked according to the word-level time marks on these Mississippi transcripts, and the prominence values were marked at the syllable level. Hence, the ToBI labels needed to be projected onto our transcription alignments. This was carried out through a

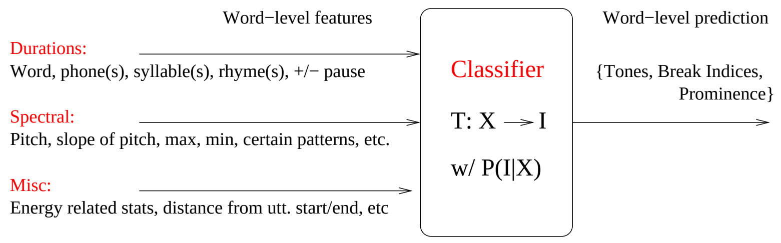
Word−level features
Durations:
Word, phone(s), syllable(s), rhyme(s), +/− pause
Spectral:
Pitch, slope of pitch, max, min, certain patterns, etc.
Misc:
Energy related stats, distance from utt. start/end, etc
Classifier
T: X → I
w/ P(I|X)
Word−level prediction
{Tones, Break Indices, Prominence}

Figure 31: Types of acoustic features used for classification of tones, breaks and prominence.

sequence of pre-processing steps that accounted for the differences in text normalization. The resulting strings were aligned using edit distance and ambiguities in alignment were resolved using constraints provided by ToBI labels (e.g., prominence can not be on a pause). This resulted in a corpus of 65,114 words annotated with ToBI labels.

4.2.3 Classifiers

With the acoustic features described above as the input, several classifiers were investigated for detecting the ToBI prosodic events. Since the input vector can have missing values such as the absence of pitch during unvoiced sound, only decision tree based classifiers were investigated. Decision trees allow several mechanisms to handle missing features gracefully. Essentially, a decision tree partitions the input feature space in a hierarchical manner to improve a purity or cost function. The design of a decision tree classifier is controlled by three components: (a) definition of feature dimensions which can be partitioned into subspaces, (b) a cost function to choose between several possible partitions, and (c) a criterion for stopping the successive partitioning. The feature dimensions can be categorical or continuous variables. Popular cost functions are GINI index of diversity, the Bayes splitting rule, information gain, gain ratio, and twoing. Examples of stopping criterion are minimum occupancy threshold and cross-validation gains. By choosing different combinations of splitting and stopping criterion an ensemble of decision trees were built which were then combined to create ensemble-based classifiers.

1. **Voting**: A voting-based classifier was tested using 33 different trees, generated by applying permutations of the following control parameters.

   - Splitting rule: GINI index of diversity, information gain, twoing, Quinlan's gain ratio.
   - Number of folds in cross-validation (values used: 5, 10, 15, 20).
   - Method of handling unknowns: the data with unknown feature component is sent down (a) a single branch with a probability proportional to that of the node, or (b) both branches in proportion to their corresponding probabilities.
   - Prepruning: extension of ungrown tree nodes are inhibited when the posterior probability of children is not within a specified range of the corresponding leaf (values used: 0, 0.01).
   - Minimum occupancy threshold: Splitting of a node is stopped if the total number of examples is less than this threshold, or if one class has more examples than half this size. Higher value speeds up search in noisy data (values used: 3, 5, 10).
   - Subsize: when data more than size 1.2 $\times$ subsize is at the splitting node, a sub-sample of size subsize is used to evaluate potential tests (values used: 900, 1000, 1100, 2000).

   Each of the 33 trees provides its own prediction. Each prediction counts as a vote of equal weight, and all votes are tallied for each word. The class with the most votes is used as the final predicted label.

2. **Bagging**: In bagging, an ensemble of classifiers is built, where each classifier is learned on a randomly chosen set of training data [Bre96]. The subsets are chosen for each class independently with replacement such that the proportion of classes is the same as in the full data set. The same 33 sets of tree-building options, described in the voting classifier, were used to generate the ensembles. However, unlike voting, the results are computed by averaging the posteriors across all trees. The class with highest average posterior probability is chosen as the final result. The ensembles for voting and bagging were generated using publicly available IND package from NASA [Bun92].

3. **Random Forest**: A random forest is a recently developed ensemble classifier, which has been shown to provide the best accuracy on certain well-studied classification and regression problems [Bre01]. Random forests add an additional layer of randomness to bagging. In addition to constructing each tree using a different sample of the data, each node is selected from a random subset of choices. This is unlike the standard trees, where a split is chosen after exhaustive evaluation of all possible splits. The results are chosen by votes among the trees. The generalization error for forests converges to a limit as the number of trees in the forest becomes large. The results compare favorably to that of Adaboost while being more robust. The random forests were tested using the publicly available implementation by Breiman [Bre01].

Temporal Constraints: In addition, two other techniques were investigated to improve performance using temporal constraints. The first is motivated by the practice in speech recognition of stacking feature vectors from the neighborhood to represent local temporal context. In a similar fashion, the acoustic features from the previous and next word were stacked on top of the feature vector to create a new feature vector. This was done per utterance, and components at the edge corresponding to missing frames were represented as such (i.e., no ad hoc values were inserted at the edges). The second technique models the prosodic events in terms of hidden Markov Models. Since there are at most only three hidden states at any given time, the hidden Markov model can be implemented efficiently using a finite state machine in a manner that allows one to extend the experiment to test second and third order Markov constraints. The observation densities for the hidden Markov models can be derived from any of the classifiers listed above. However, they provide posterior probabilities while HMMs require likelihoods for observations, and so they need to be modified as illustrated below.

Consider an utterance with four words and assume that there are three possible break index values at each juncture (which the experiments to be reported in Section 4.2.4 show provide good accuracy), namely, $s_i = 1, p$ and 4, with the corresponding likelihoods $P(F_i|1)$, $P(F_i|p)$ and $P(F_i|4)$, where F_i denotes the input feature computed for W_i. These observation likelihoods can be written in terms of their posterior probabilities using Bayes rule, as in $P(F_i|s_i) = P(s_i|F_i)P(F_i)/P(s_i)$. The Viterbi path of the HMM is computed by maximizing the joint likelihood of the hidden states and the observations.

$$\hat{s}_{1:n} = \arg\max_{s_{1:n}} \prod_i P(F_i|s_i)P(s_i|s_{i-1})$$

This can be re-written in terms of posterior probabilities of the observations.

$$\hat{s}_{1:n} = \arg\max_{s_{1:n}} \prod_i \frac{P(s_i|F_i)P(F_i)}{P(s_i)} P(s_i|s_{i-1})$$

Since $\prod_i P(F_i)$ contributes a constant factor to this expression and is not influenced by the choice of the hidden state, s_i, it can be removed. Thus, the Viterbi path of the hidden Markov model can be written in terms of the posterior probabilities of the observations. The above equation has two parts: a grammar $P(s_i|s_{i-1})$ and a sausage related to the contribution from the observations, which can be represented as in Figure 32.

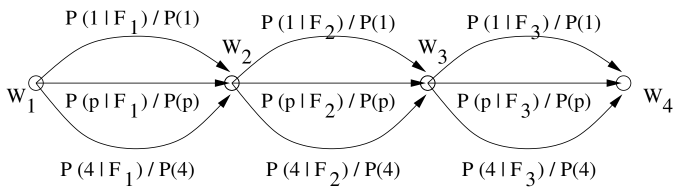

Figure 32: The contribution of the observations to the Viterbi path of the HMM.

Note, the factorization presented above holds for any order of the Markov chain. Thus, it provides an easy mechanism for computing the Viterbi path of a hidden Markov models of different orders by simply applying grammar of the appropriate order. The Viterbi state sequence is simply the shortest path after composing the observation FSM with the grammar, i.e., "fsmcompose obs.fsm g.fsm — fsmbestpath", using AT&T tools [MPR98].

4.2.4 Results on Classification of ToBI-style Prosodic Events

Before describing the results, it is useful to understand the skew in the class distribution, particularly, for break index values. The entire annotated corpus contained about 14 different classes of break index values with counts as shown in Table 9.

Class	Count	Class	Count
-	274	3	1143
0	3280	4-	1731
1-	2711	4	9010
1	35111	1p	1722
2-	282	2p	4152
2	966	3p	171
3-	1072	X	3489

Table 9: Frequency of different break index values in the corpus.

Preliminary classification experiments using a single decision tree showed that some of the classes could not be distinctly classified robustly, which may be attributed to the limited amount of training data or difficulty in making fine distinctions in a conversational speech corpora. A 14-way classification of categories listed in Table 9 gave an accuracy of 67% (chance=54%). After collapsing the 'N-' to 'N' and discarding the unknown (X and '-'), the accuracy was 72% (chance=62%). When the three fine categories of disfluent labels (1p, 2p, 3p) were collapsed into one class (p), the performance reached 73% (chance=62%). Since there were fewer instances of class '2' and '3' in the corpora, they were collapsed into '1'. Later on, due to preponderance of data for '1', the contribution from other classes with too few examples were discarded (e.g., 2, 2-). Thus, the final set of break values consisted of '1', '4' and 'p', and the results from a 10-fold cross-validation are shown in the Table 10. The results show that close juncture (value of 1) gets classified correctly most often, while loose juncture and disfluent breaks tend to be correct about 70% and 61% of the time they are predicted, respectively.

Classes	1	4	p
1	33665	922	524
4	3492	6032	1217
p	1693	1673	2679

Table 10: Baseline classification accuracy for break index values is 81.65% (chance=67.66).

The distribution of boundary tones are skewed significantly. So, to get a sense of the classification accuracy, the classes were restricted to tones associated with intonational phrase boundary (break index value of 4). The resulting 10-fold cross validation results are shown in the Table 11. The classification of the two classes L-H% and H-L% is fairly good; however, the other two classes are clearly poorly predicted. Perhaps, this is related to the small amount of data for the class H-H%. To constrain the scope of the effort for the workshop, boundary tones were not studied further.

Classes	H-H%	H-L%	L-H%	L-L%
H-H%	151	132	182	81
H-L%	82	841	560	1154
L-H%	110	452	1502	881
L-L%	39	550	632	3057

Table 11: Baseline classification accuracy for intonational boundary tones is 53.34% (chance=41.11%).

The cross-validation results for detecting prominence are given in the Table 12. Prominence can be detected with an accuracy of 78.85%, making it practical to use this classifier in applications described later in this section.

Classes	Absent	Present
Absent	37664	5482
Present	8039	12754

Table 12: Baseline accuracy of detecting prominence on words is 78.85% (chance=67.48%).

Finally, the performance of the three ensemble-based classifiers described in Section 4.2.3, are compared with baseline results in Table 13 for prominence and break index values. The results show a clear trend across the columns, although pairwise comparisons between adjacent rows are not statistically significant. The voting-based classifier gives better accuracy than the baseline in classifying both break index values and prominence. Similarly, bagging performs better than voting. Random forest does better than bagging for break index values, but not as well as bagging for prominence; however, both are substantially better than the baseline classifier. These results compare favorably with those reported on an identical task using similar amount of training data [WOK05]. However, apart from the classifier designs, there are significant differences between the two studies, in particular the feature normalization and acoustic pre-processing were performed differently, and results were reported on a single held-out set (not cross-validation results) in [WOK05].

Classifier	Break Index Values	Prominence
Baseline	81.65	78.85
Voting	82.56	79.91
Bagging	83.12	**80.44**
Random Forest	**83.27**	80.30

Table 13: Comparison of three ensemble-based classifiers with the baseline classifier for predicting break index values and prominence.

Temporal Constraints: In the baseline system, stacking the contextual local features from the previous and the following words improved the classification of break index to 81.80% (+0.15%) and prominence to 79.6% (+.75%). However, this comes at significant additional computational cost and so was not used in the rest of the studies. As expected, the additional temporal smoothing of observation densities with a hidden Markov model improved the performance, albeit marginally. The performance plateaus (or decreases) after 3rd order, as shown in Table 14. Since the additional gain over bagging comes at significant computational complexity, we chose to use the classifiers based on bagging for rest of the work.

Classifier	Break Index Values	Prominence
Baseline	81.65	78.85
Bagging	83.12	80.44
1st-order HMM	83.37	80.59
2nd-order HMM	83.49	80.73
3rd-order HMM	**83.50**	**80.77**
4th-order HMM	83.40	80.76

Table 14: Comparison of recognition accuracy between different orders of HMMs and bagging-based classifiers.

4.3 Prosody for Disfluency Detection

Section 4.2 described how a classifier can be build to detect prosodic events at the word-level with high accuracy. Note, the input to the classifier consisted of only acoustic features. However, linguists believe that humans use prosody in a way that is intricately interdependent on syntactic structure [Sel84, NV86, Ste00]. Prosody segments speech into groups of syllables and creates different levels of disjuncture between phrases. The Figure 33 illustrates a best case scenario from the corpus where the prosodic boundaries correspond with the boundaries of large syntactic constituents.
Prosodic breaks: *the weirdest fishing experience /3/ i ever had /4/ people to this day /4/ are still trying ...*
In general, the correspondence between prosody and syntax is not this close and there is no definite theory of syntax-phonology interface. However, several researchers have shown that prosodic knowledge can help spoken language processing (e.g., [SO97, VO93, NBW+02]).

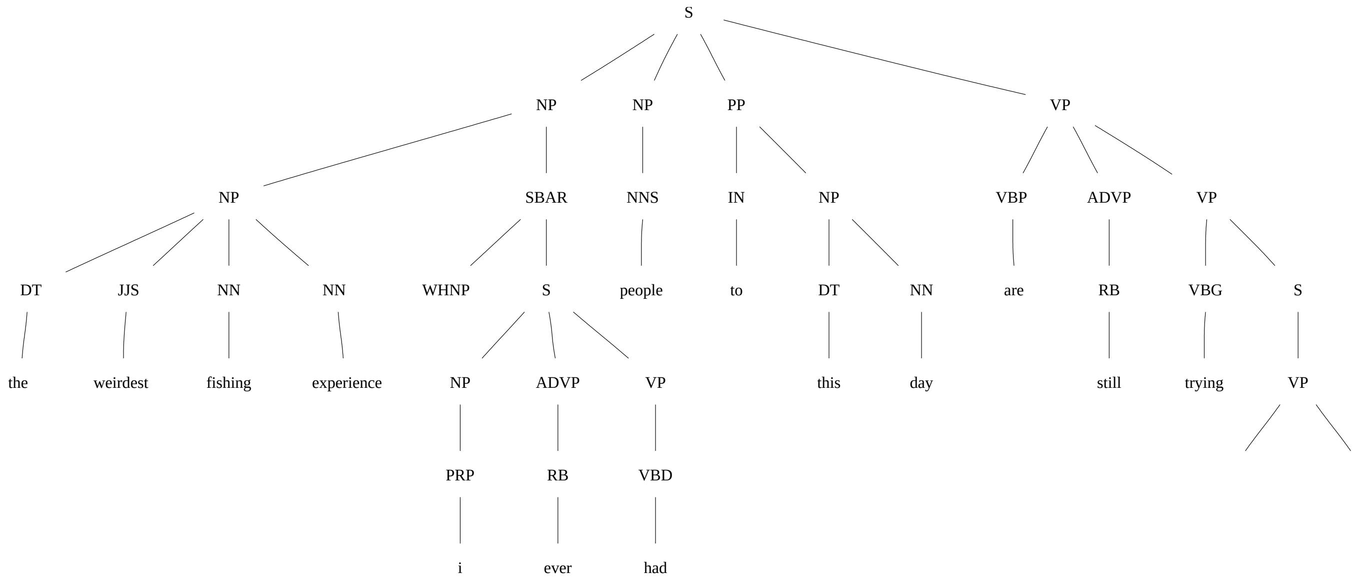

Figure 33: A partial utterance from corpus for which the prosodic structure and syntactic structures exhibit correspondence.

Here, we investigate a generative model of syntax-phonology interface for the task of disfluency detection. As mentioned previously, the interaction between metadata and parsing can be modeled by: (a) enriching the grammar, i.e., modeling it implicitly, (b) enriching the input by augmenting the words with special symbols, (c) parsing the "cleaned up" version of the input after detecting and excising the edited regions externally (e.g., [JC04]), or (d) re-ranking an initial set of hypothesis, possibly using additional cues that cannot be easily incorporated in a grammar. The re-ranking approach has proven to be successful in a number of applications; however, it requires a set of alternatives along with a good generative model to provide a compact and accurate set. Section 5 explores several ways of enriching the grammar specifically for disfluency detection. In contrast, here we investigate how to enrich the input representation to include prosodic cues.

Several researchers have found prosodic features to be useful in disfluency detection (e.g., [SBS97, LSS03]). In [LSS03], variations of pitch, energy and duration were used as input features along with strong local lexical constraints to detect disfluency. The posterior predictions of disfluency from a decision tree were quantized and combined with lexical cues. In contrast, we study a framework for incorporating prosodic structure as represented by ToBI via a context-free grammar; thus, we explore a different space of models that captures the interaction between syntax and prosody.

The prosody-syntax interface is an active area of linguistic and psycholinguistic research [Sel84, NV86, Ste00], providing several alternatives to model disfluency (e.g., close juncture disfavors disfluency [CDD97, Lic96]). As a preliminary study, we investigated the hypothesis that the ToBI label 'p' cues disfluency using simple and direct interface models. Here the disagreements between notions of disfluency in the prosodic and syntactic layers are modeled as noise in the probabilistic rules of CKY parser.

The training data for the experiment consisted of reference transcripts and the Switchboard treebank corpus [GHM92] with reference sentence units, in all about 100K words. The word- and phone-level time-marks from an ASR system were used to compute the raw prosodic features. With these inputs, the break index values for each word ending was predicted using the bagging-based classifier designed in Section 4.2. The prediction of 'p' was then transferred from the ASR-normalized words to the fringe of the treebanks by an edit distance alignment. CKY parsers were trained on the marked corpus and tested on the Fisher dev2 set with reference words and reference sentence units. The words in the test set were appended with preterminal tags using Ratnaparkhi's POS tagger [Rat96] and with the appropriate break values predicted from the acoustics. This formed the input to the parser for testing.

Four experiments were performed to test different trade-offs in precision and recall of 'p' with thresholds on posterior probability of 0.5, 0.6, and 0.7 respectively. Table 15 shows that the per word F-measure of edits in the case of 'p' tags improves significantly, however at the cost of lowering the overall F-measure of parsing. At lower thresholds, too many 'p' values are hypothesized, leading to an over-generalization. This decreases the overall F-measure. At higher thresholds, edit detection is substantially better than the baseline without altering the overall F-measure significantly. Perhaps, instead of tags based on hard decisions, a probabilistic method of integrating the prediction of 'p' would be more appropriate. The Figures 34 and 35 show two examples of edit detection in the test set. The first detects the edits correctly, while the second illustrates over-generalization. Note, the over-generalization may be related to the acoustic classifiers which have higher recall than precision. Clearly, further work is needed to understand the impact of prosodic structure on disfluency detection, particularly, in the context of ASR errors, larger amounts of training data, and the various schemes for enriching the input symbols and the grammar.

Input	Overall	Edits
(a) Baseline	67.9	22.9
(a) + $P_p > 0.5$	65.8	30.5
(a) + $P_p > 0.6$	66.4	30.4
(a) + $P_p > 0.7$	67.1	29.4

Table 15: Comparison of edit detection via parsing when the words are augmented with prosodic break value of 'p' and '1'.

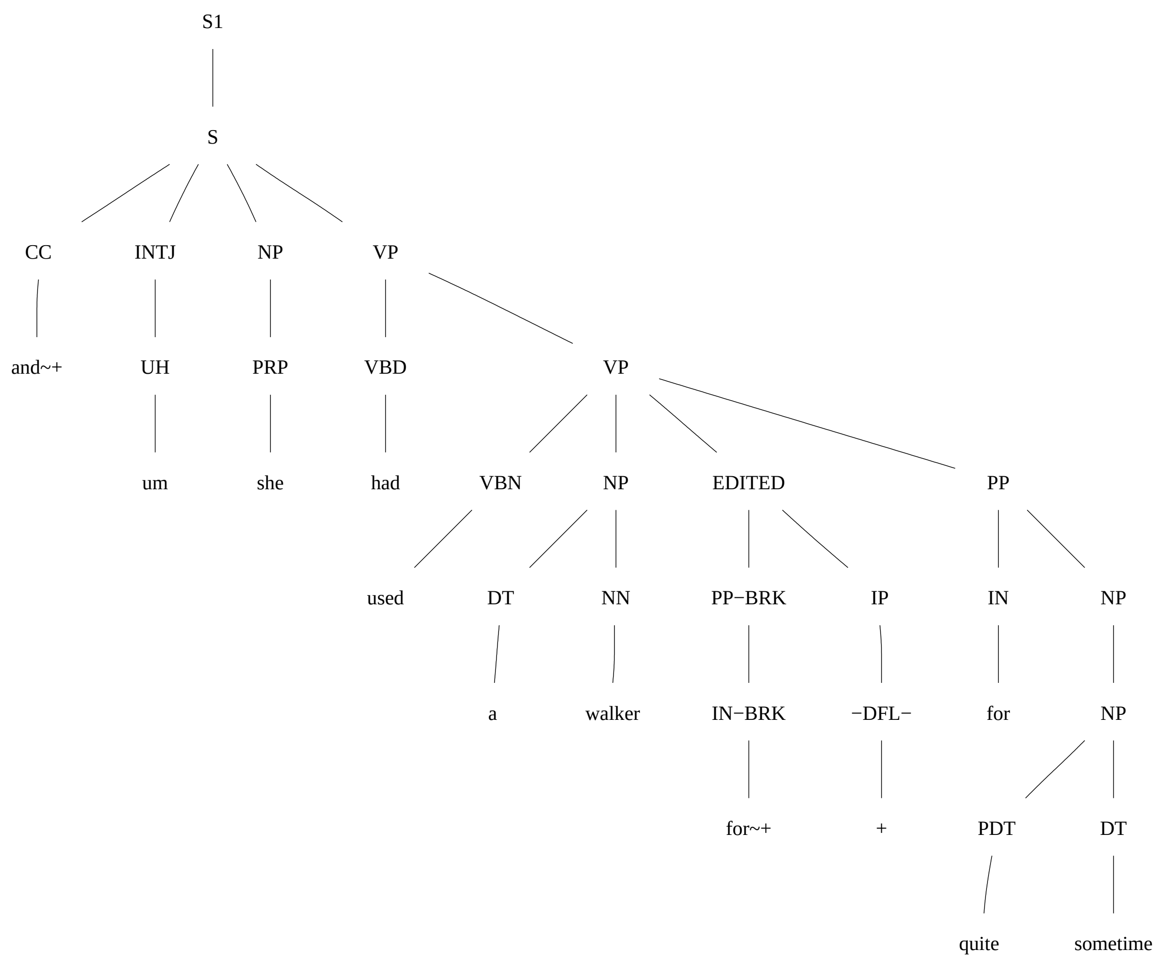

Figure 34: This example from the test corpus depicts a case of correct edit detection.

4.4 Prosody for Sentence Boundary Detection

In this section, we investigate the utility of prosodic events for detecting sentence boundaries. The focus is on testing whether the automatically generated posterior probabilities of break index values and prominence would aid an SU detection system. Even though the input features to the baseline MDE system described in Section 2 and the prosodic event detection system are similar, the intermediate symbolic layer from prosodic structure may provide additional benefits in much the same way phones help in word recognition. The prosodic events were incorporated into the SU detection system as illustrated in Figure 36.

The module in the upper path is the prosodic event classifier described earlier in Section 4.2, that we used to generate posterior probabilities for break index values and prominence at each word boundary. Each of the posterior probabilities was then cumulatively binned into 5 features (using threshold of 0.1, 0.3, 0.5, 0.7, and 0.9), in a fashion similar to the baseline MDE prosody model. The 25 quantized features (5 for each of the 3 break index and 2 prominence values) were then included in the maximum entropy based sentence boundary detection system. To understand the contribution of both the break index values and prominence, they were evaluated separately. Tests were also carried out on different WER conditions to test the hypothesis that the prosodic events might help when the accuracy of ASR transcripts are poor.

The results, in Table 16, do not exhibit any consistent improvements from the inclusion of prosodic information, even when the transcription accuracy degrades. The differences seen across the table are not statistically significant. In the next section, we investigate the interaction between prosodic and syntactic

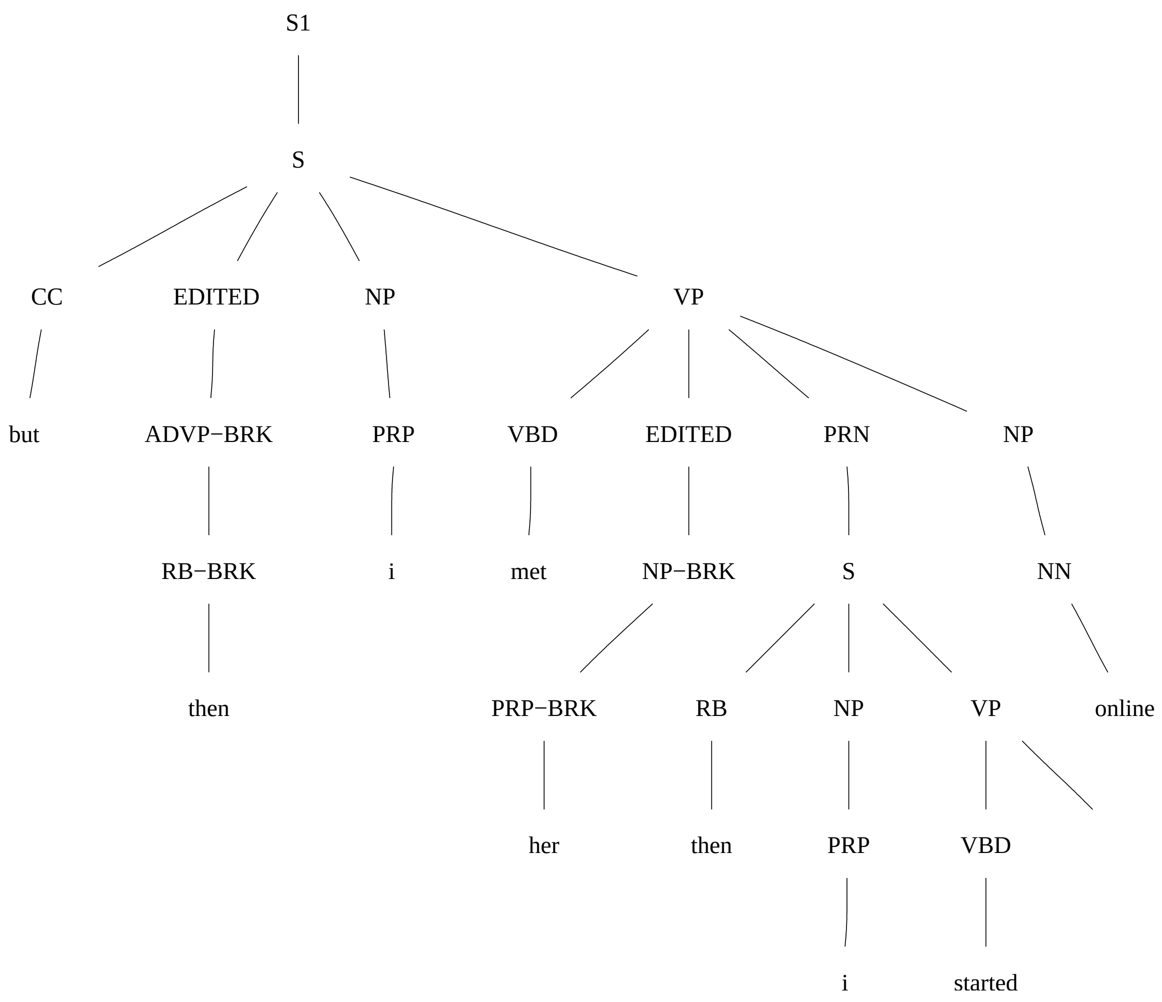

Figure 35: This example from the test corpus shows a case of over-generalization.

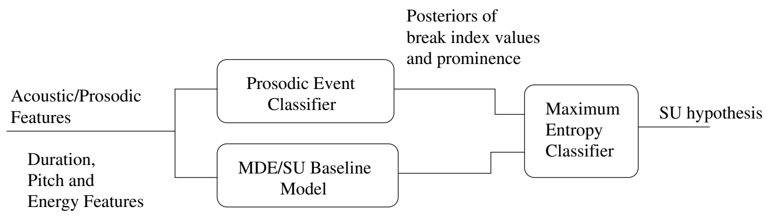

Figure 36: Approach used to incorporate prosodic events in the a baseline sentence unit detection system.

structures through the re-ranking framework, which is shown to improve the performance of SU detection in Section 6.

4.5 Prosodic-Syntactic Features

The lack of authoritative theories on syntax-phonology interface makes it difficult to develop a generative model for utilizing prosodic structure in downstream application such as metadata detection. Unlike previous work, we utilize an alternative reranking framework, which is discriminative in nature and allows complex features to be represented without the need to describe their distributions. Moreover, the specific re-ranking framework described in Section 6 has an efficient mechanism for feature selection, which

data set	transcripts	WER (%)	SU error rate (%)			
			baseline	+brk	+prom	+brk+prom
dev2	ref	0	26.99	26.66	26.75	26.75
	stt-ibmsri	11.7	35.94	35.58	35.88	35.82
	stt-sri	14.4	40.15	40.37	40.22	40.39
	stt-sri-0	23.0	43.03	42.94	42.77	42.68
	stt-sri-1pass	18.8	40.05	40.20	39.98	40.24
eval	ref	0	27.81	27.95	27.48	28.31
	stt-ibmsri	14.9	37.39	37.62	37.25	37.56
	stt-sri	18.6	41.35	41.43	40.96	41.37
	stt-sri-1pass	22.1	43.31	43.27	42.93	43.19

Table 16: Results (WER) of incorporating predictions of break index (brk) and prominence (prom) into a maximum entropy based sentence unit detection system on dev2 and eval sets.

alleviates the need for explicit analysis of a potentially large and complex feature space of syntactic and prosodic interactions.

The syntax-phonology interface has been studied extensively in linguistics and psycholinguistics (e.g., [Sel84, Sel95, Ste00, PP96, WG04]). The literature suggests that the likelihood of observing a large prosodic break at a word boundary depends on (a) size of the most recently completed constituent, (b) size of the upcoming constituent, and (c) the type of constituents involved. Watson and Gibson note that these characteristics could also be explained using preferences for head word attachment. Speakers tend to produce an intonational boundary when an incoming word does not attach locally to the pre-boundary word but attaches with a head mentioned earlier in the utterance. Likewise, listeners prefer not to attach an incoming word to a lexical head that is immediately followed by an intonational boundary. Prosodic boundaries are also influenced by the type of attachment. For example, often boundaries are placed between nouns and adjunct relative clauses that follow them.

Motivated by the above observations, we created a set of compound features from constituent syntactic and prosodic features. Prosodic features were computed using posterior probabilities of the break index values, and thus, consisted of three continuous values corresponding to break index values of '1', '4' and 'p'. The posteriors can be combined with syntactic features in a number of ways. We explored a few methods for quantizing log posterior probabilities to create a score for prosodic break, and combined it with our syntactic features. The syntactic features were computed from parses generated by Charniak's parser trained on reference transcripts of Switchboard corpus [CJ05] with dev1 as the development set.

1. **Type/Size of Constituents**: For each word, a categorical feature, comprised of five components, was created. This feature included: (a) the type and size of the biggest constituent that ends at that word, (b) the type and size of the biggest constituent that starts with the following word, and (c) the score associated with prosodic breaks.

2. **Head-Word Dependencies**: A second categorical feature captured head-word dependencies. Using the Charniak's head-word percolation rules, dependencies were computed for each word. The compound categorical variable consisted of the type of dependency relationship, the distance of the head word, and the prosodic break score.

These sets of features were computed on both ASR transcripts after applying the usual text normalization (e.g., mark's → mark 's). The computed features were mapped back to ASR words by ignoring features associated with new word endings introduced by the normalization (e.g., mark-f_x 's-$f_y \rightarrow$ mark's-f_y). Section 6 describes the application of these features in sentence boundary detection task using a re-ranking framework, and demonstrates how they can be successfully utilized.

5 Using MDE to Improve Parsing Accuracy

5.1 Introduction

As outlined in previous sections, speech is a wild and woolly beast that we are in the business of taming. For example, the speech utterance *and um she had used a walker for for quite sometime probably about six to nine months* contains several commonly used tokens that contribute to disfluency. Specifically, this utterance starts with the word *and*, repeats the word *for*, and inserts the filler *um*. Clearly, such speech utterances are a far cry from the fluent, grammatical structures that one finds in most texts and textual corpora (e.g., the Penn Treebank), as in the sentence *Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.*

What makes the analysis of speech output a challenge is that existing tools for text processing and evaluation do not match the problem of speech processing: grammars are optimized for text, not speech, and evaluation metrics are brittle. The latter issue was investigated in Section 3. What can be done to address the first issue? Two possible approaches are:

1. Enrich the speech input with a description of the change needed to make a more fluent version;
2. Enrich the grammar to cover disfluent constructions as well.

This section investigates both of these approaches, under varying conditions of training (trained versus untrained parsers) and evaluation (bracketed versus head-dependency).

Section 5.2 concerns input enrichment, i.e., the use of a Minimalist parser [Lin01] to interpret a marked up input string using both human-provided annotations and automatically assigned annotations [LSSH05] (MDE). Section 5.3 investigates grammar enrichment, i.e., the enrichment of a probabilistic context-free phrase structure grammar (PCFG) with modifications for disfluency, focusing on both unfinished phrases and syntactic parallelism in speech repairs. The final section, Section 5.4, concludes that the grammar of disfluency must be systematically related to the grammar of fluent language for these techniques to work as well as they do.

5.2 Enrich the Input

This section describes a new approach to handling input with automatic EDIT/FILLER markup, specifically MDE annotations automatically assigned using prosodic and lexical features [LSSH05]. Consider the following example:

```
INPUT: ... and I uh you know I guess as a young kid ...
```

Automatic MDE annotation produces an enriched input that we are able to feed directly into the Minipar parser [Lin01]. The enriched input for the example above is the following:

```
ENRICHED INPUT:  ...and <EDIT_ST> I <EDIT_END> <FL_ST> uh <FL_END>
<FL_ST> you know <FL_END> <EDIT_ST> I <EDIT_END> I guess as a young kid...
```

The goal is to apply a parser directly to the enriched input in order provide a more accurate parse than would otherwise be achieved by parsing the input without any markup. The choice of Minipar as our parsing approach is motivated by the need to take annotated text as input directly, without requiring training on the annotations or deletion and reinsertion of the annotations.

The three sections below describe the Minipar Parser, the MINI-BJD tool that converts Minipar output into a representation that is comparable to our treebank-style gold standard, and our evaluation results using this tool.

5.2.1 Minipar Parser

Minipar is a minimalist approach to parsing that does not follow the standard CKY or PCFG approach. Specifically, Minipar has a message-passing design where the grammar is encoded as a network in which nodes are syntactic categories that act as computing agents. The system has a simple grammar design and a minimal set of primitive relationships and operations. It is also efficient in that it attempts to produce structure that takes the least effort to generate. Minipar embodies the minimalist paradigm, where MERGE is induced through feature Percolation/Checking and MOVE is induced through binding between displaced element and trace. An important characteristic of Minipar is that the grammatical principles fall out from design (accounting for phenomena such as passive voice, raising, exceptional case marking, distransitive verbs, expletives, wh movement, that-trace, wh-island, etc.).

An advantage of Minipar is that the parser computation is monotonic: objects are constructed by MERGE are never changed. The upshot is that the parser is **exceedingly** efficient, with runtimes of 2238 words/second (on the JHU Linux machines, under a shared, but light load)—134,298 words per minute. With an average sentence length of 12.79 (including a large quantity of one word sentences in the speech transcripts), it parsed 175 sentences per second, or 10,496 sentences per minute. (On a dedicated machine, it would go a little faster.) In short, Minipar's speed enables very rapid development and evaluation cycles.

In addition, Minipar requires no training, which means we need not provide it a hand-parsed corpus in advance—an advantage for processing yet unseen languages. Finally, Minipar is directly applicable to marked-up (MDE) input, which allows one to test the impact of meta-data on parsing more directly than would be possible if tree-surgery were used. The downside of Minipar is that the scores are (not surprisingly) lower, since it is not trained on the data of interest.

Our approach to using the enriched input is to apply Minipar directly to the input sentence, and then to apply transformations, using a tool called MINI-BJD (described below), into a treebank style output that is directly comparable to the gold standard.

5.2.2 MINI-BJD Tool

This section presents the approach used for converting enriched Minipar output into a representation that is treebank-compatible. Consider the example given above, repeated here for convenience:

```
...and I uh you know I I guess as a young kid...
```

The enriched input is the following:

```
ENRICHED INPUT:  ...and <EDIT_ST> I <EDIT_END> <FL_ST> uh <FL_END>
<FL_ST> you know <FL_END> <EDIT_ST> I <EDIT_END> I guess as a young kid...
```

The Minipar output associated with this enriched input is the following:

```
(U (U and)
 (U <EDIT_ST>) (N i) (U <EDIT_END>)
 (U <FL_ST>) (N uh) (U <FL_END>)
 (U <FL_ST>) (C (I (N you) (V (V_[N] know)))) (U <FL_END>)
 (U <EDIT_ST>) (N i) (U <EDIT_END>)
 (C (I (N (NN i) (N_[C] guess) (P (P as) (N (Det a) (A young) (NN kid)))))))
```

The goal of MINI-BJD is to map this to an output representation that is more treebank-like:

```
(S1 (CC and)
 (EDITED (NP (PRP i)) (DISFL-IP +))
 (INTJ (UH uh))
 (PRN (S (NP-SBJ (PRP you)) (VP (VBP know))))
 (EDITED (NP (PRP i)) (DISFL-IP +))
 (S (NP-SBJ (PRP i) (VBP guess) (PP (IN as) (NP (DT a) (JJ young) (NN kid))))))
```

This representation can then be compared with a gold-standard tree that is in a comparable format:

```
(S1 (S (CC and)
 (EDITED (NP-SBJ (PRP i)) (DISFL-IP +))
 (INTJ (UH uh)) (PRN (S (NP-SBJ (PRP you)) (VP (VBP know))))
 (PRN (S (EDITED (NP-SBJ (PRP i)) (DISFL-IP +))
 (NP-SBJ (PRP i))
      (VP (VBP guess)))) (, ,) (PP (IN as) (NP (DT a) (JJ young) (NN kid)))))
```

The sequence of MINI-BJD transformations that converts the Minipar output above to a representation that is evaluable against treebank-style trees is shown below. Note that these transformations are achieved by very simple string-rewriting (not tree-rewriting) rules. For example, in Step 6, the first subject rule converts the string `(I (N you) ...)` into the string `(S (NP-SBJ (N you)) ...)` and the second rule then turns this string into `(S (NP-SBJ (PRP you)) ...)`. The full set of rules is given in the appendix Section B.

Minipar Input:

```
(U (U and)
 (U <EDIT_ST>) (N i) (U <EDIT_END>)
 (U <FL_ST>) (N uh) (U <FL_END>)
 (U <FL_ST>) (C (I (N you) (V (V_[N] know)))) (U <FL_END>)
 (U <EDIT_ST>) (N i) (U <EDIT_END>)
 (C (I (N (NN i) (N_[C] guess) (P (P as) (N (Det a) (A young) (NN kid)))))))
```

STEP 1: Apply edit rule `"(U <EDIT_ST>) # (U <EDIT_END>)" -> "(EDITED # (DISFL-IP +))"`
STEP 1 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (U <FL_ST>) (N uh) (U <FL_END>)
 (U <FL_ST>) (C (I (N you) (V (V_[N] know)))) (U <FL_END>)
 (EDITED (N i) (DISFL-IP +))
 (C (I (N (NN i) (N_[C] guess) (P (P as) (N (Det a) (A young) (NN kid)))))))
```

STEP 2: Apply filler rule `"(U <FL_ST>) (+ +) (U <FL_END>)" -> "(INTJ (+ +))"`
STEP 2 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (U <FL_ST>) (C (I (N you) (V (V_[N] know)))) (U <FL_END>)
 (EDITED (N i) (DISFL-IP +))
 (C (I (N (NN i) (N_[C] guess) (P (P as) (N (Det a) (A young) (NN kid)))))))
```

STEP 3: Apply parenthetical rule `"(U <FL_ST>) # (U <FL_END>)" "(PRN #)"`
STEP 3 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (PRN (C (I (N you) (V (V_[N] know)))))
 (EDITED (N i) (DISFL-IP +))
 (C (I (N (NN i) (N_[C] guess) (P (P as) (N (Det a) (A young) (NN kid)))))))
```

STEP 4: Apply complement rules `"(V_[N] #)" -> "(V #)"` and `"(N_[C] #)" -> "(N #)"`
STEP 4 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (PRN (C (I (N you) (V (V know)))))
 (EDITED (N i) (DISFL-IP +))
 (C (I (N (NN i) (N guess) (P (P as) (N (Det a) (A young) (NN kid)))))))
```

STEP 5: Apply nominal and adjectival rules `"(NN #)" -> "(N #)"` and `"(N # (A +))" -> "(N # (POS +))"`
STEP 5 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (PRN (C (I (N you) (V (V know)))))
 (EDITED (N i) (DISFL-IP +))
 (C (I (N (N i) (N guess) (P (P as) (N (Det a) (JJ young) (N kid)))))))
```

STEP 6: Apply subject rules `"(I (N #)" -> "(S (NP-SBJ #)"`, `"(NP-SBJ +)" -> "(NP-SBJ (POS +))"`, and `"(NP-SBJ # (N +)" -> "(NP-SBJ # (POS +)"`
STEP 6 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (PRN (C (S (NP-SBJ (PRP you)) (V (V know)))))
 (EDITED (N i) (DISFL-IP +))
 (C (S (NP-SBJ (PRP i) (N guess) (P (P as) (N (Det a) (JJ young) (N kid)))))))
```

STEP 7: Apply NP rule `"(N # (N +)" -> "(NP # (POS +)"`
STEP 7 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (PRN (C (S (NP-SBJ (PRP you)) (V (V know)))))
 (EDITED (N i) (DISFL-IP +))
 (C (S (NP-SBJ (PRP i) (N guess) (P (P as) (NP (Det a) (JJ young) (NN kid)))))))
```

STEP 8: Apply POS insertion rules `"(Det +)" -> "(POS +)"`, ”`''(P # (P +)" -> "(PP # (POS +)"`, and `"(V # (V +)" -> "(VP # (POS +)"`
STEP 8 Result:

```
(U (U and)
 (EDITED (N i) (DISFL-IP +))
 (INTJ (N uh))
 (PRN (C (S (NP-SBJ (PRP you)) (VP (VBP know)))))
 (EDITED (N i) (DISFL-IP +))
 (C (S (NP-SBJ (PRP i) (N guess) (PP (IN as) (NP (Det a) (JJ young) (NN kid)))))))
```

Additional Rules Applied ....
Final Result:

```
(S1 (CC and)
 (EDITED (NP (PRP i)) (DISFL-IP +))
 (INTJ (UH uh))
 (PRN (S (NP-SBJ (PRP you)) (VP (VBP know))))
 (EDITED (NP (PRP i)) (DISFL-IP +))
 (S (NP-SBJ (PRP i) (VBP guess) (PP (IN as) (NP (DT a) (JJ young) (NN kid))))))
```

The MINI-BJD algorithm makes use of a set of rules (those in Appendix B) and three input files: (1) a part-of-speech-tagged input file; (2) a backoff part-of-speech-tagged input file; and (3) metadata-annotated version of the input file (with no parts of speech). The output is a treebank-style form of the annotated input file.

The script called `apply_rules.pl` implements the MINI-BJD algorithm. To convert Minipar constituency trees to treebank-style parse trees this script is executed follows:

```
apply_rules.pl rules.txt input_file.pos backoff_pos.txt input_file > output_file
```

The `input_file.pos` file is in this format:

```
(EX there) (VBZ is) (DT a) (NN lot) (IN of) (VBG walking) (VBN involved) (RB here) ...
(RB so) (PRP you) (VBP have) (TO to) (VB walk) (RB all) (IN over) (NN campus)
(CC but) (PRP i) (VBP do) (RB n't) (RB really) (VB consider) (DT that) (JJ much) ...
```

These three lines correspond to the parts of speech for each of the words in the first three sentences of the `input_file`. (Our experiments used Charniak, but any tagger may be used.)

The `backoff_pos.txt` file contains any additional backoff POS tags that one may want to include, e.g., ones the automatic tagger could not find above. This can be an empty file or just a small set of tags. In our experiments, we have used a file whose first few lines look like this:

```
(MD 'd)
(MD 'll)
(VBP 'm)
(VBP 're)
(BES 's)
(VBP 've)
(. .)
(DT a)
(CC a-)
(NN a.)
(IN ab-)
(NN ability)
(JJ able)
(JJ abou-)
(IN about)
(RB above)
(RB abroad)
(UH absolutely)
```

The `input_file` is a set of Minipar trees (one tree per line) in this format:

```
(C (I (N there) (Be is) (N (Det a) (N lot) (P (P of) (N (N walking) (V (V_N involved) (N) (A (A here) ...
(C (A so) (I (N you) (Have have to) (V (V_[N] walk) (N (A all over) (N campus)))))
(C (A but) (I (N i) (Aux do) (V (A (A n't) (A really)) (V_I consider) (I (N that) (N (Det much) ...)))))
```

Note that this is not the usual pretty-printed version of the Minipar output. We used a script called `unindent.pl` on the output of Minipar to turn this into one tree per line. The usage of unindent.pl is as follows:

```
unindent pretty_printed_minipar_output_file > input_file
```

The `input_file` is the input to the MINI-BJD script `apply_rules.pl`.

The most complicated component of MINI-BJD is the `rules.txt` file (the rules from Appendix B). Rules are written as follows:

```
"<old string>" "<new string>"
```

For example, this rule:

```
"(N + + #)" "(N +) (N + #)"
```

converts this minipar output:

```
"(N dorm room)"
```

into this:

```
"(N dorm) (N room)"
```

The `+` refers to a single lexical item. The `#` refers to 0 or more sub-trees (where a lexical item is a degenerate case of a sub-tree). In the example above, the two `+`'s refer to `dorm` and `room` and the `#` refers to nothing (since nothing follows these two lexical items).

In addition, there is the POS marker:

```
"(V # (A + )" "(V # (ADVP (POS +)"
```

This is where the `input_file.pos` (and potentially the `backoff_pos.txt` file) would be used to replace the POS with the appropriate tag in the file. For example, the above rule converts this minipar output:

```
"(V (V lived) (A here))"
```

into this:

```
"(V (V lived) (ADVP (RB here)))"
```

Note that the original Minipar output was `"(V (V_A lived) (A here))"` but the rule above is executed after a previous rule converts `V_A` into `V`. That is, there is an assumed rule ordering associated with the MINI-BJD algorithm.

More specifically, the MINI-BJD algorithm consists of three steps:

1. The rules in `rules.txt` are applied in order of appearance.
2. Each rule is iteratively applied to each sentence until no more changes can be made. That is, `apply_rules.pl` will not move to the **next** rule until after it has executed the **previous** rule as many times as is necessary (until no further changes can be made).
3. The output is returned when all rules have been applied.

During the workshop, we found that, because of the second step above, it is easy to get into an infinite loop. That is, there is nothing to preclude the output form of a rule (the `<new string>`) from matching the input form of that very same rule (the `<old string>`). So one must be careful to make sure recursive rules are broken down into two (non-recursive) parts.

An example of a bad rule that induces the type of infinite loop described above is the following:

```
"(UH +)" "(INTJ (UH +))"
```

This rule is attempting to wrap words like `uh` inside of the `INTJ` (interjection) part of speech (to match the treebank). Unfortunately, this will create an infinite loop because the right-hand side of the rule matches the left-hand side after the rule has executed! Instead of this rule, it is better to write two non-recursive rules:

```
"(UH +)" "(INTJ (UH% +))"
"UH%" "UH"
```

Three additional (somewhat obscure) points need to be made about the rules file:

1. The special symbol `^` is a line delimiter (that can be used to indicate the beginning or the end of the line—or even both if used in the same rule), e.g.: `"^#^" "(S1 #)"`.

2. The only rules that are not applied iteratively are those that look exactly like these:

   ```
   "^#^" "<stuff>"
   "^#" "<stuff>"
   "#^" "<stuff>"
   "#" "<stuff>"
   ```

   I know, it's weird.

3. It turns out the beginning caret (`^`) is not needed when `#` is the first thing in the double-quotes, that is, the first and third rules above are equivalent and the second and fourth rules above are equivalent.

5.2.3 Evaluation of Minipar/MINI-BJD Results

There are two important results to report from our experiments with Minipar and MINI-BJD. The first is that the use of MDE markup significantly improves the parser performance. Specifically, the unlabeled (baseline) Minipar output gets a bracketed F-measure score of 34.67 on the unmarked input sentence hypotheses—but a score of 40.79 on the MDE marked-up input sentence hypotheses when scoring conversation sides with SParseval. This is the unlabeled measure (which is the only option because the Minipar categories do not match those of the gold-standard treebank trees).

Similarly, if MINI-BJD is applied, the output gets a labeled bracketed F-measure score of 46.62 when scoring the unmarked input sentence hypotheses at the conversation side level—but a score of 57.97 on the MDE marked-up input sentence hypotheses. The unlabeled bracketed F-measure score for MINI-BJD follows suit: 54.76 on the unmarked input sentence—but a score of 64.87 on the MDE marked-up input sentence.

From these results, we see that both MINI-BJD and Minipar achieve F-measure scores that are higher for the marked-up version than for the unmarked version (in both the unlabeled and labeled cases).[11] Moreover, the significant jump from 40.79 (on Minipar unlabeled output) to 64.87 (on MINI-BJD unlabeled output) is a substantial achievement given that only three of our six workshop weeks were dedicated to implementing the MINI-BJD algorithm—and no training was applied.

The results above are for human-transcribed speech and human-annotated MDE. The other four conditions follow the same pattern. The F-measure scores for all combinations are shown in Table 17 (Minipar unlabeled), Table 18 (MINI-BJD labeled), and Table 19 (MINI-BJD unlabeled).

The second important result is that, for the head-dependency F-measure, the head percolation tables (potentially) make a significant difference, depending on which parser is used. As shown above, the best

[11] All differences shown in the tables are significant, $p < .05$.

MDE Annotation	Human Transcripts		ASR Output	
	unmarked	**marked**	**unmarked**	**marked**
Human	34.67	40.79	32.42	37.08
System	30.17	36.08	28.02	32.51

Table 17: Unlabeled Minipar Bracketed F-measure Scores: Impact of metadata markup on both Human Transcripts and ASR Output

MDE Annotation	Human Transcripts		ASR Output	
	unmarked	**marked**	**unmarked**	**marked**
Human	46.62	57.97	41.52	50.60
System	45.10	53.91	39.81	46.72

Table 18: Labeled MINI-BJD Bracketed F-measure Scores: Impact of metadata markup on both Human Transcripts and ASR Output

MDE Annotation	Human Transcripts		ASR Output	
	unmarked	**marked**	**unmarked**	**marked**
Human	54.76	64.87	49.67	57.61
System	52.37	60.59	47.62	53.97

Table 19: Unlabeled MINI-BJD Bracketed F-measure Scores: Impact of metadata markup on both Human Transcripts and ASR Output

labeled-bracketed F-measure score for the MINI-BJD parser output is 57.97 on reference transcriptions with reference metadata. If we apply the head-dependency F-measure for this condition—using three different head-percolation scores (Hwa, Charniak, and Collins)—we get scores of 41.95 for Hwa, but only 40.65 for Charniak and 40.31 for Collins. The results for all four conditions, using Hwa, Charniak, and Collins head tables are given in Table 20 (Minipar unlabeled), Table 21 (MINI-BJD labeled), and Table 22 (MINI-BJD unlabeled).

Note that the Hwa table also yields higher scores than the other two tables on the unlabeled output (both Minipar and MINI-BJD). On the labeled MINI-BJD output, the Hwa table yields consistently higher scores than Charniak, and—in most cases—higher than Collins. (In the two cases where the Hwa table yields lower scores than Collins, the differences are very small: 54.87 versus 54.88 and 56.65 versus 56.87.) Interestingly, the Charniak table yields lower scores than Collins for the unlabeled output (both Minpar and MINI-BJD), but higher scores than Collins for the labeled MINI-BJD output. Note that the largest differences were between Hwa and Charniak on the MINI-BJD (labeled and unlabeled) system-annotated ASR transcripts and also on the MINI-BJD (unlabeled) system-annotated human transcripts.

Although the experiments above indicate that Hwa's tables generally yield higher scores for MINI-BJD output, this trend does not hold in experiments on alternative statistical parsers (see Section 3). Specifically, the top-scoring parser in Section 3 (the Charniak parser) yields a labeled-bracketed F-measure of 88.07 on reference transcripts with reference metadata and using edit metadata to remove edited regions prior to parsing. Under the same conditions, the head-dependency F-measure is 84.80 for Hwa, 85.68 for Charniak, and 85.51 Collins. Similar results are obtained for the other parsers evaluated in that section. So the question to ask is: why do Hwa's tables yield higher scores for MINI-BJD, but not for these other parsers?

Upon closer inspection, it appears that Hwa's tables expect short, fat (n-ary branching) trees—ones that Minipar is, in fact, proficient at generating. This is because Hwa's tables were designed to provide a

MDE Annotation	Human Transcripts		ASR Output	
	unmarked	marked	unmarked	marked
Human	20.63/19.58/20.13	22.36/20.84/21.54	18.37/17.23/17.86	19.84/18.40/19.11
System	19.53/18.47/19.26	21.30/19.77/20.75	18.96/17.64/18.44	18.96/17.64/18.44

Table 20: Unlabeled Minipar Head-dependency F-measure Scores Using Different Head Tables: Hwa/Charniak/Collins

MDE Annotation	Human Transcripts		ASR Output	
	unmarked	marked	unmarked	marked
Human	39.59/38.33/38.03	41.95/40.65/40.31	34.49/33.47/32.75	36.14/35.12/34.45
System	39.06/37.80/37.46	40.40/39.14/38.71	34.06/30.21/29.36	34.95/30.84/30.12

Table 21: Labeled MINI-BJD Head-dependency F-measure Scores Using Different Head Tables: Hwa/Charniak/Collins

MDE Annotation	Human Transcripts		ASR Output	
	unmarked	marked	unmarked	marked
Human	54.87/51.78/54.88	56.65/54.07/56.87	47.83/45.20/46.93	49.06/46.98/48.43
System	54.02/50.80/53.72	54.65/52.06/54.54	47.30/40.76/46.31	47.50/45.53/46.76

Table 22: Unlabeled MINI-BJD Head-dependency F-measure Scores Using Different Head Tables: Hwa/Charniak/Collins

mapping from constituency trees to dependency trees (Hwa's const2dep algorithm); they were not designed for the evaluation of constituency-tree output. Specifically, these tables are geared toward characterizing appropriate dependency trees where, e.g., "GO" is the head of "TO GO". The Charniak/Collins' tables, on the other hand, expect tall, thin trees—such as the nearly-binary treebank-style trees produced by Charniak-style parsers—and, thus they are geared toward evaluating syntactic trees where, e.g., "TO" is the head of "TO GO".

To test this point further, two versions of MINI-BJD were evaluated, one without binary branching of verb-phrases (an earlier, lower scoring version of MINI-BJD) and one with binary branching of verb-phrases (the MINI-BJD version evaluated above). In the un-binarized case, the differences between the Hwa and the other two tables were even greater than those of the binarized case given above. For example, the head-dependency F-measure scores for the unlabeled, unbinarized human-annotated marked-up (human) transcriptions were 56.32 for Hwa and 49.31 for Charniak—a difference of almost 7 points! In the binarized version of this same condition, the F-measure scores were 56.65 for Hwa and 54.07 for Charniak—a much smaller difference of 2.58 points. The Hwa/Collins difference is also larger in the unbinarized case than in the binarized cases given earlier. Head-dependency F-measure scores for the unbinarized version of MINI-BJD (using Hwa/Charniak/Collins) are shown in Table 23 (for the unlabeled/unbinarized case) and Table 24 (for the labeled/unbinarized case).

To understand why the differences between Hwa and Charniak/Collins are so large in these tables, consider the (un-binarized) MINI-BJD parse of the sentence *I haven't got my check* (taken from our development set). If the Hwa Head percolation table is used on this sentence—and also on the corresponding Gold Standard sentence—the head-percolation yields the representation shown in Figure 37. Note, in particular, that the Hwa tables force the verb *got* to percolate up to the top S node in both trees.

The representational analog to this—using either the Charniak or the Collins Head percolation table—is shown in Figure 38. Unlike the Hwa tables, the Charniak/Collins tables percolate the word *have* up to

MDE Annotation	**Human Transcripts**		**ASR Output**	
	unmarked	**marked**	**unmarked**	**marked**
Human	39.45/34.78/34.59	41.80/36.77/36.53	34.35/30.32/29.69	36.01/31.69/31.10
System	38.92/34.22/34.01	40.25/35.37/35.07	34.06/30.21/29.36	34.95/30.84/30.12

Table 23: Head-dependency F-measure Scores (Labeled, Unbinarized MINI-BJD): Hwa/Charniak/Collins

MDE Annotation	**Human Transcripts**		**ASR Output**	
	unmarked	**marked**	**unmarked**	**marked**
Human	54.45/47.20/50.15	56.32/49.31/51.86	47.50/41.18/42.69	48.76/42.80/44.02
System	53.59/46.23/49.02	54.32/47.45/49.75	46.96/40.76/42.12	47.24/41.51/42.54

Table 24: Head-dependency F-measure Scores (Unlabeled, Unbinarized MINI-BJD): Hwa/Charniak/Collins

the top S node in only the Gold-Standard tree (where the VP is nearly binarized, in comparison to the MINI-BJD output). The result is that the Hwa head percolation table yields a better match than that of the Charniak/Collins head percolation table. This is reflected in the F-measure score of 0.75 for Hwa and 0.375 for Charniak/Collins.

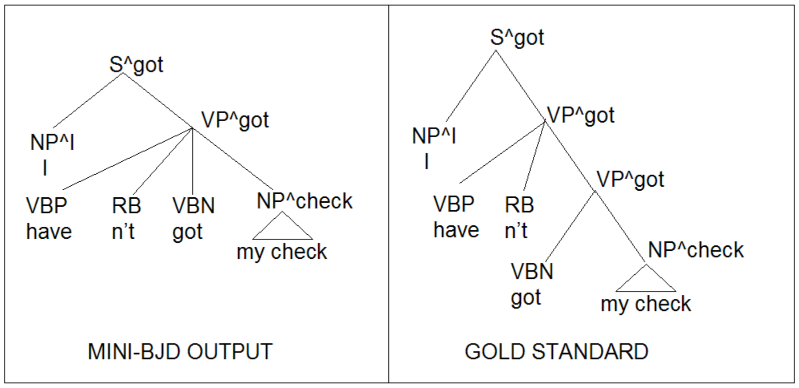

Figure 37: Impact of Head Percolation Tables: Hwa F-measure Score is 0.75

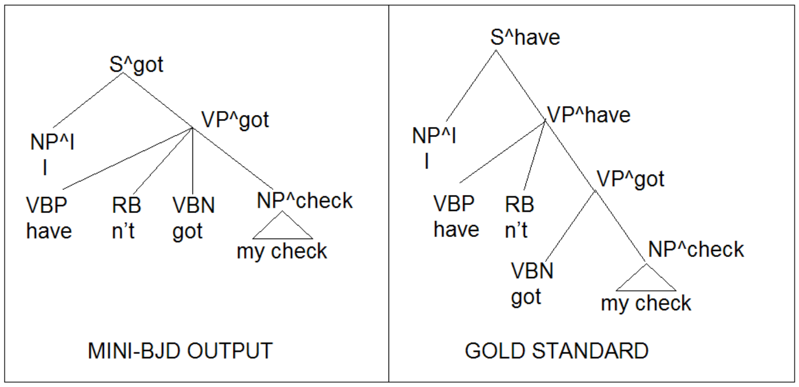

Figure 38: Impact of Head Percolation Tables: Charniak/Collins F-measure Score is 0.375

Examples such as this one, which occur frequently in the un-binarized version of MINI-BJD, lead to significantly higher head-dependency scores when Hwa's percolation tables are used. In the binarized version of MINI-BJD, the differences between Hwa and Charniak/Collins were less dramatic. It is expected that additional binarizing of other phrases (e.g., PP, NP, ADVP) would reduce the differences even further. However, the point here is clear: One must fully understand the conventions of a given parser, the format of the Gold Standard, and the standards for head percolation, before drawing definitive conclusions about parser accuracy results.

5.3 Enrich the Grammar

The alternative to enriching the *input* is to enrich a parser's *grammar*. The goal of such enrichment is, of course, to make the grammar a better model of the kind disfluent language exemplified in Section 5.1. Speech repairs, a challenging disfluency that involves structural parallelism, was a focus of our team. In particular, the PASSED team considered two ways of adapting probabilistic context-free phrase structure grammars (PCFGs) to speech repairs:

1. distinguishing a separate category of 'unfinished' phrases
2. identifying a syntactic category for reparanda (edited phrases)

These ideas derive from the work of Levelt [Lev83]. Levelt views speech repairs as the conjunction of an *unfinished* reparandum with a finished repair. Between the two conjuncts an optional editing phrase (such as "er" or "I mean") can appear. This viewpoint is codified in his Well-Formedness Rule (WFR) for repairs:

Well-formedness rule for repairs (WFR) A repair $\langle \alpha\gamma \rangle$ is well-formed if and only if there is a string β such that the string $\langle \alpha\beta and^{*}\gamma \rangle$ is well-formed, where β is a completion of the constituent directly dominating the last element of α. (*and* is to be deleted if that last element is itself a sentence connective)

In the course of building a parser for task-oriented dialogs, David McKelvie [McK98a, McK98b] implemented Levelt's WFR using metarules that derive grammar rules from other grammar rules. McKelvie's formulation adds a new feature (in the sense of constraint-based grammar [Shi92]) called `abort` which can take values true and false. For a given grammar rule of the form:

$$A \rightarrow BC$$

a metarule creates other rules of the form:

$$A\,[abort = X] \rightarrow B\,[abort = false]\,C\,[abort = X]$$

which say, in effect, that the constituent A is aborted just in case the last daughter C is aborted. This extension says which constituents are known to be aborted. The WFR is then implemented by a rule schema which McKelvie called `disfl(2)`:

$$X \rightarrow X\,[abort = true]\,(AFF)X$$

In the `disfl(2)` schema, the optional element AFF conjoins an aborted X-phrase (the reparandum) with a finished X-phrase (the repair) that comes after. McKelvie [McK98a] used this analysis to obtain parse trees for the Edinburgh MapTask corpus. Figure 39 from this corpus shows how an aborted prepositional phrase (PP) 'above th-' is repaired with another PP, one that is qualified by the adjective 'directly'. McKelvie's formulation has the advantage of being declarative; it expresses Levelt's constraint but does not specify a particular method of satisfying it [Hin83].

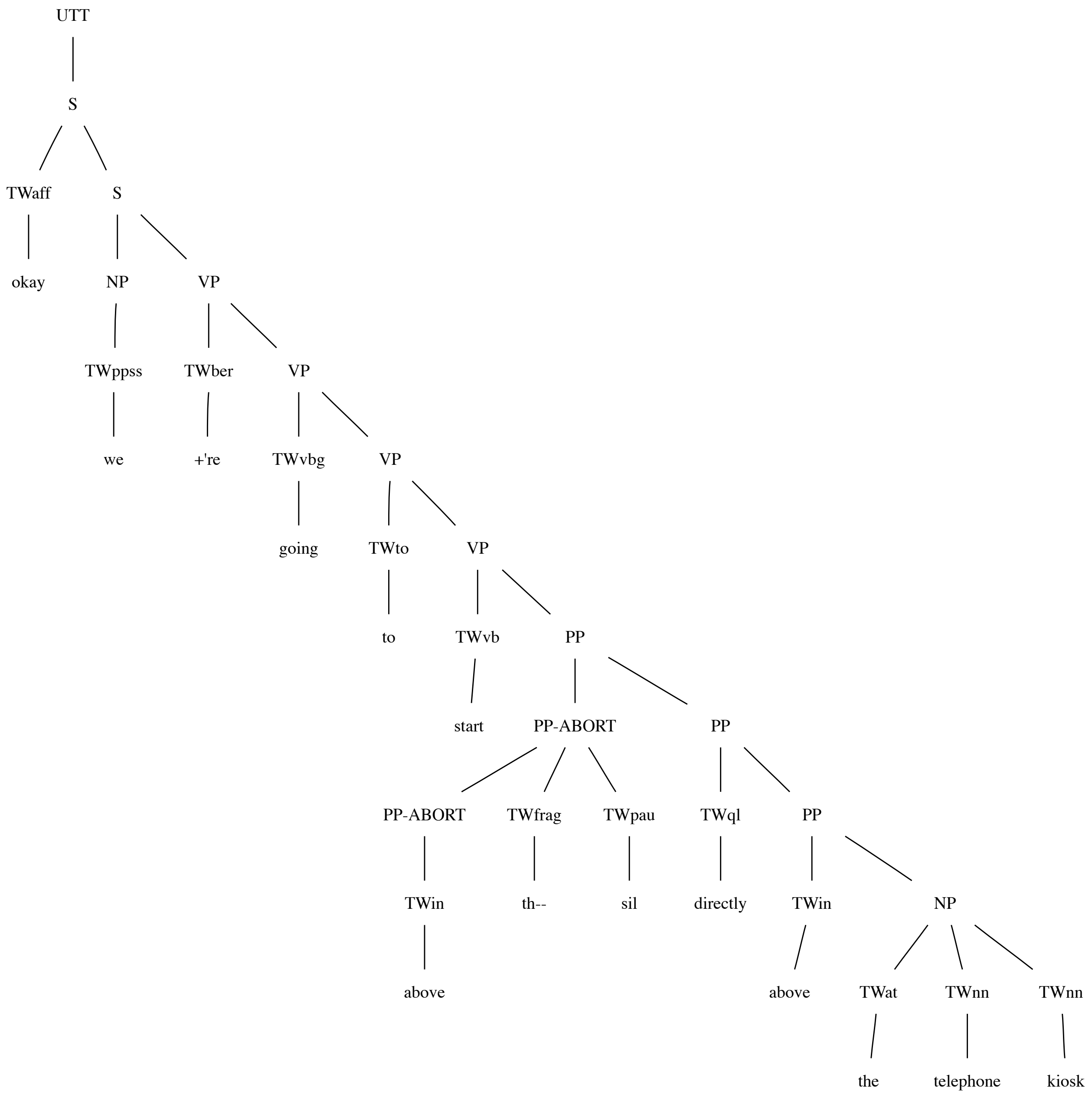

Figure 39: MapTask example of McKelvie's metarule

5.3.1 Using Levelt's Idea

It is possible to apply (McKelvie's formulation of) Levelt's idea to spontaneous speech by enriching the kinds of treebank grammars available with annotations; this approach is thus trained, in a supervised way, in a way that the untrained-parser-on-enriched data method of Section 5.2 is not. To properly express the WFR, as formalized by `disfl(2)`, the treebank trees were automatically annotated in two ways.

Unfinished phrases: Levelt's idea of repair-as-conjunction-of-unfinished-phrase relies on a distinguishable category of unfinished phrases. The policy of the Penn Treebank annotation scheme [MSM93] is to annotate the lowest node that is unfinished; it is straightforward to propagate this mark upwards on trees in the treebank to mirror the action of *abort* = *true*. Such propagation results in annotated trees, like the one in Figure 40 where the PP is clearly unfinished, since it has only one word. Propagating the -UNF annotation also yields a treebank grammar that is a better model of disfluency utterance, since it factors the model into two kinds of sublanguages that, while being related, have different properties.

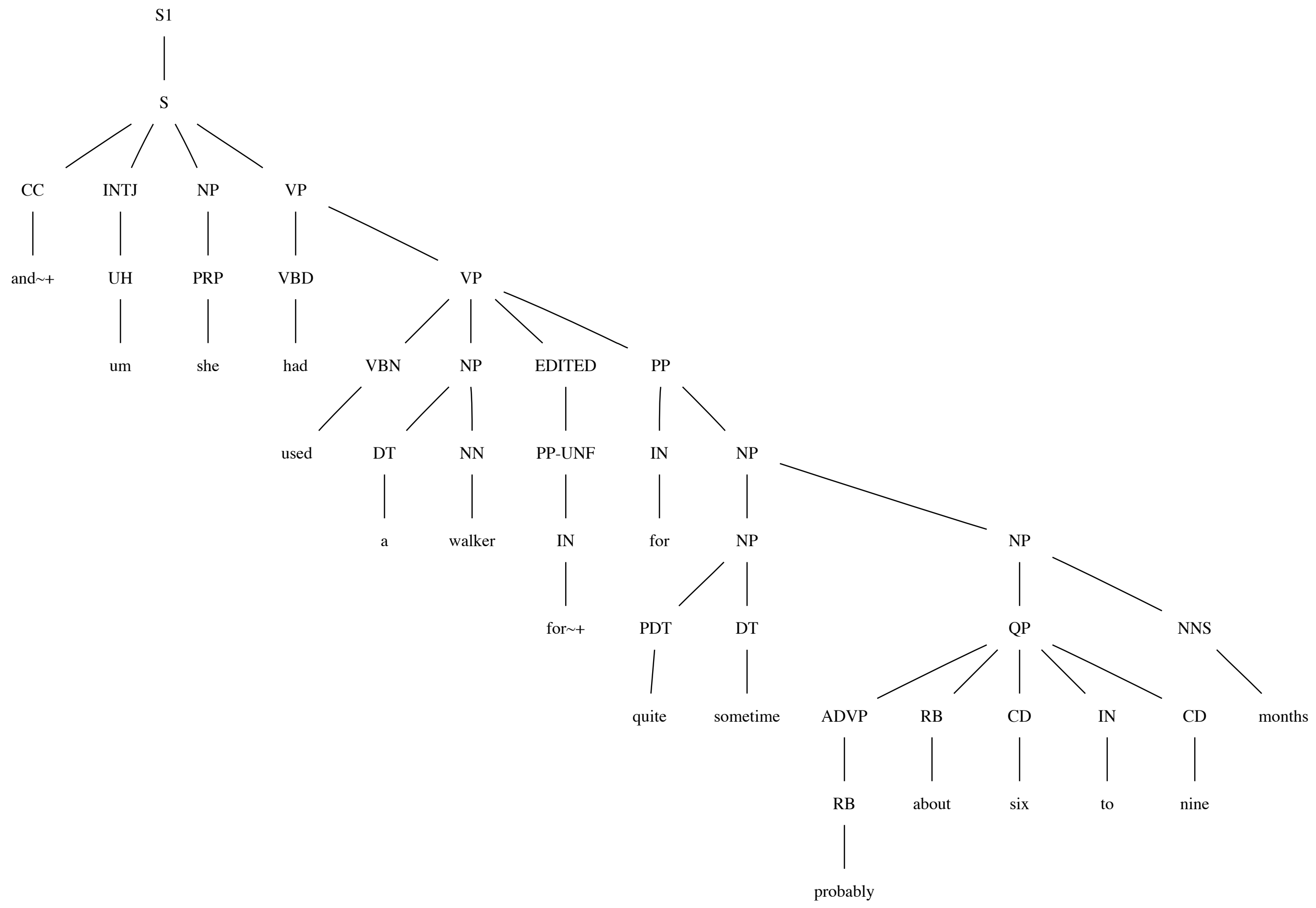

Figure 40: Unfinished Prepositional Phrase

Syntactic Parallelism: The other key element of Levelt's WFR is the idea of *conjunction* of elements that are in some sense the same. This can be implemented by phrase structure rules where the syntactic categories are the same. However, in the Penn Treebank annotation scheme, the reparanda always receive the label EDITED. This means that the syntactic category of the reparandum is hidden from any rule which could match it with that of the repair. Adding an additional mark on this EDITED node (a kind of daughter annotation) rectifies the situation, as depicted in Figure 41. With daughter annotation of EDITED nodes, a treebank PCFG can represent the generalization that, more often than not, speech repairs respect syntactic category.

5.3.2 Results

The PaSSED team evaluated these ideas by training on the Switchboard corpus and testing on the dev2 data set. The performance of a standard CYK chart parser, on the correct part of speech tags,

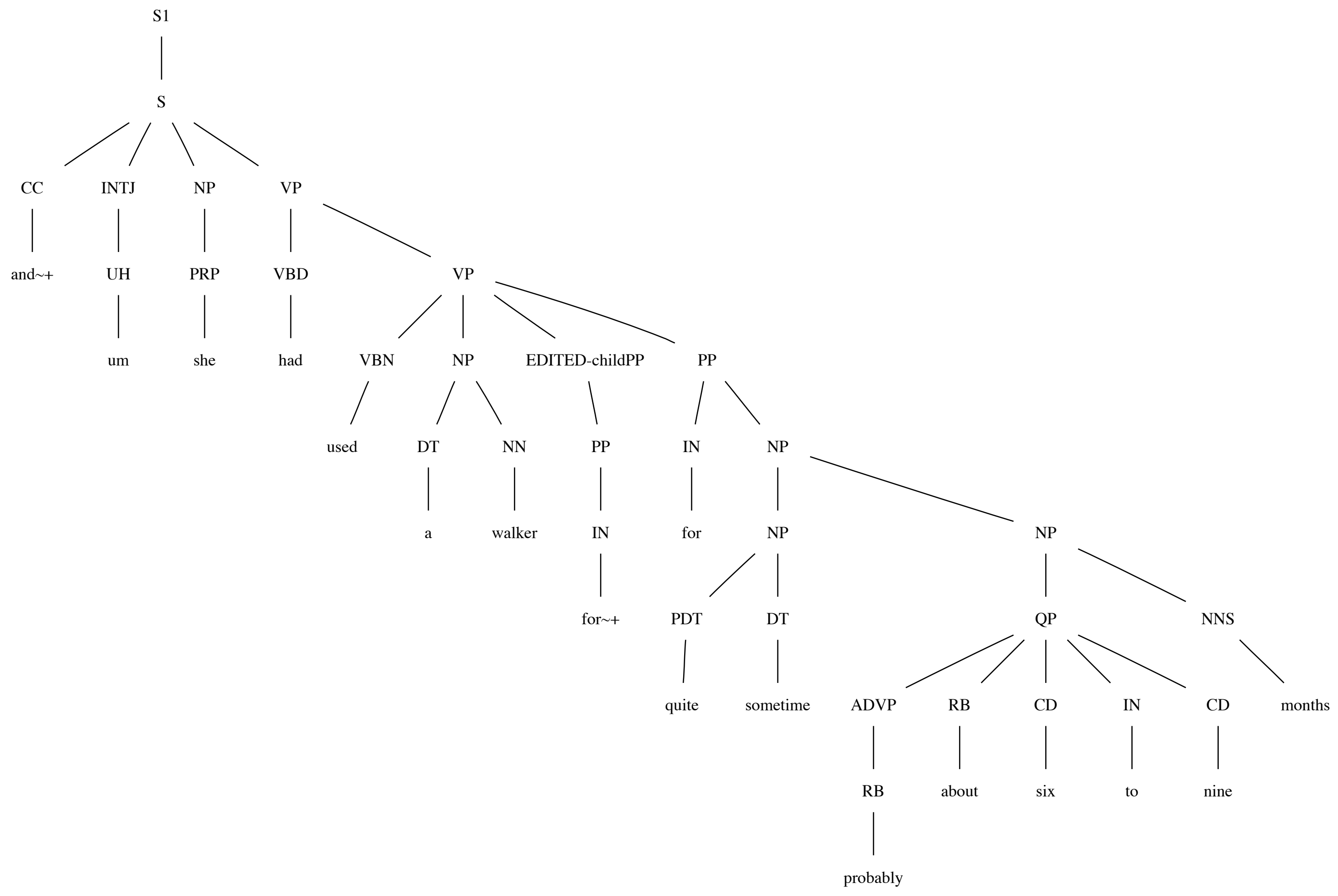

Figure 41: Daughter annotation

when trained on various combinations of annotated data, is presented in Table 25. This table reports the standard Parseval F-measure that combines precision and recall of labeled constituents. All of the annotations, despite splitting states, cause performance to go up on this metric, probably because dividing the grammar into two sub-models (of finished and unfinished speech) reflects a real difference in the language. The EDIT-finding F is the per-word F-measure quantifying how much better the grammar gets at deciding whether a word should be part of a constituent labeled EDITED or not. This metric shows that some of the improvement in overall Parseval is due to improvement on EDITED constituents.

Annotation	**Parseval F**	**EDIT-finding F**
baseline	71.15	23.0
-UNF propagation	71.75	32.0
daughter annotation	71.59	32.9
both	72.45	42.3

Table 25: Independent improvement from both features

A similar experiment was carried out using Charniak's non-reranking lexicalized parser. The results summarized in Table 26 suggest that the utility of implementing Levelt's WFR does not derive from, but is rather synergistic with the kind of head-head lexical dependencies that Charniak's more sophisticated parser uses.

Annotation	Parseval F	EDIT-finding F
baseline	82.06	53.3
-UNF propagation	79.96	59.5
daughter annotation	78.55	58.0
both	77.90	61.3

Table 26: Charniak as an improved EDIT-finder

Finding the interruption point (IP) of a speech repair is key to its successful recognition. If this point, which is necessarily the right boundary of the EDITED constituent, could be found with high accuracy by acoustic means, news of its location could be propagated upwards and used in the analysis of the entire repair. The oracle results in Table 27 show how such hypothetical perfect interruption point information would combine with the annotations already described to yield even higher performance. Section 4 discusses efforts toward prosodically-based prediction of interruption points that could be used in an automatic system.

Annotation	Parseval F	EDIT-finding F
oracle interruption point	75.84	81.7
oracle interruption points combined with -UNF and EDIT-childXP	76.53	87.9

Table 27: -UNF and daughter annotation would synergize with IP

5.4 Conclusions

We find that a comparatively small number of post-processing rules bring the output of an untrained parser into correspondence with the treebank annotation for spontaneous speech, including the hard cases of repair. These rules can take advantage of MDE markup in the input string. They also interact intricately with the head-percolation rules used evaluations where sentence boundaries cannot be taken for granted, and do not require access to the innards of the parser. On the other hand, if such access is available, conventional PCFGs can improve their handling of repairs by implementing two aspects of Levelt's well-formedness rule. This benefit extends to more sophisticated lexicalized parsers, and would synergize with any improvements in automatic interruption point detection.

These particular conclusions underlie a broader implication: that disfluency is not noise. Rather it has a structure that can be used, either post-hoc or pre-hoc, to recognize it and its surrounding syntactic structure.

6 SU Reranking Experiments

This section specifically addresses the task of SU boundary detection. Previous approaches to this task have used finite-state sequence modeling approaches, including Hidden Markov Models (HMM) [SSHTT00] and Conditional Random Fields (CRF) [LSSH05] described in Section 2. While these approaches have yielded good results, the characteristics of this task make it especially challenging for Markov models. Average SU length for conversational telephone speech is around 7; hence, most of the time the previous states will be for non-boundary positions, providing relatively impoverished state sequence information. Thus, in [LSSH05], a Maximum Entropy (MaxEnt) model that did not use state sequence information, was able to outperform an HMM by including additional rich information. The approach taken here is to rely upon a baseline model [LSS$^+$04b, LSS$^+$04a] to produce n-best lists of possible segmentations, and extract disambiguating features over entire candidate segmentations, with no Markov assumption. We present an effective n-best candidate extraction algorithm, along with a detailed investigation of the utility of a range of features for improving SU boundary detection.

In the next section we provide background on the SU detection task, baseline models, and the general reranking approach. We then present our n-best extraction algorithm and the features we investigated, followed by empirical results under a variety of conditions.

6.1 Background

In this section, we briefly summarize aspects of the baseline SU detection models described in Section 2 that are pertinent to our reranking approach and then describe that approach.

6.1.1 MDE Baseline Models

The ICSI/SRI/UW MDE system [LSS$^+$04b, LSS$^+$04a] is the baseline used for this research on SU reranking, which is described in detail in Section 2. Recall that the SU detection task can be thought of as a classification task that determines whether an interword boundary is an SU event boundary or not. To detect SUs, multiple knowledge sources are utilized, including prosodic and textual information. Typically, at each interword boundary, prosodic features are extracted to reflect pause length, duration of words and phones, pitch contours, and energy contours. These prosodic features are modeled by a decision tree classifier, which generates a posterior probability of an event given the feature set associated with a boundary. Textual cues include the contextual information of the words, their corresponding classes, and syntactic information. For SU detection, the baseline system is a linear combination of HMM and MaxEnt posterior probabilities.

6.1.2 Maximum Entropy Reranking

We used the MaxEnt reranker documented in [CJ05], which optimizes parameter weights with respect to a regularized conditional likelihood objective. The approach maximizes the conditional probability mass[12] given to those candidates in the n-best list that have the highest accuracy[13]. A Gaussian regularizer is used to control for overtraining, and the regularizer constant is set empirically. Formally, let the training data consist of N examples, let Y_i be the set of candidates for example i, and let $\hat{Y}_i$ be the set of maximum accuracy candidates for example i, i.e.,

$$\hat{Y}_i = \{y \in Y_i \mid \mathrm{ACC}(y) = \mathrm{argmax}_{y' \in Y_i} \mathrm{ACC}(y')\} \tag{7}$$

Then the parameter estimation minimizes the regularized negative conditional log probability:

$$NLL_R(\theta) = -\sum_{i=1}^{N} \log \left(\sum_{y \in \hat{Y}_i} \mathrm{P}(y \mid Y_i) \right) + R(\theta), \tag{8}$$

where $\mathrm{P}(y \mid Y_i)$ is the conditional likelihood of y given the normalized distribution over Y_i, and $R(\theta)$ is the Gaussian regularizer. By normalizing the candidate probabilities over the n-best lists, this approach uses a global normalization akin to that in CRFs. The flexibility of extracting features over the entire sequence differentiates the current approach from the baseline CRF model. See [CJ05] for more details.

[12] The probability is conditioned on the n-best list, i.e., each candidate's probability is normalized relative to the n-best list.

[13] Since the n-best list is not guaranteed to contain the truth, there may be multiple maximum accuracy candidates in the list.

Baseline Posterior x	Percent Accurate	Percent of Word Boundaries
$x > 0.95$	97.9	8.2
$x < 0.05$	99.4	77.0
$0.05 \leq x \leq 0.95$	78.1	14.8

Table 28: Accuracy of labels produced by baseline model versus posterior probability on dev2 set.

6.2 Reranking for SU Detection

In this section, we present n-best list generation and feature extraction for reranking.

6.2.1 N-best Candidate Extraction

SU detection is done over conversation sides (see Section 3.2.1 for a general description of the data resources used and Section 6.2.3 for data usage in these particular studies) which are far lengthier sequences than those in NLP tasks that are usually approached in a reranking paradigm, such as parsing. For the data processed in this workshop, the average length of a conversation side is more than 500 words, with the maximum over 1000. Since every word boundary is a potential segmentation point, the number of possible segmentations over the conversation side is exponential in the length of the string. A brute force approach to generating the 1000 best segmentations over these conversation-side sequences, i.e., choosing the 1000 highest scoring segmentations based on the baseline posterior probabilities, yields an oracle F-measure accuracy of just a percent or so better than the 1-best. As a result, other methods for generating the n-best lists were investigated.

The baseline classifier marks as segmentation points all positions with a posterior probability of 0.5 or higher. Table 28 shows the accuracy of the baseline model when the posterior probability of a segmentation boundary is either very high or very low, versus somewhere in the middle. For very high or very low posterior word positions, the baseline model is nearly always correct; whereas, other positions have much lower accuracy for the classifier. This suggests an approach where high and low posterior points are fixed to the classes provided by the baseline classifier.

We will now present a number of graphs that were used to establish our approach. Given that we intend to parse the segmented utterances, we first look at the maximum sentence length using very simple cutoff techniques. The simplest way to segment the word stream is to set a probability cutoff and mark a boundary at any location where the posterior probability is above this cutoff. Figure 42 explores the precision of such high-probability boundaries and the resulting segment lengths and shows how maximum segment length varies with the precision of the boundaries. Each data point corresponds to a unique probability cutoff; the orange numbers document a few of them. Figure 43 shows the average sentence length versus precision and recall.

If we now go to a two-stage approach, by first setting a posterior probability cutoff, and then continuing to segment when segment length is over some maximum length. This is done by recursively setting a boundary at the location of the highest posterior probability in all segments that are longer than the set maximum. We ran the experiment for different initial probability cutoffs and a range of segment length limits. Figure 44 shows the resulting segmentation's precision under these various conditions, in which a probability cutoff of .95 and a length limit of 50 provided a reasonable tradeoff that gives relatively high precision.

Figure 45 is a more detailed look at the behavior of the segmentations under high probability cutoffs. It also includes the condition where there is no high probability cutoff, i.e., where boundaries are only set to limit the length of segments. The fact that this condition performed measurably worse than the others verifies that our idea of first segmenting with a probability cutoff has merit. The very high probability cutoffs have better precision when sentences are allowed to be long, but are overtaken by lower cutoffs

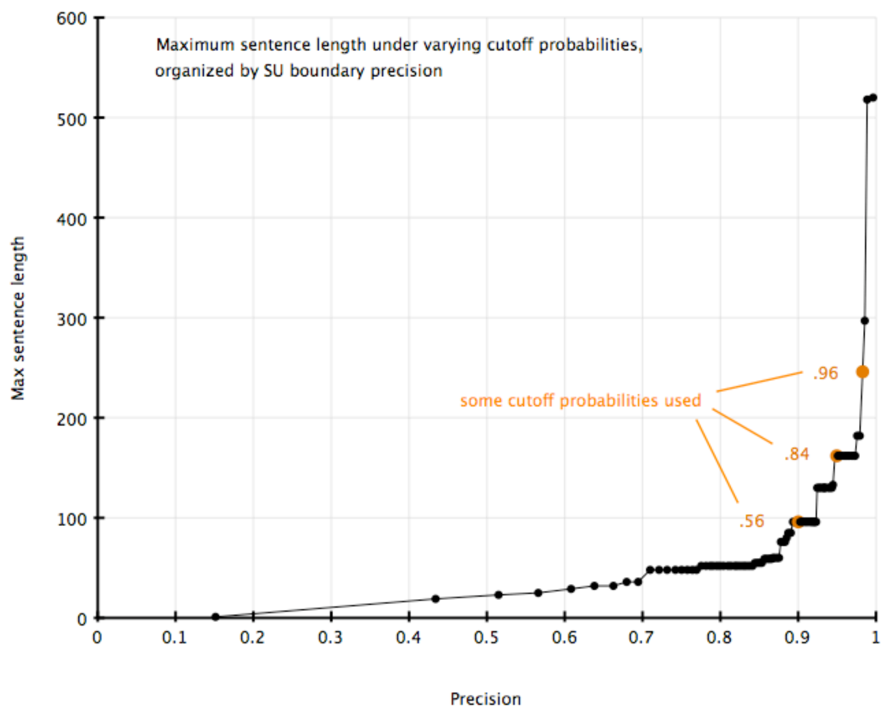

Figure 42: Maximum segment length versus precision of segmentation decisions

when sentences are shortened. This is probably because lower probability boundaries are forced to be chosen (in order to limit length) instead of higher probability boundaries that might fall in already-short segments. Figure 46 gives a sense of the average sentence length under some of these conditions. Clearly average sentence length is far shorter than the maximum length.

After high-precision boundary selection, we now need to have high-recall candidate generation between the fixed high-precision boundaries. The sub-sequences between fixed word boundaries we refer to as *fields*. The simplest procedure for candidate generation is to choose the k highest probability interior points as possible segmentation points. We want to get as high an oracle F-measure accuracy as possible, while keeping the number of candidates for any given field small. In examining different strategies, we now are dealing with three dimensions: the initial probability cutoff; the maximum allowed segment length; and now the number of internal segmentations made per field. The results of varying all three of these parameters (using the dev1 set) is displayed in figure 47. The x-axis shows k, the number of subdivisions per segment. The y-axis gives the oracle F-measure - the best we can do if we use just the subdivisions chosen and reject all of the incorrect ones. The thin lines show the results when using an initial probability cutoff of .99, while the bolder lines use .95. The latter seems significantly better at low numbers of subdivisions.

Two final graphs show that most words have either very high or very low posterior probabilities of being a segmentation point (figure 48), and that, for both reference transcripts and ASR transcripts, these extremes of posterior probability correspond to high precision decisions by the baseline model. This lends further evidence in support of our multi-stage approach to generating n-best candidates.

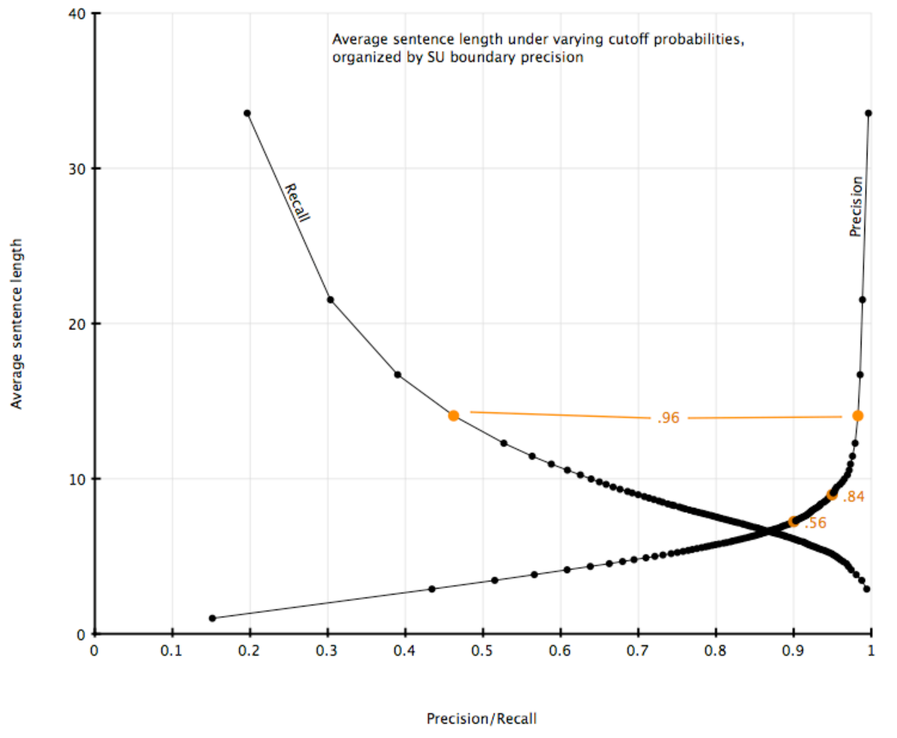

Figure 43: Average segment length versus precision and recall of segmentations

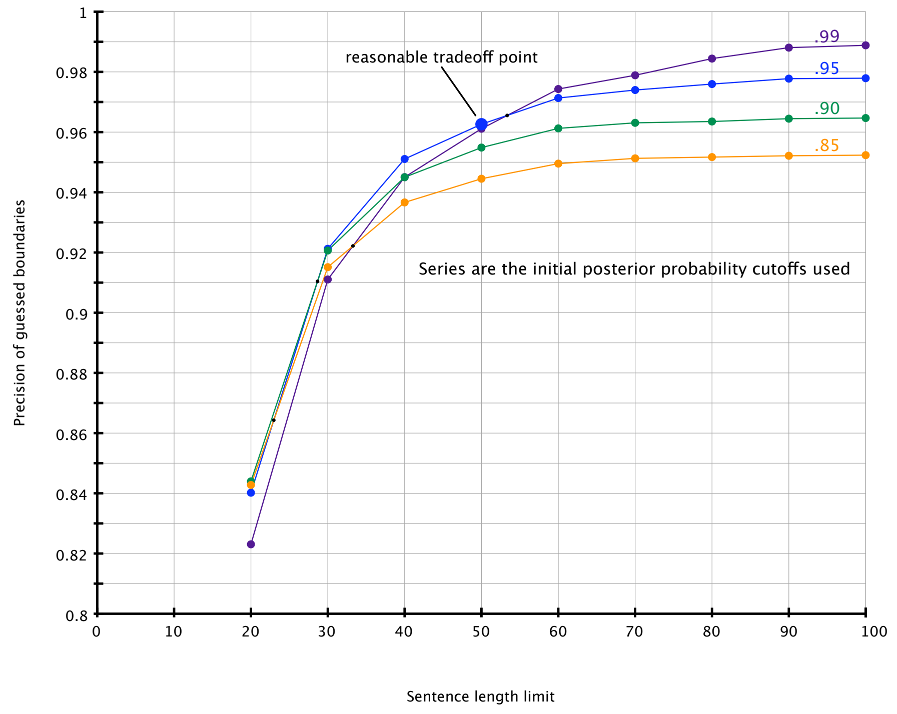

Figure 44: Two stage strategy with varying initial posterior probability cutoffs and maximum segment lengths

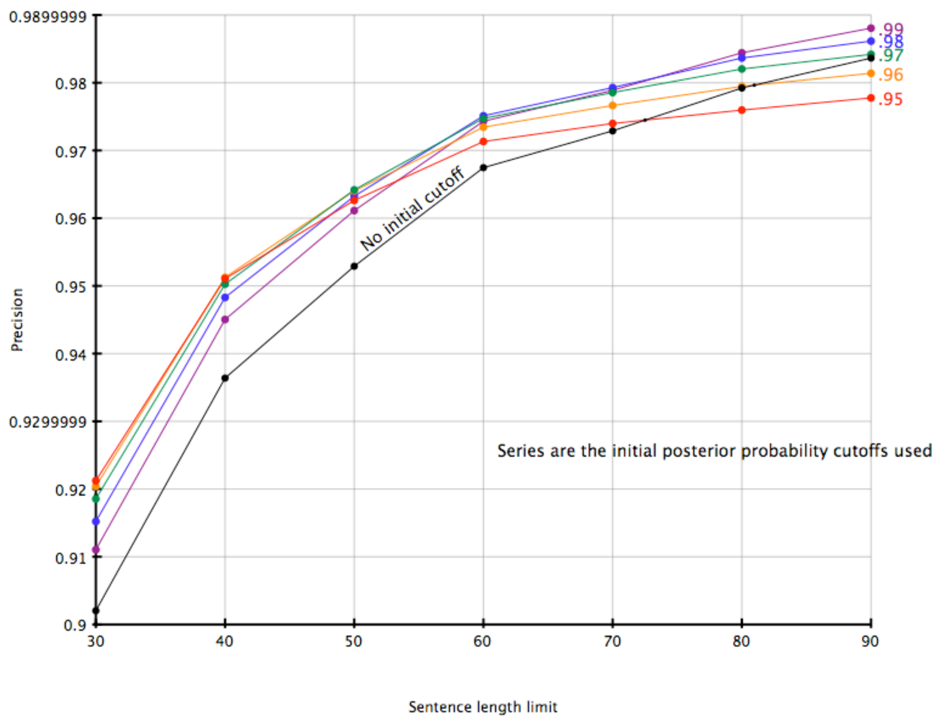

Figure 45: Two stage strategy with initial cutoffs of 0.95 and greater, including no initial cutoff

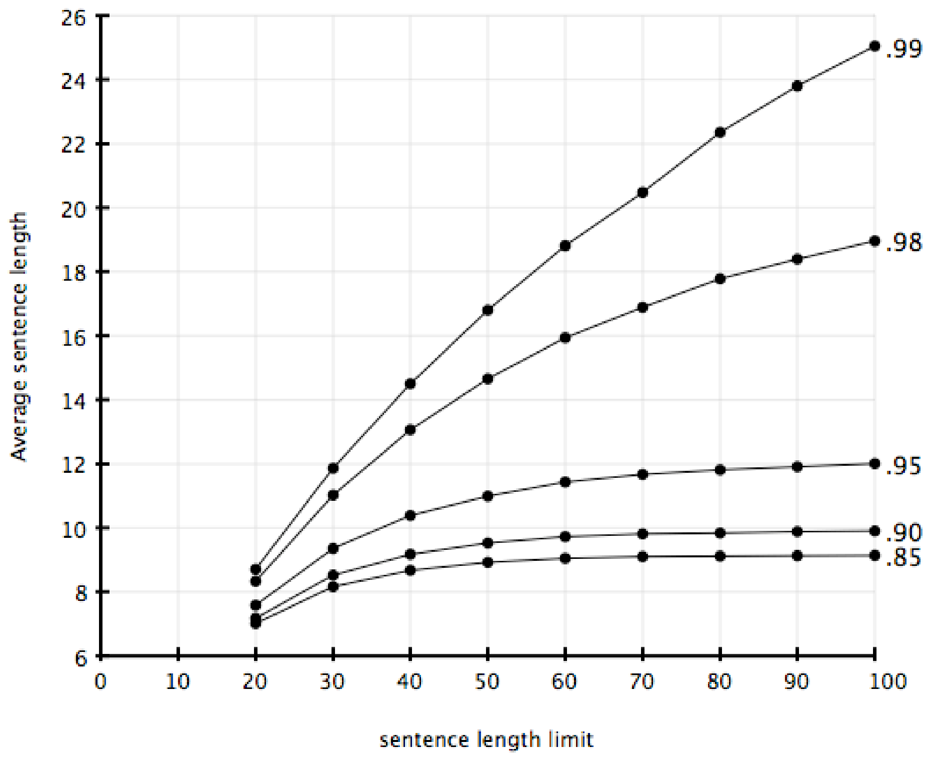

Figure 46: Average sentence length under various two-stage parameterizations

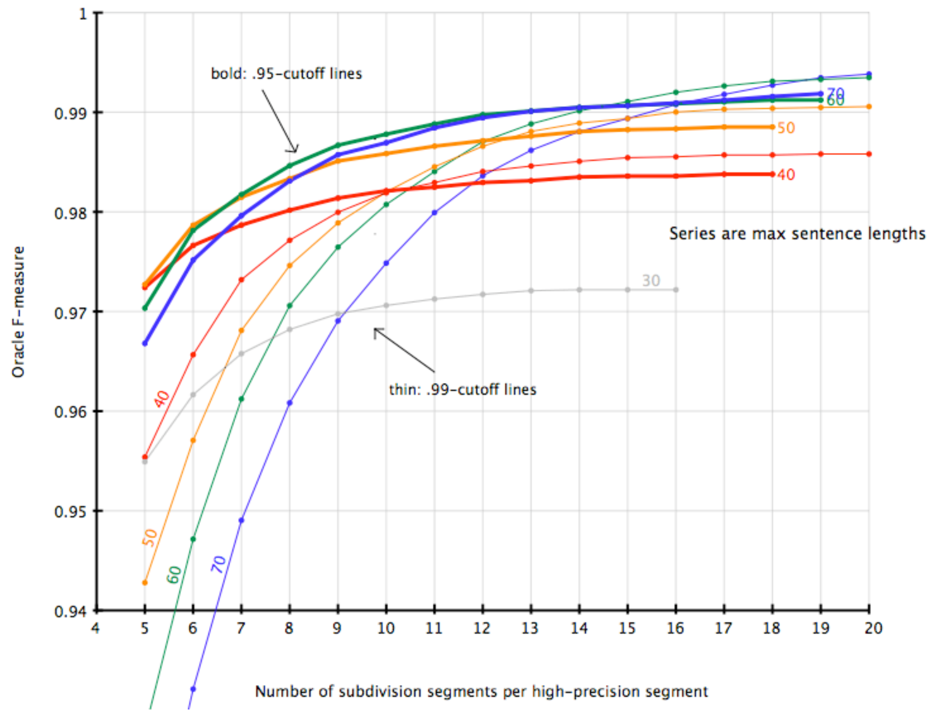

Figure 47: Oracle F-measure of n-best candidates under three parameters: initial probability cutoff, maximum sentence length, and number of field internal segmentations

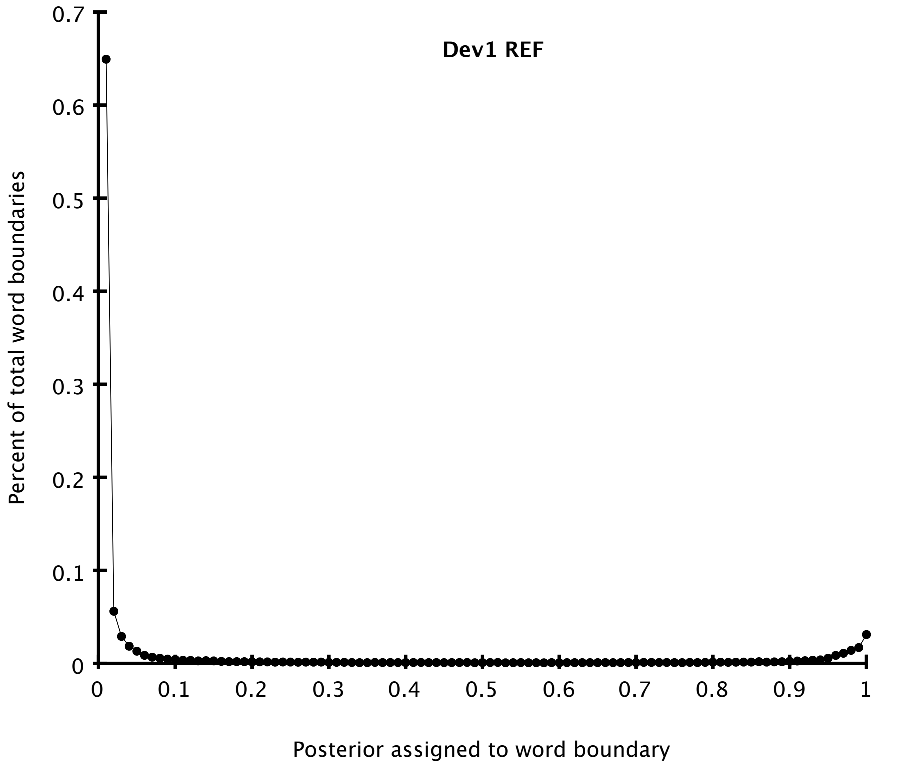

Figure 48: Number of word positions in various posterior probability bins

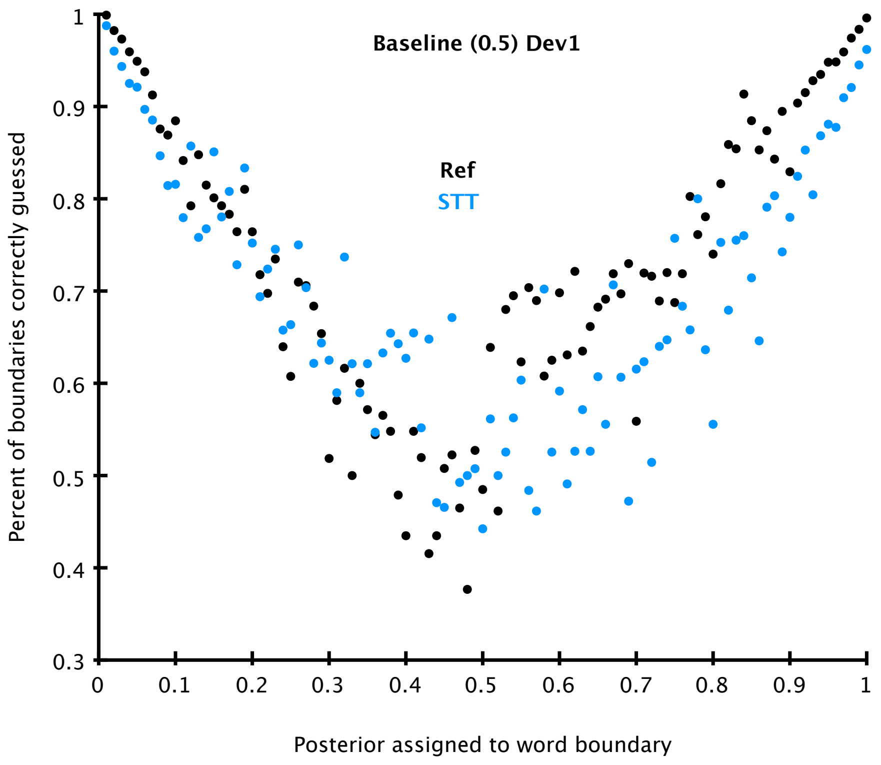

Figure 49: Percentage correct versus posterior probability

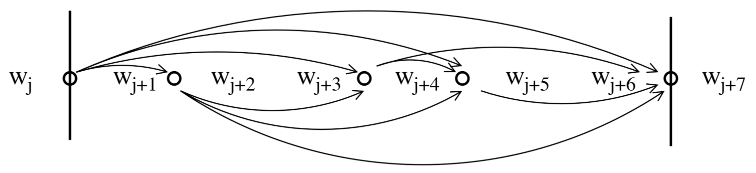

Figure 50: Picture of the n-best extraction approach

We thus used a two-stage candidate generation approach. In the first stage, we fixed a subset of the word boundaries as segmentation points, and the sub-sequences between fixed word boundaries we refer to as fields. To establish the fields, we first chose all word boundaries with a posterior probability $x>p$ for some parameter p. Since we intend to use parsers over candidate segments for feature extraction, it is important that the longest candidate segments be of tractable lengths, given polynomial complexity of parsing. Hence, we also parameterized a maximum field length k. For fields of length greater than k, the highest posterior internal word boundary was fixed as a segmentation point. This process is continued until no fields have length greater than k.

In the second stage, candidate segmentations are created for each field. This is also done using two parameters. The j highest posterior internal points are hypothesized as possible boundaries, although no internal point with posterior less than q is allowed to be hypothesized as a segmentation point. With j internal hypothesized points, there are 2^j candidate segmentations. Figure 50 presents a simple picture of how this candidate generation works, where three internal segmentation points are placed between the field boundaries, providing 8 candidate segmentations. For the trials presented in this report, we chose to generate lists for all sections under the parameterization $p = 0.95$, $k = 50$, $j = 10$, and $q = 0.05$. This yields lists with an F-measure oracle accuracy of 97.4 on the reference transcribed development set (dev2). Similar values were obtained for dev1 (97.7) and eval (97.2) using this approach.

6.2.2 Feature Extraction

Features are extracted from each candidate segmentation, for use in the reranker. We will assign letters to sets of features, which will allow us to present empirical results comparing performance on the development set using subsets of features.

a. Baseline model score: the baseline posterior score associated with the candidate segmentation. Recall that the candidates are defined by some j field-internal possible segmentation points. Each word boundary has a posterior probability x of being a segmentation point (calculated using the forward-backward algorithm), and $1-x$ of not being a segmentation point. The score of each candidate is the sum of the log posterior probabilities for its label at each possible boundary.

b. Candidate statistics: features derived from characteristics of the candidate. These included: the number of segments in the candidate; the maximum (and minimum) segment length within the candidate; and the average segment length. We also extracted segment-length n-grams (up to 4), where each segment is assigned a bin based on the number of words in the segment, and sequences of bins were extracted as features.

c. N-gram score: each candidate received a trigram score, using a model trained with segmentation points as tokens, on 1500 hours of Fisher data rapidly transcribed by WordWave.

d. Baseline model disfluency features: whether the baseline model labeled the word just before, just after, or both just before and just after a segment boundary as a speech repair.

e. ToBI-label features: annotation of a reduced set of ToBI-based labels described in Section 4. Automatic decision-tree based classifiers were built from a subset of Switchboard that was manually annotated with ToBI labels [OSSH$^+$01]. Certain labels occurred infrequently and were collapsed to obtain three types of break indices – fluent (low), disfluent (disf) and major phrase boundaries (high). The posterior probability of these three break indices was calculated from acoustic cues of prosody such as pitch, energy and duration associated with words, syllable, rhymes and phones (see [SFK$^+$05] for a detailed description of the features). Note, these acoustic cues are already present in our baseline system. In addition, however, we also explicitly combined a quantized version of posterior probabilities with the size/type of the largest syntactic constituent that begins/ends at a word, and the type/distance of the dependency. The constituents were taken from the Charniak parser output.

Other features were extracted from segments, regardless of the candidate within which they occur. Candidates then sum the feature values from the segments of which they are composed. The benefit of extracting them over segments rather than candidates is that, while the number of candidates grows exponentially with the number of internal points j, the number of unique segments only grows quadratically. Hence, with 10 internal points, there are 100 not 1000 parsing tasks.

f. Segment initial and final n-grams: for all segments in the candidate, unigrams and bigrams both sentence initial and final are extracted, as well as POS-tag unitags and bitags in the same positions.

g. Speaker change and backchannel: features derived from speaker change events from the two-sided conversations. The features were based on the distance of segment boundaries from both speaker changes and common backchannel words on both sides of the conversation.

We extracted features using three kinds of parsers: (1) the Charniak parser [Cha00, CJ05] was trained on the entire Switchboard Treebank with dev1 as the development set as in Section 3.2.2; (2) the Constraint-Dependency Grammar (CDG) parser [WH04]; and (3) the Minipar parser [Lin98]. Minipar has been described extensively in Section 5.2. Minipar dependency trees can be produced using a dependency-tree option that directly outputs a dependency format. A Constraint Dependency Grammar (CDG) parse for a sentence is represented by a set of dependencies between each word in the sentence and its head word (governor relations) and between each word and its required dependents (need relations, e.g., a verb may subcategorize for an object, and so the object would be required). Each word in the parse is also assigned a part of speech tag and values for a variety of lexical features (e.g., agr, case, verb type, voice, mood, gap, and inverted). For these experiments, we converted parse trees generated by Charniak's parser for each segment into CDG parses using the conversion procedure that was developed to generate CDG-parses for training both statistical CDG parsers [WH04] and language models [WH02, WSH04]. CDG parses differ from standard dependency parses. Although the CDG and dependency parses both incorporate labeled dependencies between words and their governors, a CDG parse also includes a variety of lexical features and dependencies between a word and its required dependents. We refer readers to the citations for additional details on these parsers and their representations.

h. Charniak parser features: each segment in the candidate is assigned features extracted from the Charniak parser output. These include a language model score by the Charniak parser for each segment. Non-terminal labels taken from the 1-best tree returned by the Charniak parser, e.g., S, NP, FRAG, etc., were extracted as features, along with an indication of whether they were at the root of the tree or not. In addition, a second feature combined each root or non-root non-terminal label with the number of children of the node.

i. CDG-derived dependency features: are generated from CDG parses of segments using the dependency feature template tool described below. Note that the features extracted involve not only governor relations (as in a standard dependency grammar), but also lexical features and need role relations, and so the representation is richer than the dependency features in (j).

j. Other dependency features: generated using the dependency feature template tool described below, based on Minipar and Charniak-derived dependency trees. To derive a dependency tree directly from Charniak parses, we used, const2dep, which uses standard head-percolation techniques and Hwa's head percolation table to define governor dependencies.

Features were extracted from Minipar, Charniak-derived, and CDG dependency trees using a feature extraction tool and an associated feature template language. The template language allows for features relating sentence position, lexical identity, part of speech tags, governor relationship, governor types, and other features generated by the particular parser. Some of the features used were:

- Whether word X was the root of the dependency tree,
- The number of children of the root of the sentence,
- The count or percentage of governor relationships of type X,
- The count of a parser generated feature X,
- The percentage of part-of-speech tags of type X,
- The part-of-speech Y and lexical features F of root word X,
- Whether the root was the first or last word in the sentence,
- The number of times word X with part-of-speech Y and governor relationship Z was in the sentence,
- The number of times that a word with part-of-speech X governed a word with part-of-speech Y,
- The need roles and lexical features of the last word in a segment.

6.2.3 SU Boundary Detection Results

The baseline MDE system was trained on roughly 400,000 words of MDE annotated Switchboard transcripts. Three additional sections of largely Fisher (with some Switchboard) data were RT'04F SimpleMDE annotated by the LDC: a 75,000 word development set (dev1); a second 35,000 word development set (dev2); and a 35,000 word evaluation set (eval). In addition, we made use of treebanks of these three sections (LDC2005E15). For all three of these sets, both reference transcripts and 1-best ASR[14] transcripts were available.

Table 29 shows results on the dev2 development set, under both reference and ASR 1-best transcript conditions. The two baseline results demonstrate that the combination of the HMM and MaxEnt models, from which we had access to forward-backward calculated posterior probabilities, perform competitively with the CRF model, from which we did not have access to the posteriors, due to the lack of such capability in the third-party toolkit being used [LSSH05]. Using the HMM+MaxEnt posteriors as feature **a** from our set, we then trained a reranker on dev1 using subsets of the features defined in the last section[15].

[14]The ASR system output came from [KMP$^+$04] and achieved WER of 11.7% on dev2 and 14.9% on eval.

[15]Note that the order in which features are introduced may obscure their contribution to the final full-feature system, which is difficult to tease apart.

	Reference		ASR	
System	Feats	NIST	Feats	NIST
Baseline CRF		25.9		35.8
Baseline HMM+MaxEnt	a	26.1	a	35.1
Reranked	a-e	25.9	a,c-e	34.6
Reranked	a-h	24.5	a,c-h	33.8
Reranked	a-i	23.2	a,c-i	**33.5**
Reranked	a-j	**23.0**	a,c-j	33.9

Table 29: NIST error results with different feature sets on dev2, both reference and ASR 1-best transcripts.

	Reference		ASR	
System	Feats	NIST	Feats	NIST
Baseline CRF		26.5		37.2
Baseline HMM+MaxEnt	a	27.0	a	36.7
Reranked, dev1 training	a-j	24.9	a,c-i	36.4
Reranked, dev1+2 training	a-j	24.4	a,c-i	35.6

Table 30: NIST error results of baseline and reranker trained on dev1 and on dev1+dev2 applied to the eval set, both reference and ASR 1-best transcripts

As can be seen from the table, feature sets **a-e** provide small improvements over the baseline for the reference condition, though more for the ASR condition. This difference is due to the ToBI-based features, which were generally found to be more useful in the ASR condition, when transcript-based features were less reliable. Feature set **b** was not found to be useful in the ASR condition, and hence was omitted. Adding features **f-h** provided large improvements under both conditions. Adding the CDG derived features (**i**) provided larger improvements in the reference condition than with ASR transcripts. Finally, adding other dependency features (**j**) provided small additional improvements in the reference condition, but no gain with ASR.

The best performing reranker models for the reference and ASR conditions were applied to the eval set, using either dev1 as training, or dev1 and dev2 together as training. The results are shown in Table 30. This resulted in gains of 2.6 percent (absolute) for the reference condition, which is statistically significant at $p < 0.00001$; and 1.1 percent (absolute) for the ASR condition, which is significant at $p < 0.02$, using a matched pair test on segments defined by long pauses [HO04].

Note that these gains are not simply due to the use of extra training data in dev1 and dev2. Adding these sets to the training data for the baseline model yielded just 0.7 percent NIST error rate reduction on the eval set for the reference condition, and 0.1 percent for the ASR condition.

6.3 Optimizing for Parse Accuracy

In a final, suggestive set of experiments, we looked at reranking with an objective other than SU boundary detection accuracy. While it is extremely useful to have human annotations of sentence-like units, according to some agreed upon labeling guidelines, the real utility of these segmentations is to allow for effective and accurate downstream processing, rather than any intrinsic utility of these segmentations in their own right. Both efficiency and accuracy of downstream processing may benefit from choices other than the human-annotated standard. To the extent that the reranking can be made to optimize on these downstream objectives, further improvements might be had on those measures.

To demonstrate this potential, we assigned each candidate segmentation a parse accuracy score (in terms of head-dependency score), using the SParseval tools. We then performed reranking to optimize on parse accuracy rather than SU-boundary detection accuracy. To do this is quite straightforward, once the parse accuracy scores are obtained. Consider the hypothetical table of seven candidates shown in Table 31.

Candidate	SU accuracy	Parsing accuracy	Best SU	Best Parsing
1	0.8	0.7		
2	0.9	0.7	←	
3	0.8	0.8		←
4	0.9	0.8	←	←
5	0.6	0.7		
6	0.7	0.6		
7	0.5	0.5		

Table 31: Different outcomes based on optimizing for SU versus parsing accuracy.

As mentioned earlier, the MaxEnt reranker puts as much of the conditional probability mass as possible on the set of most accurate candidates. We can change the objective being optimized by simply changing which candidates are chosen as "most accurate". In the above example, when optimizing for SU boundary detection, candidates 2 and 4 have the highest accuracy; however, to optimize for parsing accuracy, candidates 3 and 4 are the best. In such a way, exactly the same feature set and training approach can be made to optimize for parse accuracy instead of SU boundary detection accuracy.

Table 32 shows the results of parse accuracy optimization versus SU-boundary detection accuracy on the dev2 set, using most (but not all) of the features used in the results from the previous section[16]. From these results, we can see several things. First, our standard reranking approach, optimizing for SU-boundary detection, yields large improvements in both bracketing and head-dependency F-measure parsing accuracy for both reference (REF) and ASR transcript conditions. Optimizing specifically for head-dependency accuracy yields additional improvements in parsing accuracy, at the expense of SU accuracy. In both conditions, the NIST error returns to roughly the same level as the baseline, although the F-measure SU-boundary detection accuracy remains better than the baseline. Note that in both conditions, recall of SU-boundaries improves while precision decreases, indicating that the parser does a better job when more SU boundaries are inserted.

While these results are suggestive and encouraging, recall that they were run before many of the optimizations on feature set and regularizer were carried out. Further empirical research is required to find the best practices for this parse-accuracy focused optimization.

In summary, we have presented an approach for recasting SU detection as an n-best reranking problem with a relatively small n. Using reranking techniques with these n-best lists, we have demonstrated significant improvements over very strong baselines. We have also presented preliminary results showing that this approach can be straightforwardly made to optimize for downstream objectives other than SU boundary detection accuracy, such as parsing accuracy.

7 Summary and Future Directions

This report has described a series of experiments conducted by the PaSSED team. We have investigated the use of families of metrics implemented in the SParseval tool, in particular, bracket-based and dependency-based scores. We evaluated the sensitivity of these metrics to word errors common when processing speech

[16]These results are intended to take a particular feature set and optimize in two different ways. We have yet to run trials that correspond exactly to the feature set breakdowns presented in Table 29 or on the eval set. In addition, regularizer values were not optimized for these trials, hence they are not meant to be directly compared with previously reported trials. Nevertheless, they give an indication of the impact of this optimization on relatively rich feature sets for both reference and ASR conditions.

						Parsing performance	
		SU performance				Bracketing	H-Dep
System	Optimized for	P	R	F	NIST	F-measure	F-measure
Baseline REF		87.2	82.7	84.9	26.1	74.0	77.3
Reranked REF	SU	86.9	86.7	86.8	23.3	76.3	78.7
Reranked REF	Parse	83.8	87.9	85.8	25.9	76.9	79.1
Baseline ASR		83.3	77.7	80.4	35.1	63.9	65.8
Reranked ASR	SU	84.2	78.7	81.3	33.7	64.8	66.4
Reranked ASR	Parse	80.8	81.6	81.2	35.3	65.7	66.8

Table 32: Results on dev2 optimizing for parse accuracy versus optimizing for SU boundary detection accuracy, for both reference (REF) and ASR conditions

transcripts and to errors in sentence boundaries and edit regions detected using automatic methods across three statistical parsers. In general, the bracket scores are more sensitive to segmentation errors and less sensitive to word errors than the dependency metrics. The dependency scores were calculated based on three alternative head percolation tables; for the statistical parsers, Charniak's head rules resulted in larger dependency F-measure scores than the other two tables investigated. Hwa's table, however, provided the highest scores for the Minipar parser.

We have investigated alternative ways of incorporating prosodic information into the baseline system and into parsing. Section 4.2 described how a classifier can be built to detect ToBI prosodic events at the word-level with high accuracy. Three applications for utilizing the predictions of the classifier were investigated. The first tackled the problem of disfluency detection, wherein prosodic predictions were incorporated into a parsing framework via two simple generative models. The second involved the impact of prosodic predictions on the baseline sentence boundary detection task. The third involved using the ToBI features together with syntactic features to capture potentially useful interactions between the two structures for sentence boundary detection.

Three methods of handling edits were evaluated in this research. The first involved detecting repairs with a repair model, removing them, parsing the remaining words with the parser, reinserting each span of detected repair words under a flat EDITED constituent, and then relaxing scoring to ignore internal structure, contiguous boundaries, and attachment of EDITED constituents. Such a separation of the parsing and repair detection models is both simple and effective since each can be attacked independently, and it becomes trivial to measure how improvements in repair detection impact resulting parse accuracy (e.g., [CJ01, KLC+05]). The second approach was to enrich an SU transcript with markups for edited regions so that the parser can more effectively handle them; this approach was effectively used by the Minipar parser. The third method was to enrich the grammar so that the parser can detect and model edit structures more effectively.

When investigating the impact of perfect versus system knowledge of SUs and edits, we found that errors in SU detection cause a greater degradation in parsing accuracy than errors in edits do; hence, we carried out a series of reranking studies to improve SU detection accuracy. These studies show that by using a wide variety of knowledge sources, we can improve significantly upon an already strong baseline by using a reranking approach using either human or ASR transcripts. Additionally, we found that if we optimize for SU accuracy that parse accuracies improve, but if we optimize for parse accuracy, that they improve even more, but at the cost of SU accuracy. Parsers, in general, parse shorter sentences more accurately than longer sentences. Further empirical research is required to find the best practices for the parse-accuracy focused optimization.

Future research directions of our team include:

- **Cross-linguistic Study of MDE and Parsing:** Little work has been carried out to date on evaluating the relationship between metadata detection and parse accuracy cross-linguistically. One significant reason for this is the limited availability of data annotated with both structural metadata and with parse structure. A portion of the Levantine CallHome corpus of transcribed conversational speech was recently treebanked (LDC2005E78) for use in the *Parsing Arabic Dialects* (PAD) project at the Hopkins workshop. Due to the complexity of their project, the PAD group used the reference SU boundaries and filtered all disfluencies from their test data to simplify data processing. They reported a 21% improvement in parsing F-measure[17] from using the edit handling methods of Charniak and Johnson [CJ01]. This provides a preliminary piece of cross-linguistic evidence that standard parsing techniques in use today are less effective in the presence of disfluency, or said another way, do not model disfluent speech as well as fluent speech. This result in Levantine is of further interest because LDC recently released a small portion of the Levantine CallHome transcripts with SimpleMDE annotations, LDC2004E47 (a pilot release), so we now have an opportunity to investigate how SimpleMDE and tree annotations interact in a language other than English. While a similar pilot SimpleMDE corpus of annotated Mandarin CallHome transcripts has also been released (same catalog ID), there is no treebank for the Mandarin CallHome transcripts. Nevertheless, current trends in languages of interest to funding agencies suggest such data may become available in the near future. In all such cases, it would be interesting to begin applying many of the techniques developed in previous literature and at this workshop to these other languages. This work may highlight issues in the metadata-parsing interaction not yet encountered in English and suggest new avenues for research in these related fields.

- **SParseval Refinements:** For the purposes of the workshop, we implemented the open class dependencies by specifying the list of word tags corresponding to closed class words (i.e., IN, TO, RP, and POS). A potentially better approach is to specify the closed class words directly. Another issue arises with respect to the importance of getting the words correct in a dependency to match the corresponding gold dependency. In future, we will also explore other options for matching dependencies (e.g., matching stem or POS). Since the SParseval package is reconfigurable, we can explore the use of additional head percolation rules and alignment procedures. For other speech processing tasks like broadcast news, conversation sides may not be an appropriate chunk for SParseval, and so we will investigate other chunks for scoring parses of speech.

- **Prosodic Analysis and Integration with Parsing:** The classifiers for prosodic symbols were trained on a manually transcribed corpus, annotated by a well-trained linguist. Thus, it is expensive to generate such corpora for new languages. Clearly, there are benefits in studying how prosodic symbols could be learned in an unsupervised manner. A joint model with syntax could provide valuable clues to the unsupervised algorithm.

  To constrain the scope of the workshop, boundary tones were not studied extensively in this workshop. Preliminary classification results, show that the manual annotations need to be refined. This can be done in two ways — collapsing fine categories into coarse categories, and by verifying/correcting the annotations with stylized pitch trajectories that are computed from the acoustics. Improved boundary tones can then provide good cues for syntax-phonology interface (e.g., [WG04]).

  Preliminary experiments on a PCFG enhanced with prosodic symbols, reported in Section 4.3, indicate that the performance of the parser depends significantly on the precision-recall trade-off of the

[17] Details of this experiment can be found in the PAD final report.

prosodic symbols. This suggests a tighter integration of prosody in a PCFG through soft decisions, instead of making a hard-decision before tagging the words. Such an approach, would also allow hypothesized syntactic features to be used in predicting prosodic symbols and thus should improve their performance as well.

- **Filler Recognition and Parsing:** In Section 3.3, we described issues in evaluating the interaction between filler recognition and parsing. Two related challenges highlighted there were the lack of: (1) an existing method regarding how filler recognition should be integrated with a parsing model, and (2) an approach for scoring parses containing fillers. In future research, we will address these issues by applying the approach to fillers that is similar to what was done with edit regions in [CJ01]. If like speech repairs, we assume that fillers do not contribute significantly towards the meaning of an utterance, then it would make sense for fillers to be handled similarly to repairs. A specialized module for detecting filler regions can be integrated with the parsing model in the same way as for repairs, and we can in evaluation ignore the internal structure and attachment of filler regions. Of course, the obvious difference between repairs and fillers is their syntactic representation: repairs are represented by a distinct EDITED category; whereas, fillers are conflated with other phenomena and split across INTJ and PRN labels.

  We propose to extend the Charniak and Johnson [CJ01] representational model of repairs to encapsulate both fillers and repairs. We assume that first the SimpleMDE and tree annotations will have been made consistent, per the discussion in Section 3.3. For each set of contiguous filler words, we will bracket them with a single, flat FILLER constituent, and SPARSEVAL will ignore internal structure, contiguous boundaries, and attachment of EDITED and FILLER constituents. The distinction between adjacent EDITED and FILLER will be evaluated, but FILLER constituents within EDITED constituents will be discarded along with other internal EDITED structure. It is worth discussing here why we do not propose collapsing EDITED and FILLER into a single DISFLUENT (or perhaps NON-SEMANTIC) category. While we have argued that neither repairs nor fillers contain significant semantic content, they do differ from one another both distributionally and structurally (e.g., repairs have crossing dependencies and fillers do not). Besides suggesting that distinct cognitive phenomena are at work in these phenomena, such differences will also likely result in different detection accuracies for the two cases. Consequently, we would like to be able to separately measure the impact of each form of disfluency on parse accuracy, and representing this distinction syntactically makes it relatively easy to perform such evaluation. Finally, in terms of downstream tasks, it is certainly conceivable that applications besides Rich Transcription might also find it useful to differentiate repairs from fillers, and such differentiation ought to be possible from the syntax. To summarize, the proposed representation offers the following benefits:

  – Fillers can be distinguished from non-fillers using syntax.
  – The impact of fillers on parse accuracy can be easily measured.
  – Alternative models of filler detection can be easily plugged in.
  – Parser evaluation will focus on the more meaningful words and avoid the possibility of hill-climbing parsing techniques on filler and edit words.

- **Using MDE to Improve Parsing Accuracy:** While the current rules used by the MINI-BJD tool discussed in Section 5.2 were handcrafted by an expert, automatically induced rules using Transformation Based Learning [Bri95] could also be learned. Different rule sets could be learned for each set of head percolation rules, so as to attempt to maximize the head dependency F-measure score for each of set of head percolation rules.

The improvement in EDIT-finding performance brought about by the grammar enrichments considered in Section 5.3 appear to be synergistic: both -UNF propagation and daughter annotation help deal with the unique aspects of speech and they help more in combination. This motivates a general program of future work investigating why they do work and to what degree they are orthogonal with other features. One feature is the ToBI ‘p’ label discussed in Section 4 and hypothesized to indicate upcoming speech repairs [Lic96]. Other factors such as string similarity between the repair and reparandum [JC04], prosodic boundaries [WG04], and grammatical relations [CH05] may bear a similarly synergistic relation to the features described in this report. Each of these other dimensions of linguistic structure presents an independent direction for future work.

Also, the input could be further enriched beyond EDIT and FILLER markups, e.g., using prosodic information (see Section 4). In many cases, punctuation can be used to separate various constituents for the parser. While punctuation is present when using reference words, it is absent in the case of text recognized by an automatic speech recognition system. Automatic punctuation insertion, in addition to metadata markup, could prove useful for parsing these texts, warranting further investigation.

- **Reranking Experiments:** There are a variety of future directions we could take on the reranking research reported in Section 6. The first involves evaluating a wider variety of features for SU reranking. Not all of the features that were investigated in the workshop have been integrated into our experiments (e.g., center parse scores [Mai05]). Further optimization experiments for parse accuracy are also of called for. For example, we used dependency scores for our reranking studies involving parse score optimization; however, bracket scores are much more sensitive to SU error and so could provide a different pattern of results. Additionally, we would like to investigate reranking SU boundaries for other downstream tasks, such as machine translation.

- **Off-topic Analysis:** The data sets we used for sentence boundary detection consist of telephone conversations between strangers who are assigned a topic to discuss. Despite this assignment, there are many sections of conversation where the participants get completely off-topic (for instance, talking about the weather). The ability to automatically detect such sections could prove useful for downstream language processing tasks such as information retrieval or topic classification, in the same way that edit regions and filler words can be removed from sentences to improve parsing. We propose to build a system that can detect two types of off-topic regions: “small talk” (conversation not at all related to the topic) and “metaconversation” (conversation about the conversation). Using human-labeled conversations as training data, we will initially apply off-the-shelf machine learning algorithms to the task of classifying these regions. We will also investigate the utility of a variety of features (such as words, laughter, interruptions, and pause duration) in improving the system and evaluate the extent to which off-topic speech can be detected regardless of the assigned topic. For the task of automatically splitting conversations into on-topic and off-topic regions, we will apply various supervised and unsupervised text segmentation algorithms and ultimately propose a joint system for segmentation and classification. Such a system should be generalizable to other corpora, including meetings, broadcast news, and lectures. In the longer term, we plan to use the knowledge of off-topic regions to improve high-level language processing tasks.

A Sparseval Data Tables

This section contains a set of tables summarizing the parse scores obtained by Bikel’s, Charniak’s, and Roark’s parsers under a variety of data quality and metadata use conditions from Section 3.2.3. These scores represent composite scores over all 72 conversation sides in the dev2 set. The results for the Bikel parser appear in Tables 33, 34, 35, and 36. The Charniak parser results appear in Tables 37, 38, 39, 40.

Results from Roark's parser appear in Tables 41, 42, 43, and 44. Also included are the tables for the focus studies in Section 3.2.4. The impact of ASR transcript quality on parses produced by Charniak's parser using system metadata appears in Table 45. The effect of metadata quality using Charniak's parser to parse the dev2 reference transcripts with three metadata qualities (i.e., reference, prosody model only, and the baseline MDE model) appears in Table 46. The impact of SU and Edit metadata quality using Charniak's parser to parse the dev2 reference transcripts with four metadata conditions (i.e., reference SU and reference Edit metadata, reference SU and system Edit metadata, system SU and reference Edit metadata, system SU and system Edit metadata) appears in Table 47.

Bikel Parser: Reference Transcript, Reference Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	88.13	87.58	88.68	81.69	80.22	83.22
		Head Dep	Charniak	85.99	86.13	85.84	80.78	80.22	81.34
			Collins	85.50	85.65	85.36	80.37	79.82	80.93
			Hwa	84.92	85.06	84.78	79.71	79.16	80.26
		Open Class Dep	Charniak	86.59	86.42	86.76	81.51	80.73	82.30
			Collins	86.05	85.94	86.17	81.04	80.30	81.78
			Hwa	85.39	85.25	85.53	80.32	79.54	81.11
	Not Aligned	Head Dep	Charniak	86.03	86.18	85.89	81.18	80.63	81.75
			Collins	85.56	85.71	85.42	80.76	80.21	81.33
			Hwa	84.97	85.11	84.82	80.05	79.50	80.61
		Open Class Dep	Charniak	86.63	86.47	86.80	81.83	81.06	82.63
			Collins	86.10	85.99	86.22	81.36	80.62	82.11
			Hwa	85.43	85.29	85.57	80.63	79.85	81.43
Not Labeled	Aligned	Bracket	N/A	89.90	89.34	90.47	84.05	82.54	85.62
		Head Dep	Charniak	90.55	90.70	90.40	85.10	84.51	85.69
			Collins	89.74	89.89	89.58	84.35	83.77	84.94
			Hwa	88.80	88.95	88.65	83.41	82.84	83.99
		Open Class Dep	Charniak	91.05	90.88	91.23	85.77	84.95	86.60
			Collins	90.00	89.88	90.12	84.79	84.02	85.57
			Hwa	89.08	88.94	89.22	83.85	83.05	84.68
	Not Aligned	Head Dep	Charniak	90.57	90.72	90.42	85.48	84.89	86.08
			Collins	89.79	89.94	89.64	84.73	84.15	85.32
			Hwa	88.83	88.98	88.68	83.73	83.15	84.31
		Open Class Dep	Charniak	91.07	90.89	91.25	86.07	85.25	86.91
			Collins	90.04	89.92	90.16	85.11	84.34	85.89
			Hwa	89.11	88.96	89.25	84.15	83.34	84.98

Table 33: Bikel parser results on reference transcripts with reference metadata.

Bikel Parser: Reference Transcript, System Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	75.49	76.67	74.35	72.75	70.42	75.24
		Head Dep	Charniak	77.88	79.63	76.21	74.98	73.19	76.86
			Collins	77.59	79.33	75.93	74.75	72.96	76.62
			Hwa	76.77	78.50	75.13	73.91	72.15	75.76
		Open Class Dep	Charniak	78.10	79.40	76.84	75.53	73.43	77.76
			Collins	77.76	79.07	76.49	75.23	73.15	77.43
			Hwa	76.96	78.25	75.71	74.35	72.30	76.53
	Not Aligned	Head Dep	Charniak	78.09	79.84	76.41	75.37	73.57	77.26
			Collins	77.82	79.57	76.15	75.12	73.33	77.00
			Hwa	76.97	78.70	75.32	74.23	72.46	76.09
		Open Class Dep	Charniak	78.28	79.59	77.02	75.83	73.72	78.06
			Collins	77.97	79.28	76.70	75.51	73.42	77.72
			Hwa	77.17	78.47	75.91	74.65	72.58	76.83
Not Labeled	Aligned	Bracket	N/A	77.48	78.69	76.30	75.11	72.70	77.68
		Head Dep	Charniak	82.26	84.11	80.50	78.94	77.06	80.92
			Collins	81.60	83.43	79.85	78.37	76.50	80.34
			Hwa	80.49	82.29	78.76	77.30	75.46	79.24
		Open Class Dep	Charniak	82.30	83.68	80.98	79.39	77.18	81.73
			Collins	81.41	82.78	80.08	78.60	76.43	80.90
			Hwa	80.44	81.80	79.13	77.58	75.44	79.85
	Not Aligned	Head Dep	Charniak	82.43	84.28	80.67	79.32	77.43	81.31
			Collins	81.80	83.63	80.04	78.74	76.86	80.72
			Hwa	80.66	82.47	78.93	77.62	75.77	79.57
		Open Class Dep	Charniak	82.46	83.83	81.12	79.67	77.45	82.02
			Collins	81.60	82.97	80.26	78.88	76.70	81.18
			Hwa	80.63	81.99	79.32	77.87	75.72	80.15

Table 34: Bikel parser results on reference transcripts with system metadata.

Bikel Parser: ASR Transcript, Reference Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	75.96	75.07	76.86	70.82	68.97	72.76
		Head Dep	Charniak	72.39	71.74	73.04	68.34	67.07	69.66
			Collins	71.33	70.69	71.98	67.31	66.06	68.61
			Hwa	71.40	70.76	72.04	67.31	66.06	68.61
		Open Class Dep	Charniak	72.73	71.75	73.74	68.90	67.29	70.58
			Collins	71.47	70.49	72.49	67.60	66.04	69.23
			Hwa	71.68	70.71	72.67	67.75	66.19	69.38
	Not Aligned	Head Dep	Charniak	72.63	71.98	73.29	69.00	67.71	70.33
			Collins	71.59	70.95	72.24	67.95	66.68	69.26
			Hwa	71.66	71.02	72.31	67.96	66.69	69.27
		Open Class Dep	Charniak	73.00	72.01	74.01	69.54	67.92	71.24
			Collins	71.76	70.77	72.77	68.21	66.64	69.85
			Hwa	71.97	71.00	72.97	68.40	66.82	70.05
Not Labeled	Aligned	Bracket	N/A	79.11	78.19	80.05	74.47	72.53	76.51
		Head Dep	Charniak	76.47	75.79	77.16	72.18	70.84	73.58
			Collins	75.11	74.44	75.79	70.84	69.52	72.21
			Hwa	75.13	74.46	75.81	70.82	69.50	72.19
		Open Class Dep	Charniak	76.72	75.68	77.78	72.65	70.96	74.42
			Collins	74.98	73.95	76.04	70.89	69.25	72.60
			Hwa	75.24	74.22	76.28	71.11	69.48	72.83
	Not Aligned	Head Dep	Charniak	76.68	76.00	77.37	72.82	71.46	74.22
			Collins	75.35	74.68	76.04	71.47	70.14	72.85
			Hwa	75.38	74.70	76.06	71.44	70.11	72.82
		Open Class Dep	Charniak	76.95	75.91	78.02	73.26	71.56	75.04
			Collins	75.25	74.22	76.32	71.50	69.85	73.22
			Hwa	75.52	74.50	76.56	71.74	70.09	73.47

Table 35: Bikel parser results on ASR transcripts with reference metadata.

Bikel Parser: ASR Transcript, System Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	64.50	65.11	63.91	63.01	60.65	65.55
		Head Dep	Charniak	66.07	67.23	64.95	64.26	62.32	66.33
			Collins	65.10	66.25	64.00	63.30	61.38	65.34
			Hwa	64.94	66.09	63.84	63.13	61.22	65.16
		Open Class Dep	Charniak	66.23	66.83	65.63	64.69	62.29	67.27
			Collins	65.07	65.63	64.53	63.46	61.08	66.03
			Hwa	65.07	65.66	64.50	63.44	61.09	65.97
	Not Aligned	Head Dep	Charniak	66.59	67.76	65.45	64.87	62.91	66.96
			Collins	65.64	66.79	64.52	63.88	61.94	65.93
			Hwa	65.44	66.59	64.33	63.73	61.80	65.78
		Open Class Dep	Charniak	66.76	67.36	66.16	65.28	62.86	67.88
			Collins	65.60	66.16	65.06	64.01	61.61	66.60
			Hwa	65.60	66.19	65.02	64.07	61.70	66.62
Not Labeled	Aligned	Bracket	N/A	67.74	68.38	67.12	66.29	63.82	68.97
		Head Dep	Charniak	69.80	71.03	68.62	67.76	65.71	69.94
			Collins	68.58	69.79	67.42	66.55	64.54	68.70
			Hwa	68.37	69.57	67.21	66.26	64.26	68.40
		Open Class Dep	Charniak	69.76	70.40	69.14	68.04	65.52	70.75
			Collins	68.26	68.84	67.69	66.46	63.97	69.15
			Hwa	68.28	68.90	67.68	66.42	63.96	69.07
	Not Aligned	Head Dep	Charniak	70.26	71.49	69.07	68.35	66.28	70.55
			Collins	69.08	70.30	67.91	67.13	65.10	69.29
			Hwa	68.84	70.05	67.67	66.85	64.83	69.01
		Open Class Dep	Charniak	70.24	70.88	69.61	68.60	66.06	71.34
			Collins	68.76	69.35	68.19	67.00	64.49	69.72
			Hwa	68.77	69.39	68.17	67.03	64.56	69.71

Table 36: Bikel parser results on ASR transcripts with system metadata.

Charniak Parser: Reference Transcript, Reference Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	88.07	87.64	88.51	81.78	82.69	80.89
		Head Dep	Charniak	85.68	85.82	85.53	80.77	82.59	79.03
			Collins	85.51	85.66	85.37	80.28	82.09	78.55
			Hwa	84.80	84.94	84.66	79.83	81.63	78.11
		Open Class Dep	Charniak	86.12	85.79	86.45	81.39	82.58	80.22
			Collins	86.05	85.75	86.36	80.91	82.12	79.74
			Hwa	85.16	84.87	85.45	80.35	81.55	79.18
	Not Aligned	Head Dep	Charniak	85.72	85.87	85.58	81.25	83.08	79.50
			Collins	85.57	85.71	85.43	80.73	82.55	78.99
			Hwa	84.84	84.98	84.70	80.24	82.05	78.51
		Open Class Dep	Charniak	86.15	85.83	86.48	81.75	82.95	80.58
			Collins	86.08	85.78	86.39	81.24	82.45	80.07
			Hwa	85.19	84.90	85.48	80.69	81.90	79.51
Not Labeled	Aligned	Bracket	N/A	90.29	89.85	90.74	84.57	85.52	83.65
		Head Dep	Charniak	90.35	90.50	90.20	85.37	87.29	83.53
			Collins	90.13	90.28	89.97	84.78	86.69	82.95
			Hwa	89.05	89.20	88.90	84.10	85.99	82.28
		Open Class Dep	Charniak	90.76	90.42	91.11	85.94	87.20	84.71
			Collins	90.41	90.09	90.73	85.15	86.42	83.92
			Hwa	89.25	88.94	89.55	84.43	85.70	83.20
	Not Aligned	Head Dep	Charniak	90.39	90.54	90.24	85.81	87.75	83.96
			Collins	90.18	90.33	90.02	85.22	87.14	83.39
			Hwa	89.08	89.23	88.93	84.49	86.39	82.66
		Open Class Dep	Charniak	90.79	90.45	91.14	86.26	87.53	85.03
			Collins	90.44	90.12	90.76	85.47	86.74	84.23
			Hwa	89.26	88.96	89.56	84.75	86.02	83.52

Table 37: Charniak parser results on reference transcripts with reference metadata.

Charniak Parser: Reference Transcript, System Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	74.34	75.56	73.16	72.72	73.44	72.01
		Head Dep	Charniak	77.95	79.70	76.28	76.46	77.95	75.03
			Collins	77.64	79.38	75.98	76.02	77.49	74.60
			Hwa	76.88	78.60	75.23	75.41	76.87	74.00
		Open Class Dep	Charniak	78.03	79.14	76.94	76.77	77.50	76.06
			Collins	77.69	78.82	76.60	76.30	77.03	75.59
			Hwa	76.90	78.02	75.81	75.61	76.36	74.88
	Not Aligned	Head Dep	Charniak	78.17	79.93	76.50	76.89	78.38	75.45
			Collins	77.86	79.61	76.19	76.41	77.90	74.98
			Hwa	77.09	78.82	75.43	75.77	77.24	74.35
		Open Class Dep	Charniak	78.24	79.36	77.15	77.08	77.81	76.36
			Collins	77.89	79.02	76.80	76.57	77.30	75.85
			Hwa	77.12	78.25	76.03	75.91	76.66	75.18
Not Labeled	Aligned	Bracket	N/A	76.73	77.98	75.51	75.40	76.15	74.67
		Head Dep	Charniak	82.61	84.46	80.84	81.01	82.59	79.50
			Collins	82.12	83.96	80.36	80.37	81.93	78.87
			Hwa	81.10	82.92	79.36	79.59	81.14	78.11
		Open Class Dep	Charniak	82.54	83.73	81.39	81.20	81.97	80.44
			Collins	81.82	83.00	80.68	80.34	81.11	79.58
			Hwa	80.89	82.08	79.75	79.58	80.36	78.81
	Not Aligned	Head Dep	Charniak	82.81	84.67	81.04	81.41	82.99	79.89
			Collins	82.33	84.18	80.57	80.77	82.33	79.26
			Hwa	81.29	83.11	79.55	79.94	81.49	78.44
		Open Class Dep	Charniak	82.72	83.90	81.56	81.46	82.23	80.70
			Collins	82.00	83.19	80.85	80.59	81.37	79.84
			Hwa	81.09	82.27	79.93	79.85	80.64	79.07

Table 38: Charniak parser results on reference transcripts with system metadata.

Charniak Parser: ASR Transcript, Reference Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	76.57	75.68	77.48	71.35	71.74	70.97
		Head Dep	Charniak	72.60	71.93	73.27	69.10	69.87	68.35
			Collins	71.57	70.91	72.24	67.92	68.67	67.18
			Hwa	71.58	70.92	72.25	68.04	68.79	67.30
		Open Class Dep	Charniak	73.06	71.98	74.17	69.68	69.83	69.53
			Collins	71.83	70.75	72.94	68.28	68.44	68.12
			Hwa	71.92	70.89	72.98	68.55	68.71	68.39
	Not Aligned	Head Dep	Charniak	72.84	72.17	73.52	69.84	70.61	69.08
			Collins	71.81	71.15	72.48	68.63	69.40	67.89
			Hwa	71.83	71.17	72.50	68.75	69.51	68.00
		Open Class Dep	Charniak	73.31	72.22	74.42	70.38	70.53	70.23
			Collins	72.09	71.01	73.20	68.96	69.12	68.80
			Hwa	72.20	71.17	73.26	69.26	69.43	69.10
Not Labeled	Aligned	Bracket	N/A	79.81	78.89	80.76	75.17	75.57	74.77
		Head Dep	Charniak	76.42	75.72	77.14	72.87	73.67	72.08
			Collins	75.34	74.65	76.04	71.62	72.42	70.84
			Hwa	75.19	74.50	75.89	71.63	72.42	70.85
		Open Class Dep	Charniak	76.77	75.64	77.94	73.37	73.52	73.21
			Collins	75.27	74.14	76.43	71.68	71.84	71.51
			Hwa	75.35	74.28	76.46	71.96	72.13	71.79
	Not Aligned	Head Dep	Charniak	76.65	75.94	77.36	73.56	74.38	72.76
			Collins	75.56	74.86	76.26	72.32	73.12	71.53
			Hwa	75.42	74.73	76.12	72.29	73.09	71.50
		Open Class Dep	Charniak	77.01	75.87	78.18	74.02	74.17	73.86
			Collins	75.52	74.39	76.69	72.33	72.50	72.16
			Hwa	75.62	74.54	76.73	72.62	72.79	72.44

Table 39: Charniak parser results on ASR transcripts with reference metadata.

Charniak Parser: ASR Transcript, System Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	64.00	64.68	63.33	62.92	62.94	62.90
		Head Dep	Charniak	66.41	67.55	65.31	65.66	66.07	65.26
			Collins	65.38	66.50	64.29	64.58	64.98	64.18
			Hwa	65.34	66.46	64.25	64.54	64.94	64.14
		Open Class Dep	Charniak	66.65	67.14	66.16	66.09	65.74	66.44
			Collins	65.35	65.83	64.89	64.75	64.41	65.09
			Hwa	65.50	66.01	65.00	64.87	64.56	65.18
	Not Aligned	Head Dep	Charniak	66.96	68.11	65.85	66.36	66.78	65.95
			Collins	65.92	67.05	64.82	65.26	65.66	64.86
			Hwa	65.90	67.03	64.80	65.22	65.63	64.82
		Open Class Dep	Charniak	67.20	67.70	66.71	66.73	66.38	67.08
			Collins	65.90	66.38	65.43	65.37	65.03	65.72
			Hwa	66.10	66.62	65.59	65.56	65.25	65.88
Not Labeled	Aligned	Bracket	N/A	67.16	67.88	66.45	66.26	66.28	66.24
		Head Dep	Charniak	70.02	71.22	68.86	69.22	69.65	68.80
			Collins	68.90	70.09	67.76	68.05	68.47	67.63
			Hwa	68.80	69.98	67.66	67.97	68.39	67.55
		Open Class Dep	Charniak	69.99	70.51	69.48	69.41	69.05	69.78
			Collins	68.45	68.95	67.96	67.84	67.48	68.19
			Hwa	68.69	69.23	68.16	68.03	67.71	68.36
	Not Aligned	Head Dep	Charniak	70.53	71.74	69.36	69.89	70.32	69.46
			Collins	69.41	70.60	68.25	68.70	69.13	68.28
			Hwa	69.32	70.51	68.17	68.61	69.04	68.19
		Open Class Dep	Charniak	70.51	71.03	69.99	70.01	69.64	70.38
			Collins	68.96	69.46	68.47	68.43	68.07	68.79
			Hwa	69.23	69.77	68.70	68.68	68.35	69.01

Table 40: Charniak parser results on ASR transcripts with system metadata.

Roark Parser: Reference Transcript, Reference Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	85.65	85.20	86.11	78.94	79.96	77.94
		Head Dep	Charniak	83.26	83.40	83.12	78.08	79.72	76.50
			Collins	83.22	83.36	83.08	77.91	79.56	76.33
			Hwa	82.23	82.36	82.09	76.93	78.55	75.37
		Open Class Dep	Charniak	83.85	83.60	84.10	78.66	79.88	77.48
			Collins	83.83	83.65	84.02	78.48	79.72	77.27
			Hwa	82.67	82.49	82.84	77.36	78.57	76.18
	Not Aligned	Head Dep	Charniak	83.31	83.45	83.17	78.53	80.19	76.94
			Collins	83.28	83.42	83.14	78.37	80.02	76.78
			Hwa	82.27	82.41	82.13	77.34	78.97	75.77
		Open Class Dep	Charniak	83.89	83.64	84.15	79.09	80.31	77.90
			Collins	83.87	83.68	84.05	78.90	80.15	77.69
			Hwa	82.70	82.53	82.88	77.76	78.98	76.58
Not Labeled	Aligned	Bracket	N/A	87.57	87.11	88.04	81.47	82.52	80.44
		Head Dep	Charniak	88.05	88.20	87.91	82.75	84.50	81.08
			Collins	87.86	88.01	87.71	82.54	84.28	80.87
			Hwa	86.34	86.48	86.19	81.03	82.74	79.39
		Open Class Dep	Charniak	88.49	88.22	88.76	83.20	84.49	81.95
			Collins	88.14	87.95	88.34	82.77	84.08	81.50
			Hwa	86.52	86.34	86.70	81.18	82.45	79.94
	Not Aligned	Head Dep	Charniak	88.10	88.25	87.95	83.18	84.94	81.50
			Collins	87.91	88.06	87.77	82.99	84.74	81.31
			Hwa	86.37	86.52	86.23	81.44	83.16	79.79
		Open Class Dep	Charniak	88.52	88.26	88.79	83.57	84.87	82.31
			Collins	88.18	87.98	88.38	83.16	84.48	81.89
			Hwa	86.54	86.36	86.73	81.56	82.84	80.32

Table 41: Roark parser results on reference transcripts with reference metadata.

Roark Parser: Reference Transcript, System Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	72.90	74.11	71.72	70.53	71.55	69.53
		Head Dep	Charniak	75.67	77.37	74.05	74.03	75.44	72.66
			Collins	75.58	77.27	73.95	73.91	75.33	72.55
			Hwa	74.63	76.30	73.03	72.82	74.21	71.48
		Open Class Dep	Charniak	75.86	77.01	74.74	74.33	75.23	73.46
			Collins	75.76	76.96	74.59	74.17	75.09	73.27
			Hwa	74.76	75.95	73.61	73.01	73.92	72.13
	Not Aligned	Head Dep	Charniak	75.88	77.58	74.25	74.46	75.88	73.08
			Collins	75.79	77.49	74.16	74.31	75.73	72.94
			Hwa	74.82	76.50	73.21	73.20	74.60	71.85
		Open Class Dep	Charniak	76.04	77.20	74.91	74.73	75.63	73.85
			Collins	75.95	77.15	74.78	74.52	75.44	73.62
			Hwa	74.95	76.14	73.80	73.38	74.29	72.49
Not Labeled	Aligned	Bracket	N/A	75.01	76.26	73.80	73.01	74.07	71.98
		Head Dep	Charniak	80.29	82.09	78.56	78.45	79.95	77.01
			Collins	80.03	81.83	78.32	78.24	79.74	76.80
			Hwa	78.65	80.41	76.96	76.78	78.25	75.36
		Open Class Dep	Charniak	80.27	81.49	79.08	78.57	79.51	77.64
			Collins	79.84	81.11	78.62	78.10	79.07	77.16
			Hwa	78.49	79.73	77.28	76.65	77.60	75.73
	Not Aligned	Head Dep	Charniak	80.45	82.26	78.73	78.85	80.36	77.40
			Collins	80.24	82.04	78.52	78.64	80.15	77.19
			Hwa	78.82	80.58	77.12	77.16	78.64	75.74
		Open Class Dep	Charniak	80.39	81.62	79.20	78.90	79.85	77.97
			Collins	80.01	81.27	78.78	78.44	79.41	77.49
			Hwa	78.65	79.90	77.44	77.00	77.95	76.07

Table 42: Roark parser results on reference transcripts with system metadata.

Roark Parser: ASR Transcript, Reference Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	74.13	73.24	75.04	69.20	69.64	68.77
		Head Dep	Charniak	69.92	69.30	70.55	66.42	67.13	65.73
			Collins	69.06	68.45	69.69	65.64	66.34	64.95
			Hwa	68.78	68.17	69.40	65.20	65.89	64.52
		Open Class Dep	Charniak	70.26	69.30	71.26	66.87	67.16	66.58
			Collins	69.22	68.28	70.19	65.86	66.15	65.58
			Hwa	69.05	68.15	69.96	65.51	65.83	65.19
	Not Aligned	Head Dep	Charniak	70.16	69.54	70.80	67.14	67.85	66.44
			Collins	69.29	68.67	69.92	66.31	67.02	65.62
			Hwa	69.02	68.41	69.65	65.91	66.61	65.22
		Open Class Dep	Charniak	70.54	69.57	71.53	67.61	67.90	67.31
			Collins	69.49	68.54	70.46	66.56	66.85	66.28
			Hwa	69.33	68.43	70.25	66.26	66.59	65.94
Not Labeled	Aligned	Bracket	N/A	77.37	76.44	78.32	73.01	73.48	72.55
		Head Dep	Charniak	74.16	73.50	74.84	70.60	71.35	69.86
			Collins	73.25	72.60	73.91	69.74	70.49	69.02
			Hwa	72.85	72.20	73.51	69.16	69.90	68.44
		Open Class Dep	Charniak	74.34	73.32	75.39	70.88	71.19	70.57
			Collins	73.14	72.14	74.16	69.68	69.98	69.38
			Hwa	72.92	71.98	73.89	69.27	69.61	68.93
	Not Aligned	Head Dep	Charniak	74.40	73.74	75.08	71.30	72.06	70.55
			Collins	73.48	72.82	74.14	70.44	71.19	69.71
			Hwa	73.08	72.43	73.75	69.88	70.63	69.16
		Open Class Dep	Charniak	74.61	73.58	75.66	71.57	71.89	71.26
			Collins	73.39	72.40	74.42	70.36	70.67	70.06
			Hwa	73.19	72.24	74.16	70.02	70.36	69.68

Table 43: Roark parser results on ASR transcripts with reference metadata.

Roark Parser: ASR Transcript, System Metadata				Use Edit MDE			Ignore Edit MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	62.22	62.96	61.51	61.47	61.77	61.18
		Head Dep	Charniak	63.87	64.99	62.78	63.50	64.01	63.00
			Collins	63.02	64.13	61.95	62.81	63.31	62.32
			Hwa	62.64	63.75	61.58	62.16	62.66	61.67
		Open Class Dep	Charniak	64.12	64.63	63.61	63.77	63.68	63.86
			Collins	63.05	63.57	62.54	62.89	62.83	62.96
			Hwa	62.83	63.40	62.28	62.34	62.33	62.36
	Not Aligned	Head Dep	Charniak	64.37	65.50	63.28	64.13	64.65	63.63
			Collins	63.52	64.63	62.44	63.41	63.91	62.91
			Hwa	63.14	64.25	62.07	62.80	63.30	62.31
		Open Class Dep	Charniak	64.65	65.17	64.14	64.43	64.34	64.52
			Collins	63.59	64.12	63.08	63.52	63.46	63.59
			Hwa	63.37	63.94	62.81	63.04	63.03	63.06
Not Labeled	Aligned	Bracket	N/A	65.57	66.34	64.82	64.94	65.25	64.63
		Head Dep	Charniak	67.83	69.02	66.68	67.42	67.96	66.89
			Collins	66.91	68.09	65.78	66.66	67.19	66.13
			Hwa	66.40	67.56	65.27	65.89	66.42	65.37
		Open Class Dep	Charniak	67.83	68.37	67.29	67.48	67.38	67.57
			Collins	66.56	67.11	66.02	66.39	66.32	66.46
			Hwa	66.35	66.95	65.76	65.87	65.86	65.89
	Not Aligned	Head Dep	Charniak	68.29	69.49	67.13	68.05	68.59	67.51
			Collins	67.41	68.59	66.26	67.25	67.79	66.72
			Hwa	66.87	68.05	65.73	66.52	67.05	66.00
		Open Class Dep	Charniak	68.31	68.85	67.77	68.10	68.00	68.19
			Collins	67.08	67.63	66.53	66.98	66.91	67.05
			Hwa	66.84	67.44	66.25	66.54	66.53	66.56

Table 44: Roark parser results on ASR transcripts with system metadata.

Charniak Parser: System Metadata, Use Edit MDE				Reference Transcript			IBM+SRI			SRI ASR		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	74.34	75.56	73.16	64.00	64.68	63.33	61.13	61.15	61.11
		Head Dep	Charniak	77.95	79.70	76.28	66.41	67.55	65.31	64.01	64.39	63.63
			Collins	77.64	79.38	75.98	65.38	66.50	64.29	63.24	63.62	62.87
			Hwa	76.88	78.60	75.23	65.34	66.46	64.25	62.94	63.32	62.57
		Open Class Dep	Charniak	78.03	79.14	76.94	66.65	67.14	66.16	64.26	63.97	64.56
			Collins	77.69	78.82	76.60	65.35	65.83	64.89	63.22	62.95	63.49
			Hwa	76.90	78.02	75.81	65.50	66.01	65.00	63.02	62.80	63.24
	Not Aligned	Head Dep	Charniak	78.17	79.93	76.50	66.96	68.11	65.85	64.50	64.89	64.12
			Collins	77.86	79.61	76.19	65.92	67.05	64.82	63.74	64.12	63.36
			Hwa	77.09	78.82	75.43	65.90	67.03	64.80	63.46	63.84	63.08
		Open Class Dep	Charniak	78.24	79.36	77.15	67.20	67.70	66.71	64.77	64.48	65.07
			Collins	77.89	79.02	76.80	65.90	66.38	65.43	63.73	63.46	64.01
			Hwa	77.12	78.25	76.03	66.10	66.62	65.59	63.57	63.35	63.79
Not Labeled	Aligned	Bracket	N/A	76.73	77.98	75.51	67.16	67.88	66.45	64.51	64.53	64.49
		Head Dep	Charniak	82.61	84.46	80.84	70.02	71.22	68.86	67.45	67.85	67.05
			Collins	82.12	83.96	80.36	68.90	70.09	67.76	66.56	66.95	66.16
			Hwa	81.10	82.92	79.36	68.80	69.98	67.66	66.20	66.60	65.81
		Open Class Dep	Charniak	82.54	83.73	81.39	69.99	70.51	69.48	67.43	67.12	67.74
			Collins	81.82	83.00	80.68	68.45	68.95	67.96	66.09	65.81	66.38
			Hwa	80.89	82.08	79.75	68.69	69.23	68.16	65.93	65.71	66.16
	Not Aligned	Head Dep	Charniak	82.81	84.67	81.04	70.53	71.74	69.36	67.92	68.32	67.52
			Collins	82.33	84.18	80.57	69.41	70.60	68.25	67.03	67.43	66.63
			Hwa	81.29	83.11	79.55	69.32	70.51	68.17	66.70	67.10	66.31
		Open Class Dep	Charniak	82.72	83.90	81.56	70.51	71.03	69.99	67.91	67.60	68.23
			Collins	82.00	83.19	80.85	68.96	69.46	68.47	66.57	66.28	66.86
			Hwa	81.09	82.27	79.93	69.23	69.77	68.70	66.45	66.22	66.69
Charniak Parser: System Metadata, Ignore Edit MDE				Reference Transcript			IBM+SRI			SRI ASR		
Labeled	Aligned	Bracket	N/A	72.72	73.44	72.01	62.92	62.94	62.90	60.23	59.85	60.61
		Head Dep	Charniak	76.46	77.95	75.03	65.66	66.07	65.26	63.42	63.37	63.48
			Collins	76.02	77.49	74.60	64.58	64.98	64.18	62.63	62.58	62.69
			Hwa	75.41	76.87	74.00	64.54	64.94	64.14	62.35	62.30	62.41
		Open Class Dep	Charniak	76.77	77.50	76.06	66.09	65.74	66.44	63.81	63.02	64.62
			Collins	76.30	77.03	75.59	64.75	64.41	65.09	62.76	62.02	63.53
			Hwa	75.61	76.36	74.88	64.87	64.56	65.18	62.56	61.88	63.25
	Not Aligned	Head Dep	Charniak	76.89	78.38	75.45	66.36	66.78	65.95	64.09	64.04	64.15
			Collins	76.41	77.90	74.98	65.26	65.66	64.86	63.29	63.23	63.34
			Hwa	75.77	77.24	74.35	65.22	65.63	64.82	62.99	62.94	63.50
		Open Class Dep	Charniak	77.08	77.81	76.36	66.73	66.38	67.08	64.43	63.64	65.25
			Collins	76.57	77.30	75.85	65.37	65.03	65.72	63.37	62.62	64.15
			Hwa	75.91	76.66	75.18	65.56	65.25	65.88	63.20	62.51	63.90
Not Labeled	Aligned	Bracket	N/A	75.40	76.15	74.67	66.26	66.28	66.24	63.70	63.29	64.10
		Head Dep	Charniak	81.01	82.59	79.50	69.22	69.65	68.80	66.76	66.71	66.82
			Collins	80.37	81.93	78.87	68.05	68.47	67.63	65.90	65.85	65.96
			Hwa	79.59	81.14	78.11	67.97	68.39	67.55	65.57	65.51	65.63
		Open Class Dep	Charniak	81.20	81.97	80.44	69.41	69.05	69.78	66.88	66.06	67.73
			Collins	80.34	81.11	79.58	67.84	67.48	68.19	65.63	64.85	66.43
			Hwa	79.58	80.36	78.81	68.03	67.71	68.36	65.46	64.74	66.18
	Not Aligned	Head Dep	Charniak	81.41	82.99	79.89	69.89	70.32	69.46	67.42	67.36	67.47
			Collins	80.77	82.33	79.26	68.70	69.13	68.28	66.54	66.48	66.60
			Hwa	79.94	81.49	78.44	68.61	69.04	68.19	66.19	66.14	66.25
		Open Class Dep	Charniak	81.46	82.23	80.70	70.01	69.64	70.38	67.49	66.65	68.34
			Collins	80.59	81.37	79.84	68.43	68.07	68.79	66.21	65.42	67.02
			Hwa	79.85	80.64	79.07	68.68	68.35	69.01	66.07	65.35	66.80

Table 45: Charniak Parser on three kinds of transcripts with system metadata.

Charniak Parser: Reference Transcripts, Use Edit MDE				Reference MDE			Baseline MDE System			Prosody Only MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision
labeled	Aligned	Bracket	N/A	88.07	87.64	88.51	74.34	75.56	73.16	62.08	63.18	61.01
		Head Dep	Charniak	85.68	85.82	85.53	77.95	79.70	76.28	70.12	70.86	69.39
			Collins	85.51	85.66	85.37	77.64	79.38	75.98	69.78	70.52	69.06
			Hwa	84.80	84.94	84.66	76.88	78.60	75.23	68.95	69.68	68.24
		Open Class Dep	Charniak	86.12	85.79	86.45	78.03	79.14	76.94	69.00	68.96	69.04
			Collins	86.05	85.75	86.36	77.69	78.82	76.60	68.48	68.47	68.49
			Hwa	85.16	84.87	85.45	76.90	78.02	75.81	67.87	67.83	67.92
	Not Aligned	Head Dep	Charniak	85.72	85.87	85.58	78.17	79.93	76.50	70.33	71.07	69.60
			Collins	85.57	85.71	85.43	77.86	79.61	76.19	70.00	70.74	69.28
			Hwa	84.84	84.98	84.70	77.09	78.82	75.43	69.15	69.88	68.43
		Open Class Dep	Charniak	86.15	85.83	86.48	78.24	79.36	77.15	69.19	69.15	69.23
			Collins	86.08	85.78	86.39	77.89	79.02	76.80	68.68	68.67	68.69
			Hwa	85.19	84.90	85.48	77.12	78.25	76.03	68.07	68.03	68.11
Not Labeled	Aligned	Bracket	N/A	90.29	89.85	90.74	76.73	77.98	75.51	64.33	65.47	63.23
		Head Dep	Charniak	90.35	90.50	90.20	82.61	84.46	80.84	74.66	75.45	73.88
			Collins	90.13	90.28	89.97	82.12	83.96	80.36	74.08	74.86	73.31
			Hwa	89.05	89.20	88.90	81.10	82.92	79.36	73.23	74.00	72.47
		Open Class Dep	Charniak	90.76	90.42	91.11	82.54	83.73	81.39	73.35	73.31	73.39
			Collins	90.41	90.09	90.73	81.82	83.00	80.68	72.42	72.41	72.44
			Hwa	89.25	88.94	89.55	80.89	82.08	79.75	71.83	71.79	71.88
	Not Aligned	Head Dep	Charniak	90.39	90.54	90.24	82.81	84.67	81.04	74.87	75.66	74.09
			Collins	90.18	90.33	90.02	82.33	84.18	80.57	74.30	75.09	73.53
			Hwa	89.08	89.23	88.93	81.29	83.11	79.55	73.42	74.20	72.66
		Open Class Dep	Charniak	90.79	90.45	91.14	82.72	83.90	81.56	73.54	73.49	73.58
			Collins	90.44	90.12	90.76	82.00	83.19	80.85	72.62	72.61	72.63
			Hwa	89.26	88.96	89.56	81.09	82.27	79.93	72.02	71.98	72.07

Table 46: Charniak Parser with three metadata quality inputs using edit metadata.

Charniak Parser: Reference Transcripts, Use Edit MDE				Reference MDE			Perfect SU & System Edit		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
labeled	aligned	Bracket	N/A	88.07	87.64	88.51	83.25	84.43	82.10
		Head Dep	Charniak	85.68	85.82	85.53	81.88	83.73	80.12
			Collins	85.51	85.66	85.37	81.54	83.38	79.79
			Hwa	84.80	84.94	84.66	81.08	82.90	79.33
		Open Class Dep	Charniak	86.12	85.79	86.45	82.18	83.53	80.86
			Collins	86.05	85.75	86.36	81.83	83.21	80.50
			Hwa	85.16	84.87	85.45	81.29	82.65	79.97
	Not Aligned	Head Dep	Charniak	85.72	85.87	85.58	82.12	83.97	80.35
			Collins	85.57	85.71	85.43	81.79	83.63	80.03
			Hwa	84.84	84.98	84.70	81.32	83.15	79.56
		Open Class Dep	Charniak	86.15	85.83	86.48	82.40	83.76	81.08
			Collins	86.08	85.78	86.39	82.06	83.44	80.73
			Hwa	85.19	84.90	85.48	81.53	82.89	80.21
NOT Labeled	Aligned	Bracket	N/A	90.29	89.85	90.74	85.83	87.04	84.65
		Head Dep	Charniak	90.35	90.50	90.20	86.64	88.59	84.78
			Collins	90.13	90.28	89.97	86.18	88.12	84.33
			Hwa	89.05	89.20	88.90	85.40	87.32	83.56
		Open Class Dep	Charniak	90.76	90.42	91.11	86.90	88.33	85.51
			Collins	90.41	90.09	90.73	86.21	87.66	84.81
			Hwa	89.25	88.94	89.55	85.46	86.89	84.08
	Not Aligned	Head Dep	Charniak	90.39	90.54	90.24	86.86	88.82	84.99
			Collins	90.18	90.33	90.02	86.42	88.37	84.56
			Hwa	89.08	89.23	88.93	85.60	87.53	83.76
		Open Class Dep	Charniak	90.79	90.45	91.14	87.09	88.53	85.70
			Collins	90.44	90.12	90.76	86.42	87.88	85.02
			Hwa	89.26	88.96	89.56	85.67	87.10	84.28
Charniak Parser: Reference Transcripts, Use Edit MDE				Perfect Edit & System SU			System MDE		
Labeling	Alignment	Match Type	Head Rules	F-measure	Recall	Precision	F-measure	Recall	Precision
Labeled	Aligned	Bracket	N/A	77.86	77.85	77.87	74.34	75.56	73.16
		Head Dep	Charniak	80.87	81.17	80.58	77.95	79.70	76.28
			Collins	80.63	80.92	80.33	77.64	79.38	75.98
			Hwa	79.82	80.11	79.53	76.88	78.60	75.23
		Open Class Dep	Charniak	81.00	80.71	81.29	78.03	79.14	76.94
			Collins	80.72	80.44	81.00	77.69	78.82	76.60
			Hwa	79.85	79.62	80.09	76.90	78.02	75.81
	Not Aligned	Head Dep	Charniak	80.94	81.24	80.65	78.17	79.93	76.50
			Collins	80.69	80.99	80.40	77.86	79.61	76.19
			Hwa	79.87	80.17	79.58	77.09	78.82	75.43
		Open Class Dep	Charniak	81.06	80.77	81.35	78.24	79.36	77.15
			Collins	80.76	80.48	81.04	77.89	79.02	76.80
			Hwa	79.90	79.67	80.14	77.12	78.25	76.03
Not Labeled	Aligned	Bracket	N/A	79.92	79.91	79.93	76.73	77.98	75.51
		Head Dep	Charniak	85.55	85.86	85.24	82.61	84.46	80.84
			Collins	85.20	85.51	84.89	82.12	83.96	80.36
			Hwa	84.04	84.34	83.73	81.10	82.92	79.36
		Open Class Dep	Charniak	85.59	85.28	85.89	82.54	83.73	81.39
			Collins	84.98	84.69	85.28	81.82	83.00	80.68
			Hwa	83.88	83.64	84.13	80.89	82.08	79.75
	unaligned	Head Dep	Charniak	85.61	85.92	85.30	82.81	84.67	81.04
			Collins	85.25	85.57	84.94	82.33	84.18	80.57
			Hwa	84.08	84.39	83.77	81.29	83.11	79.55
		Open Class Dep	Charniak	85.63	85.33	85.94	82.72	83.90	81.56
			Collins	85.02	84.73	85.31	82.00	83.19	80.85
			Hwa	83.91	83.67	84.16	81.09	82.27	79.93

Table 47: Charniak Parser on reference transcripts using reference and system metadata for SUs and/or Edits.

B MINI-BJD String Rewriting Rules

```
;; Fix bad tokenization on dashes
") (U -)" "-)"
")-)" "-))"

;; Fix non-constituents
"(U < na >)" ""
"(U <na>)" ""
"< na >" ""
"<na>" ""

;; Fix unknowns
"UNK%" "U"

;; Eliminate brackets
"[" ""
"]" ""

;; Fix edits and fillers
"< EDIT_END >" "<EDIT_END>"
"< EDIT_ST >" "<EDIT_ST>"
"< FL_END >" "<FL_END>"
"< FL_ST >" "<FL_ST>"

;; Fix weird cases of edits and fillers
"(INTJ (UH <EDIT_ST>) #)" "(UH <EDIT_ST>) #"
"^(+ + <EDIT_END>)^" "(U <EDIT_ST>) (+ +) (U <EDIT_END>)"
"(UH <EDIT_ST>)" "(U <EDIT_ST>)"
"(UH <EDIT_END>)" "(U <EDIT_END>)"
"(UH <FL_ST>)" "(U <FL_ST>)"
"(UH <FL_END>)" "(U <FL_END>)"

;; Wrap U around edits and fillers
"(+ # + <EDIT_END>)" "(+ # +) (U <EDIT_END>)"
"(+ # + <EDIT_ST>)" "(+ # +) (U <EDIT_ST>)"
"(+ # + <FL_END>)" "(+ # +) (U <FL_END>)"
"(+ # + <FL_ST>)" "(+ # +) (U <FL_ST>)"

;; Put EDIT_END and FL_END at highest level (under top node).
;; (U (C (N (A s-) (U <EDIT_END>) (P in)) (V run))) ->
;; (U (C (N (A s-))) (U <EDIT_END>) (P in) (V run))
" (U <EDIT_END>) #)" ") (U <EDIT_END>) #"
"^(+ #) (U <EDIT_END>) #^" "(+ # (U <EDIT_END>) #)"
" (U <FL_END>) #)" ") (U <FL_END>) #"
"^(+ #) (U <FL_END>) #^" "(+ # (U <FL_END>) #)"

;; Put EDIT_ST and FL_ST at highest level (under top node).
;; (U (N (A roll-) (U <FL_ST>) (N um)) (U <FL_END>) (U to)) ->
;; (U (N (A roll-)) (U <FL_ST>) (N um) (U <FL_END>) (U to))
" (U <EDIT_ST>) #)" ") (U <EDIT_ST>) #"
"^(+ #) (U <EDIT_ST>) #^" "(+ # (U <EDIT_ST>) #)"
" (U <FL_ST>) #)" ") (U <FL_ST>) #"
"^(+ #) (U <FL_ST>) #^" "(+ # (U <FL_ST>) #)"
```

```
;; Now replace edits and fillers with appropriate categories.
"(U <EDIT_ST>) # (U <EDIT_END>)" "(EDITED # (DISFL-IP +))"
"(U <FL_ST>) (+ +) (U <FL_END>)" "(INTJ% (+ +))"
"(U <FL_ST>) # (U <FL_END>)" "(PRN #)"

;; Get rid of all remaining fillers and edits (that had no matching counterpart)
"(U <FL_ST>)" ""
"(U <FL_END>)" ""
"(U <EDIT_ST>)" ""
"(U <EDIT_END>)" ""

;; Weird preprocessing for intransitive verbs
"(V # (V +))" "(V # (V-dont-change +))"
"(V + #)" "(V (V% + #))"
"(V% + #)" "(V + #)"
"(V-dont-change +)" "(V +)"

;; Preprocessing: Remove subcategorization
"(N_N_P #)" "(N #)"
"(A_N_P #)" "(A #)"
"(P_N_P #)" "(P #)"
"(V_N_P #)" "(V #)"

"(N_N_N #)" "(N #)"
"(A_N_N #)" "(A #)"
"(P_N_N #)" "(P #)"
"(V_N_N #)" "(V #)"

"(N_N_C #)" "(N #)"
"(A_N_C #)" "(A #)"
"(P_N_C #)" "(P #)"
"(V_N_C #)" "(V #)"

"(N_C_N #)" "(N #)"
"(A_C_N #)" "(A #)"
"(P_C_N #)" "(P #)"
"(V_C_N #)" "(V #)"

"(N_C_P #)" "(N #)"
"(A_C_P #)" "(A #)"
"(P_C_P #)" "(P #)"
"(V_C_P #)" "(V #)"

"(N_P_C #)" "(N #)"
"(A_P_C #)" "(A #)"
"(P_P_C #)" "(P #)"
"(V_P_C #)" "(V #)"

"(N_P_P #)" "(N #)"
"(A_P_P #)" "(A #)"
"(P_P_P #)" "(P #)"
"(V_P_P #)" "(V #)"
```

```
"(N_P #)" "(N #)"
"(A_P #)" "(A #)"
"(P_P #)" "(P #)"
"(V_P #)" "(V #)"

"(N_N #)" "(N #)"
"(A_N #)" "(A #)"
"(P_N #)" "(P #)"
"(V_N #)" "(V #)"

"(N_A #)" "(N #)"
"(A_A #)" "(A #)"
"(P_A #)" "(P #)"
"(V_A #)" "(V #)"

"(N_C #)" "(N #)"
"(A_C #)" "(A #)"
"(P_C #)" "(P #)"
"(V_C #)" "(V #)"

"(N_I #)" "(N #)"
"(A_I #)" "(A #)"
"(P_I #)" "(P #)"
"(V_I #)" "(V #)"

"(NN #)" "(N #)"

;; Sometimes the edit-deletion rule makes weird groupings that need to be reduced.
;"(N (+ +))" "(+ +)"
;(A (+ +))" "(+ +)"
;"(V (+ +))" "(+ +)"
;"(P (+ +))" "(+ +)"

;; Preprocessing: Multi-lexical-item Single-lexical-item

"(N # (N +) # (N + + #)" "(N # (N +) # (NML (N +) (N + #))" ; introducing NML
"(N # (N + + #) # (N +)" "(N # (NML (N +) (N + #)) # (N +)" ; introducing NML
"(NML # (N +) (N + + #)" "(NML # (N +) (N +) (N + #)"       ; introducing NML
"(N # (N + + #) #)" "(N # (N +) (N + #) #)"
"(N + + #)" "(POS +) (N + #)"
"(A # + + #)" "(A # (A% +) (A% +) #)"
"(A # (A% +) + #)" "(A # (A% +) (A% +) #)"
"(A # + (A% +) #)" "(A # (A% +) (A% +) #)"
"A%" "A"
"(A (A (A +) #) #)" "(A (A +) # #)" ; get rid of overzealous A insertion.
"(P + + #)" "(P +) (P + #)"
"(V + + #)" "(V +) (V + #)"

;; Be rules for verbal complements
"(Be +) (V (A n't) #)" "(V (V +) (RB n't) #)"
"(Be aren't) # (V #)" "(V (V are) (RB n't) # #)"
"(Be isn't) # (V #)" "(V (V is) (RB n't) # #)"
"(Be wasn't) # (V #)" "(V (V was) (RB n't) # #)"
"(Be weren't) # (V #)" "(V (V were) (RB n't) # #)"
```

```
"(Be + not) # (V #)" "(V (V +) (RB not) # #)"
"(Be +) # (V #)" "(V (V +) # #)"
"(Be +) # (V #)" "(V (V +) # #)"

;; Other Be rules
"(# (Be aren't) #)" "(# (V (V are) (RB n't) #))"
"(# (Be isn't) #)"  "(# (V (V is) (RB n't) #))"
"(# (Be wasn't) #)" "(# (V (V was) (RB n't) #))"
"(# (Be weren't) #)" "(# (V (V were) (RB n't) #))"
"(# (Be + not) #)"  "(# (V (V +) (RB not) #))"
"(# (Be +) #)"      "(# (V (V +) #))"

;; Have rules for verbal complements
"(Have +) (V (A n't) #)" "(V (V +) (RB n't) #)"
"(Have hadn't) # (V #)" "(V (V had) (RB n't) # #)"
"(Have hasn't) # (V #)" "(V (V has) (RB n't) # #)"
"(Have haven't) # (V #)" "(V (V have) (RB n't) # #)"
"(Have +) # (V #)" "(V (V +) # #)"
"(Have + not) # (V #)" "(V (V +) (RB not) # #)"
"(Have + to) (V #)" "(V (V +) (S (NP-SBJ (-NONE- *)) (VP (TO to) #)))"

;; Adverb/Adjective rules
"(A # (A +) # )" "(A% # (POS +) # )"
"(A% # (A +) # )" "(A% # (POS +) # )"

"(N # (A +)" "(N # (POS +)"
"(N # (A% #)" "(N # (ADJP #)"

"(V # (A +)" "(V # (ADVP (POS +))"
"(V # (A% #)" "(V # (ADVP #)"

"(C # (A +)" "(C # (ADVP (POS +))"
"(C # (A% #)" "(C # (ADVP #)"

"(I # (A +)" "(I # (ADVP (POS +))"
"(I # (A% #)" "(I # (ADVP #)"

;; Subject rules
"(I (N there)" "(S (NP-SBJ (EX there))"
"(I (N)" "(S (NP-SBJ (-NONE- *))"
"(I (N #)" "(S (NP-SBJ #)"
"(I (THAT +)" "(S (NP-SBJ (N +))"
"(NP-SBJ +)" "(NP-SBJ (POS +))"

;; NP rules
"(NP-SBJ # (N +)" "(NP-SBJ # (POS +)"
"(NML # (N +) #)" "(NML # (POS +) #)"
"(N # (N +)" "(NP # (POS +)"
"(N +)" "(NP (POS +))"
"(N)" "(NP (-NONE- *))"
"(PRO)" "(NP-SBJ (-NONE- *))"
"(LPRO)" "(NP-SBJ (-NONE- *))"
"(NP # (NP #) #)"  "(NP # # #)"    ; Get rid of overzealous NP insertion
```

```
;; Sentential rules
"(C (COMP +)" "(SBAR (POS +)"
"(I #)" "(S #)"
"(C (S #))" "(S #)"
"(C # (S" "(SBAR # (S"
"(SentAdjunct + + #)" "(ADVP (POS +) (SentAdjunct% + #))"
"(SentAdjunct% + #)" "(POS +) (SentAdjunct% #)"
"(+ +) (SentAdjunct%)" "(+ +)"
"(SentAdjunct (ADVP" "(SBAR (ADVP"
"(SentAdjunct (SentAdjunct +)" "(SBAR (POS +)" ;; Conjunction

;; Possessive pronoun and determiner rules
"(Det + + #)" "(POS +) (Det + #)"
"(Det +)" "(POS +)"
"(PostDet + + #)" "(POS +) (PostDet + #)"
"(PostDet +)" "(POS +)"
"(PreDet + + #)" "(POS +) (PreDet + #)"
"(PreDet +)" "(POS +)"

;; PP rules
"(P (PpSpec +) (P +" "(PP-LOC-PRD (ADVP (POS +)) (POS +"
"(P # (P +)" "(PP # (POS +)"
"(P +)" "(POS +)"

;; VP rules
"(V # (V +)" "(VP # (VB% +)"
"(V +)" "(VB% +)"
"(Aux + + #) # (VP #" "(VP (Aux% + + #) # (VP #)"
"(Aux% + + #)" "(POS +) (Aux% + #)"
"(Aux% +)" "(POS +)"
"(Aux +) # (VP #" "(VP (POS +) # (VP #)"

;; Predicative rules
"(VP # (VB% +) (NP #) #)" "(VP # (POS +) (NP #) #)"
"(VP # (VB% +) (A #) #)"  "(VP # (POS +) (A #) #)"
"(VP # (VB% +) (RB #) #)"  "(VP # (POS +) (RB #) #)"
"(VP # (VB% +) (PP #) #)" "(VP # (POS +) (PP #) #)"
"(VP # (VB% +) (VP #) #)" "(VP # (POS +) (VP #) #)"
"(VP # (VB% +) (S #) #)" "(VP # (POS +) (S #) #)"
"(VP # (VB% +) (SBAR #) #)" "(VP # (POS +) (SBAR #) #)"
"(VB% +)" "(POS +)"

;; AP rules (adverbials and adjectives)
"(A +)" "(POS +)"
"(A (JJ +))" "(POS +)"
"(A (JJR" "(ADJP (JJR"
"(A # (WRB +) #)" "(WHADVP # (WRB +) #)"
"(A # (RB +) #)" "(ADVP # (RB +) #)"
"(RB% +)" "(RB +)"
"(A #)" "(ADJP #)" ; Remaining adjectives.

;; Backoff to POS and U-removal
"(U +)" "(POS +)"
"(U #)" "#"
```

```
;; Interjections
"(INJ #)" "(INTJ #)"
"(INTJ% (NP (UH +)))" "(INTJ (UH% +))"
"(UH +)" "(INTJ (UH% +))"
"(NP (INTJ (UH% +)))" "(INTJ (UH% +))"
"UH%" "UH"
"INTJ%" "INTJ"
"(INTJ (INTJ #)" "(INTJ #" ; Get rid of overzealous INTJ insertion.

;; Add top-node at the end.
"^#^" "(S1 #)"

;; Fix "unfound word" tag (edited).
"(NP (XXX +))" "(EDITED (X (XX +)) (DISFL-IP +))"
"(XXX +)" "(EDITED (X (XX +)) (DISFL-IP +))"
"(EDITED (X (XX +)) (DISFL-IP +))"
"(XXX +)" "(EDITED (X (XX +)) (DISFL-IP +))"

;; Binary branching!
"(S # (VB+ #) #)" "(S # (VP (VB+ #) #))"
"(S # (VB #) #)" "(S # (VP (VB #) #))"

"(SBAR # (VB+ #) #)" "(SBAR # (VP (VB+ #) #))"
"(SBAR # (VB #) #)" "(SBAR # (VP (VB #) #))"

"(NP # (VB+ #) #)" "(NP # (VP (VB+ #) #))"
"(NP # (VB #) #)" "(NP # (VP (VB #) #))"

"(PP # (VB+ #) #)" "(PP # (VP (VB+ #) #))"
"(PP # (VB #) #)" "(PP # (VP (VB #) #))"

"(VP # (VB+ #) # (VB+ #) #)" "(VP # (VB+ #) # (VP (VB+ #) #))"
"(VP # (VB #) # (VB+ #) #)" "(VP # (VB #) # (VP (VB+ #) #))"
"(VP # (VB+ #) # (VB #) #)" "(VP # (VB+ #) # (VP (VB #) #))"
"(VP # (VB #) # (VB #) #)" "(VP # (V+ #) # (VP (VB #) #))"
```

C Acknowledgments

The authors would like to thank CLSP at Johns Hopkins for hosting the summer workshop where much of this work was done. We would also like to thank Eugene Charniak, Dustin Hillard, Mark Johnson, Jeremy Kahn, Mari Ostendorf, Elizabeth Shriberg, Andreas Stolcke, and Wen Wang for advice, data, know how, and software made available to the team during the course of this project. We thank Stephanie Strassel, Ann Bies, and the LDC treebanking team for their assistance on the RT'04 CTS Treebank, as well as NSA and DARPA for providing additional funds to support the effort. This report is based upon work supported by DARPA under contract number MDA972-02-C-0038, by the National Science Foundation (NSF) under grant numbers 0121285 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.

References

[BAF$^+$91] 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. A procedure for quantitatively comparing syntactic coverage of English grammars. In *Proceedings 4th DARPA Speech & Natural Lang. Workshop*, pages 306–311, 1991.

[BE93] M. Beckman and G. A. Elam. Guidelines for ToBI labelling. Technical report, Ohio State University, 1993. http://www.ling.ohio-state.edu/research/phonetics/E_ToBI.

[BFKM95] A. Bies, M. Ferguson, K. Katz, and R. MacIntyre. Bracketting guideliness for Treebank II style Penn treebank project. Technical Report MS-CIS-95-06, Linguistic Data Consortium, University of Pennsylvania, 1995.

[Bik04] D. M. Bikel. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* PhD thesis, University of Pennsylvania, 2004.

[BMW05] A. Bies, J. Mott, and C. Warner. *Addendum to the Switchboard Treebank Guidelines.* Linguistic Data Consortium, 2005.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001.

[Bri95] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[Bun92] W. Buntine. Tree classication software. In *Technology 2002: The Third National Technology Transfer Conference and Exposition*, Baltimore, 1992.

[Car98] J. Carroll, editor. *The Evaluation of Parsing Systems: Workshop at the 1st International Conference on Language Resources and Evaluation (LREC).* Number CSRP 489. Granada, Spain, 1998.

[CDD97] A. Cutler, D. Dahan, and W. A. Van Donselaar. Prosody in the comprehension of spoken language: A literature review. *Language and Speech*, 40(2):141–202, 1997.

[CFL$^+$02] J. Carroll, A. Frank, D. Lin, D. Prescher, and H. Uszkoreit, editors. *Proceedings of the Workshop 'Beyond PARSEVAL — Towards improved evaluation measures for parsing systems' at the 3rd International Conference on Language Resources and Evaluation (LREC).* Las Palmas, Gran Canaria, 2002.

[CH05] A. Cooper and J. Hale. Promotion of disfluency in syntactic parallelism. In *Proceedings of Disfluency in Spontaneous Speech*, 2005.

[Cha00] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of NAACL-00*, pages 132–139, 2000.

[CJ01] E. Charniak and M. Johnson. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 118–126, 2001.

[CJ05] E. Charniak and M. Johnson. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL-05*, pages 173–180, 2005.

[Col99] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

[DAR03] DARPA Information Processing Technology Office. Effective, affordable, reusable speech-to-text EARS. http://www.darpa.mil/ipto/programs/ears/, 2003.

[ECJ02] D. Engel, E. Charniak, and M. Johnson. Parsing and disfluency placement. In *Proceedings EMNLP*, pages 49–54, 2002.

[Fal04] Fall 2004 rich transcription (RT-04F) evaluation plan, version 14. Technical report, NIST, 2004. Available as http://www.nist.gov/speech/tests/rt/rt2004/fall/.

[Fis96] W. Fisher. *A C implementation of Daniel Kahn's theory of English syllable structure.* National Institute of Standards and Technology, 1996. ftp://jaguar.ncsl.nist.gov/pub/tsylb2-1.1.tar.Z.

[Fis01] J. Fiscus. SClite- score speech recognition system output. http://computing.ee.ethz.ch/sepp/sctk-1.2c-be/sclite.htm, 2001.

[GHM92] J. J. Godfrey, E. C. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, volume I, pages 517–520, San Francisco, 1992.

[Hin83] D. Hindle. Deterministic parsing of syntactic non-fluencies. In *Proceedings ACL*, pages 123–128, 1983.

[HO04] D. Hillard and M. Ostendorf. Scoring structural MDE: Towards more meaningful error rates. In *Proceedings of EARS RT-04 Workshop*, 2004.

[HRW$^+$05] R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Special Issue of the Journal of Natural Language Engineering on Parallel Texts*, June 2005.

[JC04] M. Johnson and E. Charniak. A TAG-based noisy channel model of speech repairs. In *Proceedings of ACL*, pages 33–39, 2004.

[JCL04] M. Johnson, E. Charniak, and M. Lease. An improved model for recognizing disfluencies in conversational speech. In *Proceedings of the Rich Text 2004 Fall Workshop*, 2004.

[JWG$^+$03] D. Jones, F. Wolf, E. Gibson, E. Williams, E. Fedorenko, D. Reynolds, and M. Zissman. Measuring the readability of automatic speech-to-text transcripts. In *Proceedings of Eurospeech*, 2003.

[Kah76] D. Kahn. *Syllable-based generalizations in English phonology.* PhD thesis, MIT (Published by Garland, New York in 1980), 1976.

[KLC$^+$05] J. G. Kahn, M. Lease, E. Charniak, M. Johnson, and M. Ostendorf. Effective use of prosody in parsing conversational speech. In *Proceedings of EMNLP-05*, 2005.

[KMP$^+$04] B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, G. Zweig, A. Stolcke, R. Gadde, W. Wang, J. Zheng, B. Chen, and Q. Zhu. The bicoastal IBM/SRI CTS STT system. In *Rich Transcription (RT-04F) Workshop*, 2004.

[KOC04] J. G. Kahn, M. Ostendorf, and C. Chelba. Parsing conversational speech using enhanced segmentation. In *HLT-NAACL 2004*, pages 125–128, 2004.

[Lev83] W. J. M. Levelt. Monitoring and self-repair in speech. *Cognitive Science*, 14:41–104, 1983.

[Lic96] R. J. Lickley. Juncture cues to disfluency. In *Proceedings ICSLP*, 1996.

[Lin98] D. Lin. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems, Granada, Spain*, 1998.

[Lin01] D. Lin. LaTaT: Language and Text Analysis Tools. In *Proceedings of the Human Language Technology Conference*, 2001.

[Liu04] Y. Liu. *Structural Event Detection for Rich Transcription of Speech.* PhD thesis, Purdue University, 2004.

[LSS03] Y. Liu, E. Shriberg, and A. Stolcke. Automatic disfluency identification in conversational speech using multiple knowledge sources. In *Proceedings of EUROSPEECH*, pages 957–960, 2003.

[LSS$^+$04a] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, B. Peskin, and M. Harper. The ICSI-SRI-UW metadata extraction system. In *Proceedings of ICSLP*, 2004.

[LSS$^+$04b] Y. Liu, E. Shriberg, A. Stolcke, B. Peskin, and M. Harper. The ICSI/SRI/UW RT-04 structural metadata extraction system. In *Proceedings of EARS RT-04 Workshop*, 2004.

[LSS$^+$05] Y. Liu, E. Shriberg, A. Stolcke, B. Peskin, J. Ang, D. Hillard, M. Ostendorf, M. Tomalin, P. Woodland, and M. Harper. Sructural metadata research in the EARS program. In *Proceedings of ICASSP*, 2005.

[LSSH05] Y. Liu, A. Stolcke, E. Shriberg, and M. Harper. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of ACL-05*, pages 451–458, 2005.

[Mai05] E. Maia. Finding consensus in probabilistic parsing. Master's thesis, School of Electrical and Computer Engineeing, Purdue University, 2005.

[McK98a] D. McKelvie. SDP – Spoken Dialog Parser. ESRC project on Robust Parsing and Part-of-speech Tagging of Transcribed Speech Corpora, May 1998.

[McK98b] D. McKelvie. The syntax of disfluency in spontaneous spoken language. ESRC project on Robust Parsing and Part-of-speech Tagging of Transcribed Speech Corpora, May 1998.

[MDK$^+$97] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proceedings of Eurospeech*, pages 1895–1898, Rhodes, Greece, 1997.

[MPR98] M. Mohri, F. C. N. Pereira, and M. Riley. A rational design for weighted finite-state transducer library. *Lecture notes in Computer Science*, 1998.

[MSM93] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[NBW$^+$02] E. Nöth, A. Batliner, V. Warnke, J-P. Haas, M. Boros, J. Buckow, R. Huber, F. Gallwitz, M. Nutt, and H. Niemann. On the use of prosody in automatic dialogue understanding. *Speech Communication*, 36(1–2):45–62, 2002.

[NV86] M. Nespor and I. Vogel. *Prosodic phonology.* Foris: Dordrecht, 1986.

[OSB03] M. Ostendorf, I. Shafran, and R. Bates. Prosody models for conversational speech recognition. In *Proceedings of the 2nd Plenary Meeting and Symposium on Prosody and Speech Processing, Invited Paper*, pages 147–154, 2003.

[OSSH$^+$01] M. Ostendorf, I. Shafran, S. Shattuck-Hufnagel, L. Carmichael, and W. Byrne. A prosodically labeled database of spontaneous speech. In *Proceedings of the ISCA Workshop on Prosody in Speech Recognition and Understanding*, pages 119–121, 10 2001.

[PBH94] J. Pitrelli, M. Beckman, and J. Hirschberg. Evaluation of prosodic transcription labeling reliability in the tobi framework. In *Proceedings of ICSLP*, volume 1, pages 123–126, 1994.

[PP96] J. Pynte and B. Prieur. Prosodic breaks and attachment decisions in sentence parsing. *Language and cognitive processes*, 11(1/2):165–191, 1996.

[Rat96] A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of Empirical Methods in Natural Language Processing Conference*, pages 133–141, 1996.

[Roa01] B. Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001.

[San90] B. Santorini. Part-of-speech tagging guidlines for the Penn Treebank Project (3rd revision). Technical report, University of Pennsylvania, 1990. Available as http://www.cis.upenn.edu/ treebank/home.html.

[SBB$^+$00] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. Ramana Rao Gadde, M. Plauché, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng. The SRI March 2000 Hub-5 conversational speech transcription system. In *NIST 2000 Speech Transcription Workshop*, 2000.

[SBP$^+$92] H. F. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirshberg. ToBI: A standard for labeling English prosody. In *Proceedings of ICSLP*, volume 2, pages 867–870, 1992.

[SBS97] E. Shriberg, R. Bates, and A. Stolcke. A prosody-only decision-tree model for disfluency detection. In *Proceedings of EUROSPEECH*, volume 5, pages 2383–2386, 1997.

[SC97] S. Sekine and M. J. Collins. The evalb software. http://cs.nyu.edu/cs/projects/proteus/evalb, 1997.

[Sel84] E. Selkirk. *Phonology and syntax: The relation between sound and structure.* MIT Press, Cambridge, 1984.

[Sel95] E. Selkirk. *The Handbook of Phonological Theory.* Blackwell, London, 1995.

[SFK$^+$05] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A . Stolcke. Modeling prosodic feature sequences for speaker recognition. *Speech Communication*, 46(3-4):455–472, 2005.

[Shi92] S. M. Shieber. *Constraint-based grammar formalisms: Parsing and type inference for natural and computer languages.* MIT Press, 1992.

[Sj01] K. Sjlander. *The Snack sound visualization module.* Royal Institute of Technology in Stockholm, 2001. http://www.speech.kth.se/SNACK.

[SO97] M. Swerts and M. Ostendorf. Prosodic and lexical indications of discourse structure in human-machine interactions. *Speech Communication*, 22(1):25–41, 1997.

[SSHTT00] E. Shriberg, A. Stolcke, D. Hakkani-Tur, and G. Tur. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, pages 127–154, 2000.

[SSHW98] K. Sonmez, E. Shriberg, L. Heck, and M. Weintraub. Modeling dynamic prosodic variation for speaker verification. In *Proceedings of ICSLP*, volume 7, pages 3189–3192, 1998.

[Ste00] M. Steedman. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 31:649–689, 2000.

[Str04] S. Strassel. *Simple Metadata Annotation Specification V6.2*, 2004.

[Tay95] A. Taylor. Revision of Meteer, Marie et al. dysfluency annotation stylebook for the Switchboard corpus. Technical report, Linguistic Data Consortium, University of Pennsylvania, 1995.

[Tay96] A. Taylor. *Bracketing Switchboard: An Addendum to the Treebank II Bracketing Guidelines*. Linguistic Data Consortium, 1996.

[VO93] N. M. Veilleux and M. Ostendorf. Prosody/parse scoring and its application in ATIS. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 335–340, Princeton, NJ, 1993.

[Wan03] W. Wang. *Statistical Parsing and Language Modeling Based on Constraint Dependency Grammar*. PhD thesis, Purdue University, 2003. School of Electrical and Computer Engineering, West Lafayette, IN.

[WG04] D. Watson and E. Gibson. The relationship between intonational phrasing and syntactic structure in language production. *Language and Cognitive Processes*, 19:713–755, 2004.

[WH02] W. Wang and M. Harper. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of EMNLP*, pages 238–247, 2002.

[WH04] W. Wang and M. P. Harper. A statistical constraint dependency grammar CDG parser. In *Proceedings of the ACL-04 Workshop on Incremental Parsing*, 2004.

[WOK05] D. Wong, M. Ostendorf, and J. G. Kahn. Using weakly supervised learning to improve prosody labeling. Technical Report UWEETR-2005-0003, University of Washington Electrical Engineering Dept., 2005.

[WSH04] W. Wang, A. Stolcke, and M. Harper. The use of a linguistically motivated language model in conversational speech recognition. In *Proceedings of ICASSP*, 2004.