Parsing Conversational Speech Using Enhanced Segmentation

Jeremy G. Kahn and **Mari Ostendorf**
SSLI, University of Washington, EE
{jgk,mo}@ssli.ee.washington.edu

Ciprian Chelba
Microsoft Research
chelba@microsoft.com

Abstract

The lack of sentence boundaries and presence of disfluencies pose difficulties for parsing conversational speech. This work investigates the effects of automatically detecting these phenomena on a probabilistic parser's performance. We demonstrate that a state-of-the-art segmenter, relative to a pause-based segmenter, gives more than 45% of the possible error reduction in parser performance, and that presentation of interruption points to the parser improves performance over using sentence boundaries alone.

1 Introduction

Parsing speech can be useful for a number of tasks, including information extraction and question answering from audio transcripts. However, parsing conversational speech presents a different set of challenges than parsing text: sentence boundaries are not well-defined, punctuation is absent, and disfluencies (edits and restarts) impact the structure of language.

Several efforts have looked at detecting sentence boundaries in speech, e.g. (Kim and Woodland, 2001; Huang and Zweig, 2002). Metadata extraction efforts, like (Liu et al., 2003), extend this task to include identifying self-interruption points (IPs) that indicate a disfluency or restart. This paper explores the usefulness of identifying boundaries of sentence-like units (referred to as SUs) and IPs in parsing conversational speech.

Early work in parsing conversational speech was rule-based and limited in domain (Mayfield et al., 1995). Results from another rule-based system (Core and Schubert, 1999) suggests that standard parsers can be used to identify speech repairs in conversational speech. Work in statistically parsing conversational speech (Charniak and Johnson, 2001) has examined the performance of a parser that removes edit regions in an earlier step. In contrast, we train a parser on the complete (human-specified) segmentation, with edit-regions included. We choose to work with all of the words within edit regions anticipating that making the parallel syntactic structures of the edit region available to the parser can improve its performance in identifying that structure. Our work makes use of the Structured Language Model (SLM) as a parser and an existing SU-IP detection algorithm, described next.

2 Background

2.1 Structured Language Model

The SLM assigns a probability $P(W,T)$ to every sentence W and its every possible binary parse T. The terminals of T are the words of W with POS tags, and the nodes of T are annotated with phrase headwords and non-terminal labels. Let W be a sentence of length n words with added sentence boundary markers w_0 =<s> and w_{n+1} =</s>. Let $W_k = w_0 \ldots w_k$ be the word k-prefix of the sentence — the words from the beginning of the sentence up to the current position k — and W_kT_k the *word-parse k-prefix*. Figure 1 shows a word-parse k-prefix; h_0 .. h_{-m} are the *exposed heads*, each head being a pair (headword, non-terminal label), or (word, POS tag) in the case of a root-only tree. The exposed heads at a given position k in the input sentence are a function of the word-parse k-prefix.

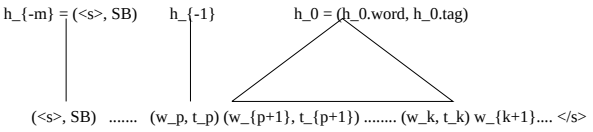

Figure 1: A word-parse k-prefix

The joint probability $P(W,T)$ of a word sequence W and a complete parse T can be broken into:

$$P(W,T) = \prod_{k=1}^{n+1} [P(w_k|W_{k-1}T_{k-1}) \cdot P(t_k|W_{k-1}T_{k-1},w_k) \cdot \prod_{i=1}^{N_k} P(p_i^k|W_{k-1}T_{k-1},w_k,t_k,p_1^k \cdots p_{i-1}^k)] \quad (1)$$

where:
- $W_{k-1}T_{k-1}$ is the word-parse $(k-1)$-prefix
- w_k is the word predicted by the WORD-PREDICTOR
- t_k is the tag assigned to w_k by the TAGGER
- $N_k - 1$ is the number of operations the CONSTRUCTOR executes at sentence position k before passing control to the WORD-PREDICTOR (the N_k-th operation at position k is the null transition); N_k is a function of T
- p_i^k denotes the i-th CONSTRUCTOR operation carried out at position k in the word string; the operations performed by the CONSTRUCTOR are illustrated in Figures 2-3 and they ensure that all possible binary branching parses, with all possible headword and non-terminal label assignments for the $w_1 \ldots w_k$ word sequence, can

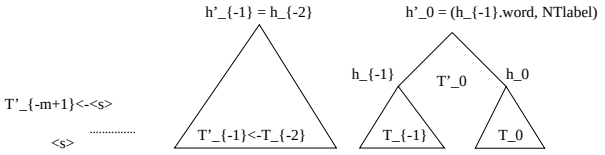

Figure 2: Result of adjoin-left under NT label

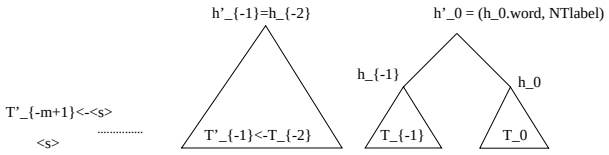

Figure 3: Result of adjoin-right under NT label

be generated. The $p_1^k \ldots p_{N_k}^k$ sequence of CONSTRUCTOR operations at position k grows the word-parse $(k-1)$-prefix into a word-parse k-prefix.

The SLM is based on three probabilities, each estimated using deleted interpolation and parameterized (approximated) as follows:

$$P(w_k|W_{k-1}T_{k-1}) = P(w_k|h_0,h_{-1}), \quad (2)$$

$$P(t_k|w_k,W_{k-1}T_{k-1}) = P(t_k|w_k,h_0,h_{-1}), \quad (3)$$

$$P(p_i^k|W_kT_k) = P(p_i^k|h_0,h_{-1}). \quad (4)$$

Since the number of parses for a given word prefix W_k grows exponentially with k, $|\{T_k\}| \sim O(2^k)$, the state space of our model is huge even for relatively short sentences, so the search strategy uses pruning.

Each model component is initialized from a set of parsed sentences after undergoing headword percolation and binarization. The position of the headword within a constituent is specified using a rule-based approach. Assuming the index of the headword on the right-hand side of the rule is k, we binarize the constituent by following one of the two binarization schemes in Figure 4. Intermediate nodes created receive the label Z'[1]. The choice between the two schemes is made according to the identity of the Z label on the left-hand-side of a rewrite rule.

An N-best EM variant is employed to jointly reestimate the model parameters so that perplexity on training data is decreased, i.e. increasing likelihood. Experimentally, the reduction in perplexity carries over to the test set.

[1]Any resemblance to X-bar theory is purely coincidental.

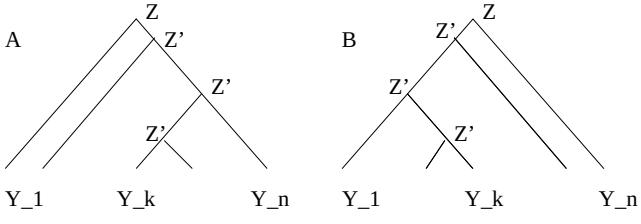

Figure 4: Binarization schemes

The SLM can be used for parsing either as a generative model $P(W,T)$ or as a conditional model $P(T|W)$. In the latter case, the $P(w_k|W_{k-1}T_{k-1})$ prediction is omitted in Eq. (1). For further details on the SLM, see (Chelba and Jelinek, 2000).

2.2 SU and IP Detection

The system used here for SU and IP detection is (Kim et al., 2004), modulo differences in training data. It combines decision tree models of prosody with a hidden event language model in a hidden Markov model (HMM) framework for detecting events at each word boundary, similar to (Liu et al., 2003). Differences include the use of lexical pattern matching features (sequential matching words or POS tags) as well as prosody cues in the decision tree, and having a joint representation of SU and IP boundary events rather than separate detectors. On the DARPA RT-03F metadata test set (NIST, 2003), the model has 35.0% slot error rate (SER) for SUs (75.7% recall, 87.7% precision), and 68.8% SER for edit IPs (41.8% recall, 79.8% precision) on reference transcripts, using the `rt_eval` scoring tool.[2] While these error rates are relatively high, it is a difficult task and the SU performance is at the state of the art.

Since early work on "sentence" segmentation simply looked at pause duration, we designed a decision tree classifier to predict SU events based only on the pause duration after a word boundary. This model served as a baseline condition, referred to here as the "naïve" predictor since it makes no use of other prosodic or lexical cues that are important for preventing IPs or hesitations from triggering false SU detection. The naïve predictor has SU SER of 68.8%, roughly twice that of the HMM, with a large loss in recall (43.2% recall, 79.0% precision).

3 Corpus

The data used in this work is the treebank (TB3) portion of the Switchboard corpus of conversational telephone speech, which includes sentence boundaries as well as the reparandum and interruption point of disfluencies. The data consists of 816 hand-transcribed conversation sides (566K words), of which we reserve 128 conversation sides (61K words) for evaluation testing according to the 1993 NIST evaluation choices.

We use a subset of Switchboard data – hand-annotated for SUs and IPs – for training the SU/IP boundary event detector, and for providing the oracle versions of these events as a control in our experiments. The annotation conventions for this data, referred to as V5 (Strassel, 2003), are slightly different from that used in the TB3 an-

[2]Note that the IP performance figures are not comparable to those in the DARPA evaluation, since we restrict the focus to IPs associated with edit disfluencies.

notations in a few important ways. Notably for this work, V5 annotates IPs for both conversational fillers (such as filled pauses and discourse markers) and self-edit disfluencies, while TB3 represents only edit-related IPs. This difference is addressed by explicitly distinguishing between these types in the IP detection model. In addition, the V5 conventions define an SU as including only one independent main clause, so the size of the "segments" available for parsing is sometimes smaller than in TB3. Further, the SU boundaries were determined by annotators who actually listened to the speech signal, vs. annotated from text alone as for TB3. One consequence of the differences is a small amount of additional error due to train/test mismatch. More importantly, the "ground truth" for the syntactic structure must be mapped to the SU segmentation, both for training and test.

In many cases, the original syntactic constituents span multiple SUs, but we follow a simple rule in generating this new SU-based truth: only those constituents contained entirely within an SU would be retained. In practice, this means eliminating a few high-level constituents. The effect is usually to change the interpretation of some sentence-level conjunctions to be discourse markers, rather than conjoining two main clauses. This change is arguably an improvement, since the SU annotation relies on a human annotation that takes into consideration acoustic information (not only the words).

4 Experiments

In all experiments, the SLM parser was trained on the baseline truth Switchboard corpus described above, with hand-annotated SUs and optionally IPs. For testing, the system was presented with conversation sides segmented according to the various SU-predictions, and evaluated on its performance in predicting the true syntactic structure.

4.1 Experimental Variables

We seek to explore how much impact current metadata detection algorithms have over the naïve pause-based segmentation. To this end, we test along two experimental dimensions: SU segmentation and IP detection.

Some type of **segmentation** is critical to most parsers. In the SU dimension, we tested three conditions. Across these conditions, the parser training was held constant, but the test segmentation varied across three cases: (i) *oracle*, hand-labeled SU segmentation; (ii) *automatic*, SU segmentation from the automatic detection system using both prosody and lexical cues (Kim et al., 2004); and (iii) *naïve*, SU segmentation from a decision tree predictor using only pause duration cues. The SUs are included as words, similar to sentence boundaries in prior SLM work. By varying the SU segmentation of the test data for our system, we gain insight into how the performance of SU detection changes the overall accuracy of the parser.

We expect **interruption points** to be useful to parsing, since edit points often indicate a restart point, and the preceding syntactic phrase should attach to the tree differently. In the IP dimension, we examined two conditions (present and absent). For each condition, we retrained the parser including hand-labeled IPs, since the vocabulary of available "words" is different when the IP is included as an input token. The two IP conditions are: (a) *No IP*, training the parser on syntax that did not include IPs as words, and testing on segmented input that also did not include IP tokens; and (b) *IP*, training and testing on input that includes IPs as words. The incorporation of IPs as words may not be ideal, since it reduces the number of true words available to an N-gram model at a given point, but it has the advantages of simplicity and consistency with SU treatment. Because the naïve system does not predict IPs, we only have experiments for 5 of the 6 possible combinations.

4.2 Evaluation

We evaluated parser performance by using bracket precision and recall scores, as well as bracket-crossing, using the parseval metric (Sekine and Collins, 1997; Black et al., 1991). This bracket-counting metric for parsers, requires that the input words (and, by implication, sentences) be held constant across test conditions. Since our experiments deliberately vary the segmentation, we needed to evaluate each conversation side as a single "sentence" in order to obtain meaningful results across different segmentations. We construct this top-level sentence by attaching the parser's proposed constituents for each SU to a new top-level constituent (labeled TIPTOP). Thus, we can compare two different segmentations of the same data, because it ensures that the segmentations will agree at least at the beginning and end of the conversation. Segmentation errors will of course cause some mismatches, but that possibility is what we are investigating.

For evaluation, we ignore the TIPTOP bracket (which always contains the entire conversation side), so this technique does not interfere with accurate bracket counting, but allows segmentation errors to be evaluated at the level of bracket-counting. The SLM parser uses binary trees, but the syntactic structures we are given as truth often branch in N-ary ways, where $N > 2$. The parse trees used for training the SLM use bar-level nodes to transform N-ary trees into binary ones; the reverse mapping of SLM-produced binary trees back to N-ary trees is done by simply removing the bar-level constituents. Finally, to compare the IP-present conditions with the non-IP conditions, we ignore IP tokens when counting brackets.

4.3 Results

Table 1 shows the parser performance: average bracket-crossing (lower is better), precision and recall (higher is

Avg. crossing	Oracle+P	Oracle	Auto	Naïve
No IP	35.88	46.80	66.21	80.40
IP	35.03	44.56	58.09	—
%Recall	Oracle+P	Oracle	Auto	Naïve
No IP	69.91	77.45	72.47	68.05
IP	70.36	77.98	74.25	—
%Precision	Oracle+P	Oracle	Auto	Naïve
No IP	79.30	68.40	63.09	58.67
IP	79.65	68.42	64.46	—

Table 1: Bracket crossing, precision and recall results.

better). The number of bracket-crossings per "sentence" is quite high, due to evaluating all text from a given conversation side into one "TIPTOP sentence". Precision and recall are regarding the bracketing of all the tokens under consideration (i.e., not including bar-level brackets, and not including IP token labeling). All differences are highly significant ($p < 0.005$ according to a sign test at conversation level) except for comparing oracle results with and without IPs.

We find that the HMM-based SU detection system achieves a 7% improvement in precision and recall over the naïve pause-based system, and an 18% reduction in average bracket crossing. Further, the use of IPs as input tokens improves parser performance, especially when the segmentation is imperfect. While segmentation has an impact on parsing, it is not the limiting factor: the best possible bracketing respecting the automatic segmentation has a 96.50% recall and 99.35% precision.

Adding punctuation to the oracle case (Oracle+P) improves performance, as seen more clearly with the F-measure because of changes in the precision-recall balance. The F-measures goes from 72.6 for oracle/no-IP to 74.3 for oracle+P/no-IP to 74.7 for oracle+P/IP. The fact that punctuation is useful on top of the oracle segmentation suggests that a richer representation of structural metadata would be beneficial. The reasons why IPs do not have much of an impact in the oracle case are not clear – it could be a modeling issue or it could be simply that the IPs add robustness to the automatic segmentation.

5 Discussion

In comparison to the naïve pause-based SU detector, using an SU detector based on prosody and lexical cues gives us more than 45% of the possible gain to the best possible (oracle) case, despite a relatively high SU error rate. We hypothesize that low SU and IP recall in the naïve segmenter created a much larger parse search space, leading to more opportunities for errors. The improvement is promising, and suggests that research to improve metadata extraction can have a direct impact on the performance of other natural language applications that deal with conversational speech.

The use of SUs and IPs as input words may result in a loss of information, reducing the "true" word history available to the parser component models. Further research using the structured language model could incorporate these metadata directly into the model, allowing it to take advantage of higher-level metadata without reducing the effective number of words available to the model. In addition, just as the SLM is useful for both parsing and language modeling, it could be used to predict metadata for its own sake or to improve word recognition, with or without the word-based representation.

Acknowledgments

We thank J. Kim for providing the SU-IP detection results, using tools developed under DARPA grant MDA904-02-C-0437. This work is supported by NSF grant no. IIS085940. Any opinions or conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of these agencies.

References

E. Black et al. 1991. A procedure for quantitatively comparing syntactic coverage of English grammars. In *Proc. 4th DARPA Speech & Natural Lang. Workshop*, pages 306–311.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proc. 2nd NAACL*, pages 118–126.

C. Chelba and F. Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332, October.

M. Core and K. Schubert. 1999. Speech repairs: A parsing perspective. In *Satellite Meeting ICPHS 99*.

J. Huang and G. Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proc. Eurospeech*.

J.-H. Kim and P. Woodland. 2001. The use of prosody in a combined system for punctuation generation and speech recognition. In *Proc. Eurospeech*, pages 2757–2760.

J. Kim, S. E. Schwarm, and M. Ostendorf. 2004. Detecting structural metadata with decision trees and transformation-based learning. In *Proc. HLT-NAACL*.

Y. Liu, E. Shriberg, and A. Stolcke. 2003. Automatic disfluency identification in conversational speech using multiple knowledge sources. In *Proc. Eurospeech*, volume 1, pages 957–960.

L. Mayfield *et al.* 1995. Parsing real input in JANUS: a concept-based approach. In *Proc. TMI 95*.

NIST. 2003. Rich Transcription Fall 2003 Evaluation Results. `http://www.nist.gov/speech/tests/rt/rt2003/fall/`.

S. Sekine and M. Collins. 1997. EVALB. As in Collins ACL 1997; `http://nlp.cs.nyu.edu/evalb/`.

S. Strassel, 2003. *Simple Metadata Annotation Specification V5.0*. Linguistic Data Consortium.