Parsing Speech: A Neural Approach to Integrating Lexical and Acoustic-Prosodic Information

Trang Tran[*1], Shubham Toshniwal[*2], Mohit Bansal[3], Kevin Gimpel[2], Karen Livescu[2], Mari Ostendorf[1]

[1]Electrical Engineering, University of Washington
[2]Toyota Technological Institute at Chicago
[3]Department of Computer Science, UNC Chapel Hill

{ttmt001, ostendor}@uw.edu, mbansal@cs.unc.edu,
{shtoshni, kgimpel, klivescu}@ttic.edu

Abstract

In conversational speech, the acoustic signal provides cues that help listeners disambiguate difficult parses. For automatically parsing spoken utterances, we introduce a model that integrates transcribed text and acoustic-prosodic features using a convolutional neural network over energy and pitch trajectories coupled with an attention-based recurrent neural network that accepts text and prosodic features. We find that different types of acoustic-prosodic features are individually helpful, and together give statistically significant improvements in parse and disfluency detection F1 scores over a strong text-only baseline. For this study with known sentence boundaries, error analyses show that the main benefit of acoustic-prosodic features is in sentences with disfluencies, attachment decisions are most improved, and transcription errors obscure gains from prosody.

1 Introduction

While parsing has become a relatively mature technology for written text, parser performance on conversational speech lags behind. Speech poses challenges for parsing: transcripts may contain errors and lack punctuation; even perfect transcripts can be difficult to handle because of disfluencies (restarts, repetitions, and self-corrections), filled pauses ("um", "uh"), interjections ("like"), parentheticals ("you know", "I mean"), and sentence fragments. Some of these phenomena can be handled in standard grammars, but disfluencies typically require extensions of the model. Different approaches have been explored in both constituency parsing (Charniak and Johnson, 2001; Johnson and Charniak, 2004) and dependency parsing (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014).

Despite these challenges, speech carries helpful extra information – beyond the words – associated with the prosodic structure of an utterance and encoded via variation in timing and intonation. Speakers pause in locations that are correlated with syntactic structure (Grosjean et al., 1979), and listeners use prosodic structure in resolving syntactic ambiguities (Price et al., 1991). Prosodic cues also signal disfluencies by marking the interruption point (Shriberg, 1994). However, most speech parsing systems in practice take little advantage of these cues. Our study focuses on this last challenge, aiming to incorporate prosodic cues in a neural parser, handling disfluencies as constituents via a neural attention mechanism.

A challenge of incorporating prosody in parsing is that multiple acoustic cues interact to signal prosodic structure, including pauses, lengthening, fundamental frequency modulation, and spectral shape. These cues also vary with the phonetic segment, emphasis, emotion and speaker, so feature extraction typically involves multiple time windows and normalization techniques. The most successful constituent parsers have mapped these features to prosodic boundary posteriors by using labeled training data (Kahn et al., 2005; Hale et al., 2006; Dreyer and Shafran, 2007). The approach proposed here takes advantage of advances in neural networks to automatically learn a good feature representation without the need to explicitly represent prosodic constituents. To narrow the scope of this work and facilitate error analysis, our experiments use known transcripts and sentence segmentation.

Our work offers the following contributions. We introduce a framework for directly integrating acoustic-prosodic features with text in a neural encoder-decoder parser that does not require hand-annotated prosodic structure. We demonstrate improvements in constituent parsing of conversational

[*]Equal Contribution.

Proceedings of NAACL-HLT 2018, pages 69–81
New Orleans, Louisiana, June 1 - 6, 2018.

speech over a high-quality text-only parser and provide analyses showing where prosodic features help and that assessment of their utility is affected by human transcription errors.

2 Task and Model Description

Our model maps a sequence of word-level input features to a linearized parse output sequence. The word-level input feature vector consists of the concatenation of (learnable) word embeddings $\boldsymbol{e_i}$ and several types of acoustic-prosodic features, described in Section 2.3.

2.1 Task Setup

We assume the availability of a training treebank of conversational speech (in our case, Switchboard-NXT (Calhoun et al., 2010)) and corresponding constituent parses. The transcriptions are preprocessed by removing punctuation and lower-casing all text to better mimic the speech recognition setting. Following Vinyals et al. (2015), the parse trees are linearized, and pre-terminals are normalized as "XX" (see Appendix A.1).

2.2 Encoder-Decoder Parser

Our attention-based encoder-decoder model is similar to the one used by Vinyals et al. (2015). The *encoder* is a deep long short-term memory recurrent neural network (LSTM-RNN) (Hochreiter and Schmidhuber, 1997) that reads in a word-level inputs,[1] represented as a sequence of vectors $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{T_s})$, and outputs high-level features $\boldsymbol{h} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{T_s})$ where $\boldsymbol{h}_i = \text{LSTM}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1})$.[2]

The *parse decoder* is also a deep LSTM-RNN that predicts the linearized parse sequence $\boldsymbol{y} = (y_1, \cdots, y_{T_o})$ as follows:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{T_o} P(y_t|\boldsymbol{h}, \boldsymbol{y_{<t}})$$

In attention-based models, the posterior distribution of the output y_t at time step t is given by:

$$P(y_t|\boldsymbol{h}, \boldsymbol{y_{<t}}) = \text{softmax}(\boldsymbol{W}_s[\boldsymbol{c}_t; \boldsymbol{d}_t] + \boldsymbol{b}_s),$$

where vector $\boldsymbol{b}_s$ and matrix $\boldsymbol{W}_s$ are learnable parameters; $\boldsymbol{c}_t$ is referred to as a *context vector* that summarizes the encoder's output $\boldsymbol{h}$; and $\boldsymbol{d}_t$ is the decoder hidden state at time step t, which captures the previous output sequence context $\boldsymbol{y_{<t}}$.

$$u_{it} = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_1\boldsymbol{h}_i + \boldsymbol{W}_2\boldsymbol{d}_t + \boldsymbol{b}_a)$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{u}_t) \qquad \boldsymbol{c}_t = \sum_{i=1}^{T_s} \alpha_{ti}\boldsymbol{h}_i$$

where vectors $\boldsymbol{v}$, $\boldsymbol{b}_a$ and matrices $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ are learnable parameters; $\boldsymbol{u}_t$ and $\boldsymbol{\alpha}_t$ are the attention score and attention weight vector, respectively, for decoder time step t.

The above attention mechanism is only *content*-based, i.e., it is only dependent on $\boldsymbol{h}_i$, $\boldsymbol{d}_t$. It is not *location*-aware, i.e., it does not consider the "location" of the previous attention vector. For parsing conversational text, location awareness is beneficial since disfluent structures can have duplicate words/phrases that may "confuse" the attention mechanism.

In order to make the model location-aware, the attention mechanism takes into account the previous attention weight vector $\boldsymbol{\alpha}_{t-1}$. In particular, we use the attention mechanism proposed by Chorowski et al. (2015), in which $\boldsymbol{\alpha}_{t-1}$ is represented via a feature vector $\boldsymbol{f}_t = \boldsymbol{F} * \boldsymbol{\alpha}_{t-1}$, where $\boldsymbol{F} \in \mathcal{R}^{k \times r}$ represents k learnable convolution filters of width r. The filters are used for performing 1-D convolution over $\boldsymbol{\alpha}_{t-1}$ to extract k features $\boldsymbol{f}_{ti}$ for each time step i of the input sequence. The extracted features are then incorporated in the alignment score calculation as:

$$u_{it} = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_1\boldsymbol{h}_i + \boldsymbol{W}_2\boldsymbol{d}_t + \boldsymbol{W}_f\boldsymbol{f}_{ti} + \boldsymbol{b}_a)$$

where $\boldsymbol{W}_f$ is another learnable parameter matrix. Finally, the decoder state $\boldsymbol{d}_t$ is computed as $\boldsymbol{d}_t = \text{LSTM}([\tilde{\boldsymbol{y}}_{t-1}; \boldsymbol{c}_{t-1}], \boldsymbol{d}_{t-1})$, where $\tilde{\boldsymbol{y}}_{t-1}$ is the embedding vector corresponding to the previous output symbol y_{t-1}. As we will see in Sec. 4.1, the location-aware attention mechanism is especially useful for handling disfluencies.

2.3 Acoustic-Prosodic Features

In previous work using encoder-decoder models for parsing (Vinyals et al., 2015; Luong et al., 2016), vector $\boldsymbol{x}_i$ is simply the word embedding $\boldsymbol{e}_i$ of the word at position i of the input sentence. For parsing conversational speech, we can incorporate acoustic-prosodic features. Here we explore four types of features widely used in computational models of prosody: pauses, duration lengthening, fundamental frequency, and energy. Since prosodic cues are

[1] As in Vinyals et al. (2015) the input sequence is processed in reverse order, as shown in Figure 1.

[2] For brevity we omit the LSTM equations. The details can be found, e.g., in Zaremba et al. (2014).

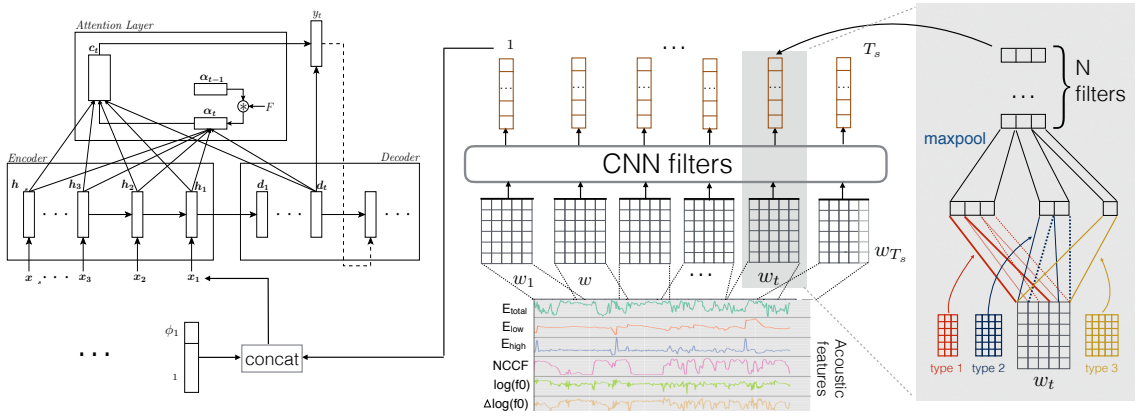

Figure 1: Left – An attention-based encoder-decoder reading the input $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{T_s}$, where $\boldsymbol{x}_i = [\boldsymbol{e}_i \ \boldsymbol{\phi}_i \ \boldsymbol{s}_i]$ is composed of word embeddings $\boldsymbol{e}_i$, prosodic features $\boldsymbol{\phi}_i$, and learned (CNN-based) features $\boldsymbol{s}_i$. The encoder reads the input in *reverse* order and the decoder outputs the linearized parse $y_1, \cdots, y_t, \cdots$. Right – Detailed illustration of acoustic-prosodic feature learning module. CNN features are computed from input energy and pitch features; here the CNN filter parameters are $m = 3$ and $w = [3, 4, 5]$.

at sub- and multi-word time scales, they are integrated with the encoder-decoder using different mechanisms.

All features are extracted from transcriptions that are time-aligned at the word level.[3] We use time alignments associated with the corpus to be consistent with other studies. In a small number of cases, the time alignment for a particular word boundary is missing. Some cases are due to tokenization. For example, contractions, such as *don't* in the original transcript, are treated as separated words for the parser (*do* and *n't*), and the internal word boundary time is missing. In such cases, these internal times are estimated. In other cases, there are transcription mismatches that lead to missing time alignments, where we cannot estimate times. For the roughly 1% of sentences where time alignments are missing, we simply back off to the text-based parser.

Pause. The pause feature vector $\boldsymbol{p}_i$ for word i is the concatenation of pre-word pause feature $\boldsymbol{p}_{pre,i}$ and post-word pause feature $\boldsymbol{p}_{post,i}$, where each subvector is a learned embedding for 6 pause categories: no pause, missing, $0 < p \leq 0.05$ s, 0.05 s $< p \leq 0.2$ s, $0.2 < p \leq 1$ s, and $p > 1$ s (including turn boundaries). The bins are chosen based on the observed distribution (see Appendix A.1). We did not use (real-valued) pause duration directly, for two main reasons: (1) to handle missing time alignments; and (2) duration of pause does not matter beyond a threshold (e.g. $p > 1$ s).

Word duration. Both word duration and word-final duration lengthening are strong cues to prosodic phrase boundaries (Wightman et al., 1992; Pate and Goldwater, 2013). The word duration feature δ_i is computed as the actual word duration divided by the mean duration of the word, clipped to a maximum value of 5. The sample mean is used for frequent words (count ≥ 15). For infrequent words we estimate the mean as the sum over the sample means for the phonemes in the word's dictionary pronunciation. We refer to the manually defined prosodic feature pair of $\boldsymbol{p}_i$ and δ_i as ϕ_i.

Fundament frequency (f0) and Energy (E) contours (f0/E). We use a CNN to automatically learn the mapping from the time series of f0/E features to a word-level vector. The contour features are extracted from 25-ms frames with 10-ms hops using Kaldi (Povey et al., 2011). Three f0 features are used: warped Normalized Cross Correlation Function (NCCF), log-pitch with Probability of Voicing (POV)-weighted mean subtraction over a 1.5-second window, and the estimated derivative (delta) of the raw log pitch. Three energy features are extracted from the Kaldi 40-mel-frequency filter bank features: E_{total}, the log of total energy normalized by dividing by the speaker side's max total energy; E_{low}, the log of total energy in the lower 20 mel-frequency bands, normalized by total energy, and E_{high}, the log of total energy in the higher 20 mel-frequency bands, normalized by total energy. Multi-band energy features are used as a

[3] The assumption of known word alignments is standard for prosodic feature extraction in many spoken language processing studies. Time alignments can be obtained as a by-product of recognition or from forced alignment.

simple mechanism to capture articulatory strengthening at prosodic constituent onsets (Fourgeron and Keating, 1997).

Figure 1 summarizes the feature learning approach. The f0 and E features are processed at the word level: each sequence of frames corresponding to a time-aligned word (and potentially its surrounding context) is convolved with N filters of m sizes (a total of mN filters). The motivation for the multiple filter sizes is to enable the computation of features that capture information on different time scales. For each filter, we perform a 1-D convolution over the 6-dimensional f0/E features with a stride of 1. Each filter output is max-pooled, resulting in mN-dimensional speech features $\boldsymbol{s}_i$. Our overall acoustic-prosodic feature vector is the concatenation of $\boldsymbol{p}_i$, δ_i, and $\boldsymbol{s}_i$ in various combinations.

3 Experiments

3.1 Dataset

Our core corpus is Switchboard-NXT (Calhoun et al., 2010), a subset of the Switchboard corpus (Godfrey and Holliman, 1993): 2,400 telephone conversations between strangers; 642 of these were hand-annotated with syntactic parses and further augmented with richer layers of annotation facilitated by the NITE XML toolkit (Calhoun et al., 2010). Our sentence segmentations and syntactic trees are based on the annotations from the Treebank set, with a few manual corrections from the NXT release. This core dataset consists of 100K sentences, totaling 830K tokens forming a vocabulary of 13.5K words. We use the time alignments available from NXT, which is based on a corrected word transcript that occasionally differs from the Treebank, leading to some missing time alignments. We follow the sentence boundaries defined by the parsed data available,[4] and the data split (90% train; 5% dev; 5% test) defined by related work done on Switchboard (Charniak and Johnson, 2001; Kahn et al., 2005; Honnibal and Johnson, 2014).

3.2 Evaluation Metrics and Baselines

The standard evaluation metric for constituent parsing is the *parseval* metric which uses bracketing precision, recall, and F1, as in the canonical implementation of EVALB.[5] For written text, punctuation is sometimes represented as part of the sequence and impacts the final score, but for speech the punctuation is not explicitly available so it does not contribute to the score. Another challenge of transcribed speech is the presence of disfluencies. Speech repairs are indicated under "EDITED" nodes in Switchboard parse trees, which include structure under these nodes that is not of interest for simple text clean-up. Therefore, some studies report *flattened-edit parseval* F1 scores ("flat-F1"), which is *parseval* computed on trees where the structure under edit nodes has been eliminated so that all leaves are immediate children. We report both scores for the baseline text-only model showing that the differences are small, then use the standard *parseval* F1 score for most results.[6]

Disfluencies are particularly problematic for statistical parsers, as explained by Charniak and Johnson (2001), and some systems incorporate a separate disfluency detection stage. For this reason, and because it is useful for understanding system performance, most studies also report disfluency detection performance, which is measured in terms of the F1 score for detecting whether a word is in an edit region. Our approach does not involve a separate disfluency detection stage, but identifies disfluencies implicitly via the parse structure. Consequently, the disfluency detection results are not competitive with work that directly optimize for disfluency detection. We report disfluency detection scores primarily as a diagnostic.

Most previous work on integrating prosody and parsing has used the Switchboard corpus, but it is still difficult to compare results because of differences in constraints, objectives and the use of constituent vs. dependency structure, as discussed further in Section 6. The most relevant prior studies (on constituent parsing) that we compare to are a bit old. The text-only result from our neural parser represents a stronger baseline and is important for decoupling the impact of prosody vs. the parsing framework.

3.3 Model Training and Inference

Both the encoder and decoder are 3-layer deep LSTM-RNNs with 256 hidden units in each layer. For the location-aware attention, the convolution operation uses 5 filters of width 40 each. We use 512-dimensional embedding vectors to repre-

[4] Note that these sentence units can be inconsistent with other layers of Switchboard annotations, such as *slash units*.

[5] http://nlp.cs.nyu.edu/evalb/

[6] A variant of the "flat-F1" score is used in (Charniak and Johnson, 2001; Kahn et al., 2005), which uses a relaxed edited node precision and recall but also ignores filled pauses.

sent words and linearized parsing symbols, such as "(S".[7]

A number of configurations are explored for the acoustic-prosodic features, tuning based on dev set parsing performance. Pause embeddings are tuned over $\{4, 16, 32\}$ dimensions. For the CNN, we try different configurations of filter widths $w \in \{[10, 25, 50], [5, 10, 25, 50]\}$ and number of filters $N \in \{16, 32, 64, 128\}$ for each filter width.[8] These filter size combinations are chosen to capture f0 and energy phenomena on various levels: $w = 5, 10$ for sub-word, $w = 25$ for word, and $w = 50$ for word and extended context. Our best model uses 32-dimensional pause embeddings and $N = 32$ filters of widths $w = [5, 10, 25, 50]$, which corresponds to $m = 4$ and 128 filters.

For optimization we use Adam (Kingma and Ba, 2014) with a minibatch size of 64. The initial learning rate is 0.001 which is decayed by a factor of 0.9 whenever training loss, calculated after every 500 updates, degrades relative to the worst of its previous 3 values. All models are trained for up to 50 epochs with early stopping. For regularization, dropout with 0.3 probability is applied on the output of all LSTM layers (Pham et al., 2014).

For inference, we use a greedy decoder to generate the linearized parse. The output token with maximum posterior probability is chosen at every time step and fed as input in the next time step. The decoder stops upon producing the end-of-sentence symbol. We use TensorFlow (Abadi et al., 2015) to implement all models.[9]

4 Results

4.1 Text-only Results

Model	F1	flat-F1	fluent	disf
Berkeley	85.41	85.91	90.52	83.08
C-attn	83.33	83.20	90.86	79.94
CL-attn	**87.85**	**87.68**	**92.07**	**85.95**

Table 1: Scores of text-only models on the dev set: 2044 fluent and 3725 disfluent sentences. C-attn denotes *content*-only attention; CL-attn denotes *content+location* attention.

[7]The number of layers, dimension of hidden units, dimension of embedding, and convolutional attention filter parameters of the text-only parser were explored in earlier experiments on the development set and then fixed as described.

[8]Note that a filter of width 10 has size 6×10, since the features are of dimension 6.

[9]Our code resources can be found in Appendix A.1.

Model	Parse	Disf
Berkeley (text only)	85.41	62.45
CL-attn (text only)	87.85	79.50
CL-attn text and		
+ p	88.37	80.24
+ δ	88.04	77.41
+ p + δ	88.21	80.84
+ f0/E-CNN	88.52	80.81
+ p + f0/E-CNN	88.45	**81.19**
+ δ + f0/E-CNN	88.44	80.09
+ p + δ + f0/E-CNN	**88.59**	80.84

Table 2: Parse and disfluency detection F1 scores on the dev set. Flat-F1 scores were consistently 0.1%-0.3% lower for our models, but 0.2% higher for the Berkeley parser (85.64).

We first show our results on the model using only text (i.e. $\boldsymbol{x}_i = \boldsymbol{e}_i$) to establish a strong baseline, on top of which we can add acoustic-prosodic features. We experiment with the *content*-only attention model used by Vinyals et al. (2015) and the *content+location* attention of Chorowski et al. (2015). For comparison with previous non-neural models, we use a high-quality latent-variable parser, the Berkeley parser (Petrov et al., 2006), retrained on our Switchboard data. Table 1 compares the three text-only models. In terms of F1, the *content+location* attention beats the Berkeley parser by about 2.5% and *content*-only attention by about 4.5%. Flat-F1 scores for both encoder-decoder models is lower than their corresponding F1 scores, suggesting that the encoder-decoder models do well on predicting the internal structure of EDIT nodes while the reverse is true for the Berkeley parser.

To explain the gains of *content+location* attention over *content*-only attention, we compare their scores on fluent (without EDIT nodes) and disfluent sentences, shown in Table 1. It is clear that most of the gains for *content+location* attention are from disfluent sentences. A possible explanation is the presence of duplicate words or phrases in disfluent sentences, which can be problematic for a *content*-only attention model. Since our best model is the *content+location* attention model, we will henceforth refer to it as the "CL-attn" text-only model. All models using acoustic-prosodic features are extensions of this model, which provides a strong text-only baseline.

Model	Parse	Disf
CL-attn	87.79 (0.11)	78.65 (0.46)
best model	88.15 (0.41)	80.48 (0.70)

Table 3: Parse and disfluency detection F1 scores on the dev set: mean (and standard deviation) over 10 runs for the baseline text-only model (CL-attn) and the best model with prosody.

Model	Parse	Disfl
Berkeley	85.87	63.44
CL-attn	87.99	76.69
best model	**88.50**	**77.47**

Table 4: Parse and disfluency detection F1 scores on the test set. The best model has *statistically significant* gains over the text-only baseline with p-value < 0.02.

4.2 Adding Acoustic-Prosodic Features

We extend our CL-attn model with the three kinds of acoustic-prosodic features: pause (p), word duration (δ), and CNN mappings of fundamental frequency (f0) and energy (E) features (f0/E-CNN).

The results of several model configurations on our dev set are presented in Table 2. First, we note that adding any combination of acoustic-prosodic features (individually or in sets) improves performance over the text-only baseline. However, certain combinations of acoustic-prosodic features are not always better than their subsets. The *text* + p + δ + *f0/E-CNN* model that uses all three types of features has the best performance with a gain of 0.7% over the already-strong text-only baseline. We will henceforth refer to the *text* + p + δ + *f0/E-CNN* model as our "best model".

As a robustness check, we report results of averaging 10 runs on the CL-attn text-only and the best model in Table 3. We performed a bootstrap test (Efron and Tibshirani, 1993) that simulates 10^5 random test draws on the models giving median performance on the dev set. These *median* models gave a statistically significant difference between the text-only and best model (p-value < 0.02). Additionally, a simple t-test over the two sets of 10 results also shows statistical significance p-value < 0.03.

Table 4 presents the results on the test set. Again, adding the acoustic-prosodic features improves over the text-only baseline. The gains are statistically significant for the best model with p-value < 0.02, again using a bootstrap test with simulated 10^5 random test draws on the two models.

Model	Parse	Disfl
Text Only		
Kahn et al. (2005)	86.4	78.2
Hale et al. (2006)	71.16	41.7
CL-attn (text only)	87.99	76.7
Text + Prosody		
Kahn et al. (2005)	86.6	78.2
Hale et al. (2006)	71.05	36.2
best model	88.50	77.5

Table 5: Parse and disfluency detection F1 scores on the test set comparing to other reported results.

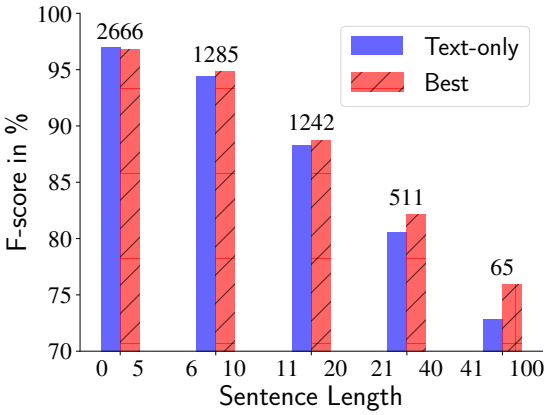

Figure 2: F1 scores of the text-only model and our best model as a function of sentence length.

Table 5 includes results from prior studies that compare systems using text alone with ones that incorporate prosody, given hand transcripts and sentence segmentation. It is difficult to compare systems directly, because of the many differences in the experimental set-up. For example, the original Charniak and Johnson (2001) result (reporting F=85.9 for parsing and F=78.2 for disfluencies) leverages punctuation in the text stream, which is not realistic for speech transcripts and not used in most other work. Our work benefits from more text training material than others, but others benefit from gold part-of-speech tags. Kahn et al. (2005) use a modified sentence segmentation. There are probably minor differences in handling of word fragments and scoring edit regions. Thus, this table primarily shows that our framework leads to more benefits from sentence-internal prosodic cues than others have obtained.

5 Analysis

Effect of sentence length. Figure 2 shows performance differences between our best model and the text-only model for varying sentence lengths.

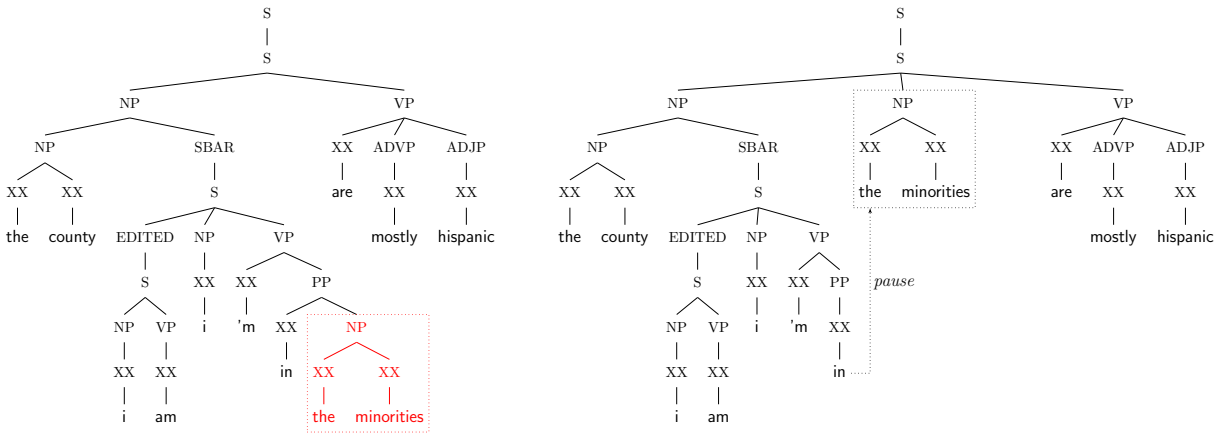

Figure 3: An example sentence from development data – *the county i am i 'm in [pause] the minorities are mostly hispanic*. The text-only parser (on the left) makes a PP Attachment error. The prosody-enhanced parser (on the right) uses the pause indicator to correctly predict a constituent change after the word *in*.

Model	fluent	disfluent
text-only	92.07	85.90
best model	92.03	**87.02**

Table 6: Dev set F1-score of text-only and best model on fluent (2029) vs. disfluent (3689) sentences.[10]

Both models do worse on longer sentences, as expected since the corresponding parse trees tend to be more complex. The performance difference between our best model and the text-only model increases with sentence length. This is likely because longer sentences more often have multiple prosodic phrases and disfluencies.

Effect of disfluencies. Table 6 presents parse scores on the subsets of fluent and disfluent sentences, showing that the performance gain is in the disfluent set (65% of the dev set sentences). Because sentence boundaries are given, and so many fluent sentences in spontaneous speech are short, there is less potential for benefit from prosody in the fluent set.

Types of errors. We use the Berkeley Parser Analyzer (Kummerfeld et al., 2012) to compare the types of errors made by the different parsers.[10] Table 7 presents the relative error reductions over the text-only baseline achieved by the text + p model and our best model for disfluent sentences. The two models differ in the types of error reductions they provide. Including pause information gives largest improvements on PP attachment and Modifier attachment errors. Adding the remaining acoustic-prosodic features helps to correct more types of attachment errors, especially VP and NP attachment. Figure 3 demonstrates one case where the pause feature helps in correcting a PP attachment error made by a text-only parser. Other interesting examples (see Appendix A.2) suggest that the learned f0/E features help reduce NP attachment errors where the audio reveals a prominent word at the constituent boundary, even though there is no pause at that word.

Error Type	Disfluent Sentences	
	text + p	best model
Clause Att.	5.7%	1.3%
Diff. Label	7.6%	4.2%
Modifier Att.	9.7%	19.1%
NP Att.	-2.7%	14.5%
NP Internal	7.8%	7.4%
PP Att.	10.1%	7.8%
1-Word Phrase	6.3%	6.8%
Unary	-1.1%	8.9%
VP Att.	0.0%	12.0%

Table 7: Relative error reduction over the text-only baseline in the disfluent subset (3689 sentences) of the development set. Shown here are the most frequent error types (with count $\geq$ 100 for the text-only model).

Effect of transcription errors. The results and analyses so far have assumed that we have reliable transcripts. In fact, the original transcripts contained errors, and the Treebank annotators used these without reference to audio files. Mississippi State University (MS-State) ran a clean-up project

[10] This analysis omits the 1% of the sentences that did not have timing information.

that produced more accurate word transcripts and time alignments (Deshmukh et al., 1998). The NXT corpus provides reconciliation between Treebank and MS-State transcripts in terms of annotating missed/extra/substituted words, but parses were not re-annotated. The transcript errors mean that the acoustic signal is inconsistent with the "gold" parse tree. Below are some examples of "fluent" sentences (according to the Treebank transcripts) with transcription errors, for which prosodic features "hurt" parsing. Words that transcribers missed are in brackets and those inserted are underlined.

S1: and because <uh> like if your spouse died <all of a sudden you be> all alone it 'd be nice to go someplace with people similar to you to have friends

S2: uh uh <i have had> my wife 's picked up a couple of things saying uh boy if we could refinish that 'd be a beautiful piece of furniture

Multi-syllable errors are especially problematic, leading to serious inconsistencies between the text and the acoustic signal. Further, the missed words lead to an incorrect attachment in the "gold" parse in S1 and a missing restart edit in S2. Indeed, for sentences with consecutive transcript errors, which we expect to impact the prosodic features, there is a statistically significant (p-value < 0.05) negative effect on parsing with prosody. Not included in this analysis are sentence boundary errors, which also change the "gold" parse. Thus, prosody may be more useful than results here indicate.

6 Related Work

Related work on parsing conversational speech has mainly addressed four problems: speech recognition errors, unknown sentence segmentation, disfluencies, and integrating prosodic cues. Our work addresses the last two problems, which involve studies based on hand-transcribed text and known sentence boundaries, as in much speech parsing work. The related studies are thus the focus of this discussion. We describe studies using the Switchboard corpus, since it has dominated work in this area, being the largest source of treebanked English spontaneous speech.

One major challenge of parsing conversational speech is the presence of disfluencies, which are grammatical and prosodic interruptions. Disfluencies include repetitions ('I am + I am'), repairs ('I am + we are'), and restarts ('What I + Today is the...'), where the '+' corresponds to an interruption point. Repairs often involve parallel grammatical constructions, but they can be more complex, involving hedging, clarifications, etc. Charniak and Johnson (Charniak and Johnson, 2001; Johnson and Charniak, 2004) demonstrated that disfluencies are different in character than other constituents and that parsing performance improves from combining a PCFG parser with a separate module for disfluency detection via parse rescoring. Our approach does not use a separate disfluency detection module; we hypothesized that the location-sensitive attention model helps handle these differences based on analysis of the text-only results (Table 1). However, more explicit modeling of disfluency pattern match characteristics in a dependency parser (Honnibal and Johnson, 2014) leads to better disfluency detection performance (F = 84.1 vs. 76.7 for our text only model). Pattern match features also benefit a neural model for disfluency detection alone (F = 87.0) (Zayats et al., 2016), and similar gains are observed by formulating disfluency detection in a transition-based framework (F = 87.5) (Wang et al., 2017). Experiments with oracle disfluencies as features improve the CL-attn text-only parsing performance from 87.85 to 89.38 on the test set, showing that more accurate disfluency modeling is a potential area of improvement.

It is well known that prosodic features play a role in human resolution of syntactic ambiguities, with more than two decades of studies seeking to incorporate prosodic features in parsing. A series of studies looked at constituent parsing informed by the presence (or likelihood) of prosodic breaks at word boundaries (Kahn et al., 2004, 2005; Hale et al., 2006; Dreyer and Shafran, 2007). Our approach improves over performance of these systems using raw acoustic features, without the need for hand-labeling prosodic breaks. The gain is in part due to the improved text-based parser, but the incremental benefit of prosody here is similar to that in these prior studies. (In prior work using acoustic feature directly (Gregory et al., 2004), prosody actually degraded performance.) Our analyses of the impact of prosody also extends prior work.

Prosody is also known to provide useful cues to sentence boundaries (Liu et al., 2006), and automatic sentence segmentation performance has been shown to have a significant impact on parsing performance (Kahn and Ostendorf, 2012). In our study, sentence boundaries are given so as to focus on the role of prosody in resolving sentence-internal parse ambiguity, for which prior work had

obtained smaller gains. Studies have also shown that parsing lattices or confusion networks can improve ASR performance (Kahn and Ostendorf, 2012; Yoshikawa et al., 2016). Our analysis of performance degradation for the system with prosody when the gold transcript and associated parse are in error suggests that prosody may have benefits for parsers operating on alternative ASR hypotheses.

The results we compare to in Section 4 are relatively old. More recent parsing results on spontaneous speech involve dependency parsers using only text (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Yoshikawa et al., 2016), with the exception of a study on unsupervised dependency parsing (Pate and Goldwater, 2013). With the recent success of transition-based neural approaches in dependency parsing, researchers have adapted transition-based ideas to constituent parsing (Zhu et al., 2013; Watanabe and Sumita, 2015; Dyer et al., 2016). These approaches have not yet been used with speech, to our knowledge, but we expect it to be straightforward to extend our prosody integration framework to these systems, both for dependency and constituency parsing.

7 Conclusion

We have presented a framework for directly integrating acoustic-prosodic features with text in a neural encoder-decoder parser that does not require hand-annotated prosodic structure. On conversational sentences, we obtained strong results when including word-level acoustic-prosodic features over using only transcriptions. The acoustic-prosodic features provide the largest gains when sentences are disfluent or long, and analysis of error types shows that these features are especially helpful in repairing attachment errors. In cases where prosodic features hurt performance, we observe a statistically significant negative effect caused by imperfect human transcriptions that make the "ground truth" parse tree and the acoustic signal inconsistent, which suggests that there is more to be gained from prosody than observed in prior studies. We thus plan to investigate aligning the Treebank and MS-State versions of Switchboard for future work.

Here, we assumed known sentence boundaries and hand transcripts, leaving open the question of whether increased benefits from prosody can be gained by incorporating sentence segmentation in parsing and/or in parsing ASR lattices. Most prior work using prosody in parsing has been on constituent parsing, since prosodic cues tend to align with constituent boundaries. However, it remains an open question as to whether dependency, constituency or other parsing frameworks are better suited to leveraging prosody. Our study builds on a parser that uses reverse order text processing, since it provides a stronger text-only baseline. However, the prosody modeling component relies only on a 1 second lookahead of the current word (for pause binning), so it could be easily incorporated in an incremental parser.

Acknowledgement

We thank the anonymous reviewers for their helpful feedback. We also thank Pranava Swaroop Madhyastha, Hao Tang, Jon Cai, Hao Cheng, and Navdeep Jaitly for their help with initial discussions and code setup. This research was partially funded by a Google Faculty Research Award to Mohit Bansal, Karen Livescu, and Kevin Gimpel; and NSF grant no. IIS-1617176. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, ..., and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

Sasha Calhoun, Jean Carletta, Jason M. Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The NXT-format Switchboard Corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation* 44(4).

Eugene Charniak and Mark Johnson. 2001. Edit Detection and Parsing for Transcribed Speech. In *Proc. NAACL*.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. 2015. Attention-Based Models for Speech Recognition. *CoRR* abs/1506.07503.

Neeraj Deshmukh, Andi Gleeson, Joseph Picone, Aravind Ganapathiraju, and Jonathan Hamaker. 1998. Resegmentation of SWITCHBOARD. In *Proc. ICSLP*.

Markus Dreyer and Izhak Shafran. 2007. Exploiting prosody for PCFGs with latent annotations. In *Proc. Interspeech*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. In *Proc. NAACL*.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Cécile Fourgeron and Patricia A. Keating. 1997. Articulatory strengthening at edges of prosodic domains. *Journal of the Acoustical Society of America* 101(6).

John J. Godfrey and Edward Holliman. 1993. *Switchboard-1 Release 2*. Linguistic Data Consortium.

Michelle L Gregory, Mark Johnson, and Eugene Charniak. 2004. Sentence-Internal Prosody Does not Help Parsing the Way Punctuation Does. In *Proc. NAACL*.

Franćois Grosjean, Lysiane Grosjean, and Harlan Lane. 1979. The patterns of silence: Performance structures in sentence production. *Cognitive Psychology* .

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Mathew Snover, and Robin Stewart. 2006. PCFGs with Syntactic and Prosodic Indicators of Speech Repairs. In *Proc. COLING-ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8).

Matthew Honnibal and Mark Johnson. 2014. Joint Incremental Disfluency Detection and Dependency Parsing. *TACL* .

Mark Johnson and Eugene Charniak. 2004. A TAG-based Noisy Channel Model of Speech Repairs. In *Proc. ACL*.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective Use of Prosody in Parsing Conversational Speech. In *Proc. HLT/EMNLP*.

Jeremy G. Kahn and Mari Ostendorf. 2012. Joint reranking of parsing and word recognition with automatic segmentation. *Computer Speech & Language* .

Jeremy G. Kahn, Mari Ostendorf, and Ciprian Chelba. 2004. Parsing Conversational Speech Using Enhanced Segmentation. In *Proc. NAACL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser Showdown at the Wall Street Corral: An Empirical Investigation of Error Types in Parser Output. In *Proc. EMNLP*.

Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper. 2006. Enriching Speech Recognition with Automatic Detection of Sentence Boundaries and Disfluencies. *IEEE TASLP* 14.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *Proc. ICLR*.

John Pate and Sharon Goldwater. 2013. Unsupervised Dependency Parsing with Acoustic Cues. *TACL* 1.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proc. COLING-ACL*.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves Recurrent Neural Networks for Handwriting Recognition. In *Proc. ICFHR*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *Proc. ASRU*.

Patti Price, Mari Ostendorf, Stefanie Shattuck-Hufnagel, and Cynthia Fong. 1991. The Use of Prosody in Syntactic Disambiguation. In *Proc. Workshop on Speech and Natural Language*.

Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint Parsing and Disfluency Detection in Linear Time. In *Proc. EMNLP*.

Elizabeth Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Department of Psychology, University of California, Berkeley, CA.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Proc. NIPS*.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based Disfluency Detection using LSTMs. In *Proc. EMNLP*.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based Neural Constituent Parsing. In *Proc. ACL*.

Colin W. Wightman, Stefanie Shattuck-Hufnagel, Mari Ostendorf, and Patti J. Price. 1992. Segmental durations in the vicinity of prosodic phrase boundaries. *Journal of the Acoustical Society of America* 91(3).

Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts. In *Proc. EMNLP*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *CoRR* abs/1409.2329.

Victoria Zayats, Hannaneh Hajishirzi, and Mari Ostendorf. 2016. Disfluency Detection using a Bidirectional LSTM. In *Proc. Interspeech*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and Accurate Shift-Reduce Constituent Parsing. In *Proc. ACL*.

A Appendix

A.1 Miscellany

Our main model code is available at https://github.com/shtoshni92/speech_parsing. Most of the data preprocessing code is available at https://github.com/trangham283/seq2seq_parser/tree/master/src/data_preps. Part of our data preprocessing pipeline also uses https://github.com/syllog1sm/swbd_tools.

Table 8 shows statistics of our Switchboard dataset. As defined, for example, in (Charniak and Johnson, 2001; Honnibal and Johnson, 2014), the splits are: conversations sw2000 to sw3000 for training, sw4500 to sw4936 for validation (dev), and sw4000 to sw4153 for evaluation (test). In addition, previous work has reserved sw4154 to sw4500 for "future use" (dev2), but we added this set to our training set. That is, all of our models are trained on Switchboard conversations sw2000 to sw3000 as well as sw4154 to sw4500.

Section	# sentences	# words
Train	97,113	729,252
Dev	5,769	50,445
Test	5,901	48,625

Table 8: Data statistics.

Figure 4 illustrates the data preprocessing step. On the decoder end, we also use a post-processing step that merges the original sentence with the decoder output to obtain the standard constituent tree representation. During inference, in rare cases (and virtually none as our models converge), the decoder does not generate a valid parse sequence, due to the mismatch in brackets and/or the mismatch in the number of pre-terminals and terminals, i.e., num(XX) $\neq$ num(tokens). In such cases, we simply add/remove brackets from either end of the parse, or add/remove pre-terminal symbols XX in the middle of the parse to match the number of input tokens.

Figure 5 shows the distribution of pause durations in our training data. Our pause buckets of

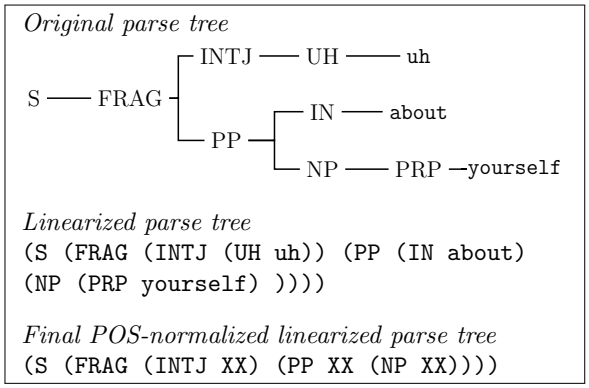

Figure 4: Data preprocessing. Trees are linearized; POS tags (pre-terminals) are normalized as "XX". Also note the annotation standard used for Switchboard data: The root node of the tree is an "S" node although it is not a complete sentence.

$0 < p \leq 0.05$ s, 0.05 s $< p \leq 0.2$ s, $0.2 < p \leq 1$ s, and $p > 1$ s described in the main paper were based on this distribution of pause lengths.

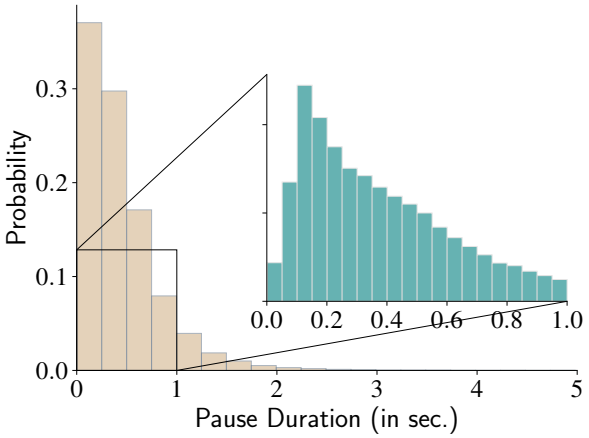

Figure 5: Histogram of inter-word pause durations in our training set. As expected, most of the pauses are less than 1 second. Further binning of pause durations ≤ 1 second reveals that the plot peaks around 0.2 seconds and continuously decays from there on. In some very rare cases, pauses of 5+ seconds occur within a sentence.

Table 9 shows the comprehensive error counts in all error categories defined in the Berkeley Parse Analyzer (Kummerfeld et al., 2012) in both the fluent and disfluent subsets.

A.2 Tree Examples

In figures 6, 7, and 8, we follow node correction notations as in (Kummerfeld et al., 2012). In particular, missing nodes are marked in blue on the gold tree, extra nodes are marked red in the predicted tree, and yellow nodes denote crossing.

Error Type	Fluent Subset			Disfluent Subset		
	text-only	text + p	best model	text-only	text + p	best model
Clause Attach.	126	132	123	631	595	600
Co-ordination	1	2	1	10	10	5
Different label	112	116	124	288	266	300
Modifier Attach.	119	127	112	320	289	325
NP Attach.	92	89	94	332	341	283
NP Internal	71	61	65	231	213	232
PP Attach.	171	152	149	524	471	470
1-Word Phrase	334	342	328	1054	988	1030
UNSET add	86	81	64	353	352	356
UNSET move	85	93	95	466	447	439
UNSET remove	73	70	56	334	324	318
Unary	246	239	236	1088	1100	1074
VP Attach.	36	41	25	167	167	172
XoverX Unary	36	35	34	54	57	54

Table 9: Parse error counts comparison on the fluent (2029 sentences) and disfluent (3689 sentences) subsets of the development set across three parsers.

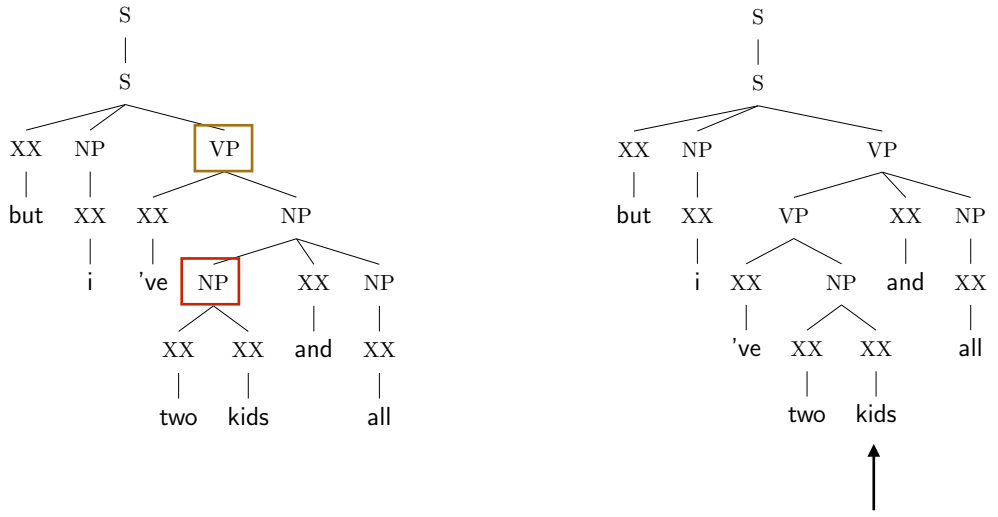

Figure 6: An example sentence from development data – *but i 've two kids and all*. Even though there are no pauses between all words, the word *kids* is lengthened in the audio sample, helping the prosody-enhanced parser (right) to recognize a major syntactic boundary, avoiding the NP Attachment error made by the text-only parser (left).

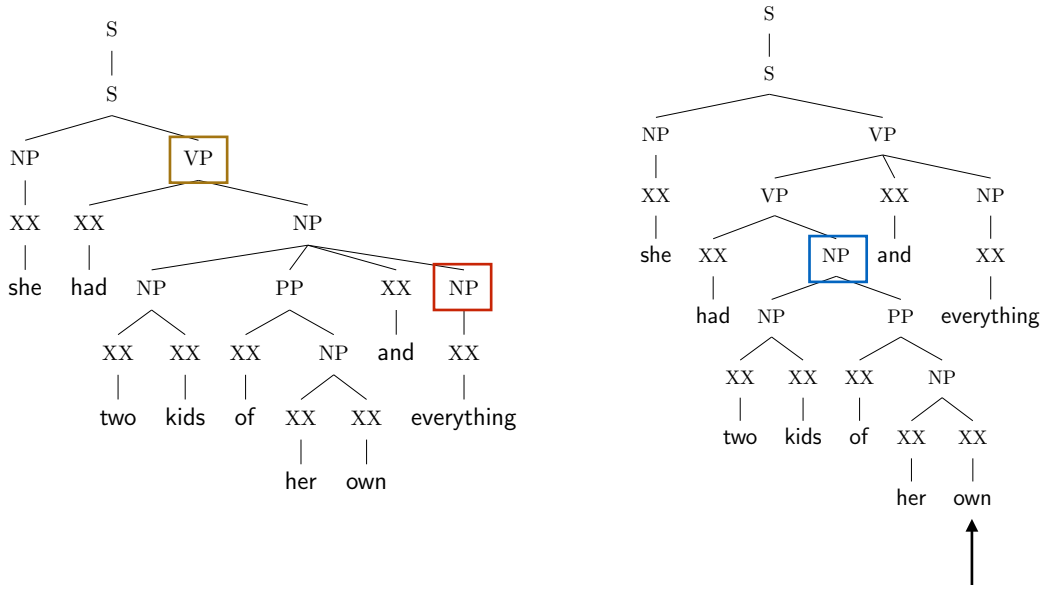

Figure 7: An example sentence from development data – *she had two kids of her own and everything*. There were no pauses between all words in this sentence, the audio sample showed that the word *own* was both lengthened and raised in intonation, giving the prosody-enhanced parser (right) a signal that *own* is on a syntactic boundary. On the other hand, the text-only parser (left) had no such information and made an NP-attachment error. This sentence also illustrates an interesting case where, in isolation, the text-only parse makes sense (i.e. *everything* being an object of *had*). However, in the context of this conversation (the speaker was talking about another person in an informal manner), *and everything* acts more like filler - e.g. "i play the violin *and stuff*"

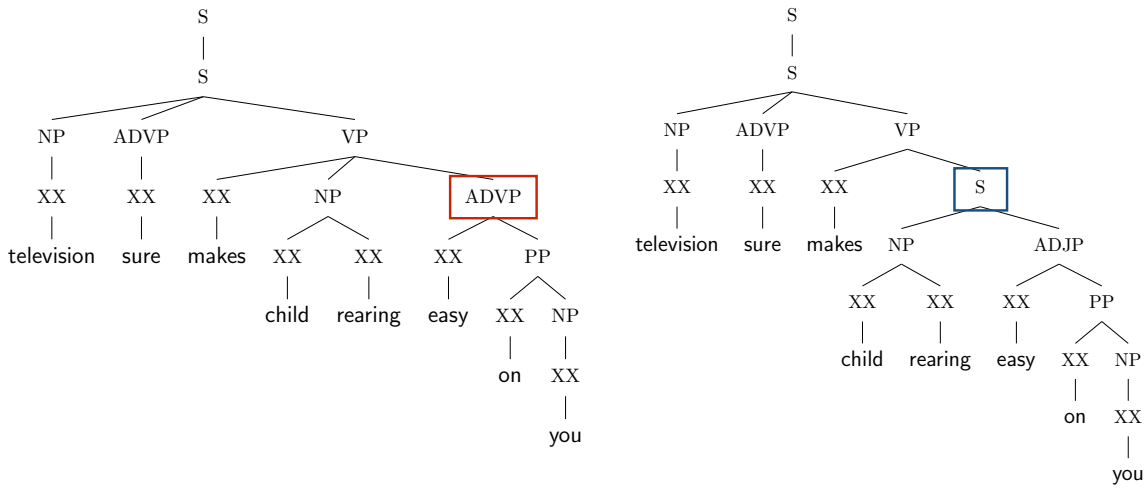

Figure 8: An example sentence from development data – *television sure makes child rearing easy on you*. This is an example where our prosody-enhanced parser (left) did worse than the text-only parser (right), which made no errors. The error type illustrated here is Different Label and Modifier Attachment. In the first iteration, the analyzer identifies a Different Label error (ADVP node), and in the second pass identifies the Modifier Attachment error.