Joint Modeling of Text and Acoustic-Prosodic Cues for Neural Parsing

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 a spoken utterance, 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 word-based prosodic features. We find that different types of acoustic-prosodic features are individually helpful, and together improve parse F1 scores significantly over a strong text-only baseline. For this study with known sentence boundaries, error analysis shows that the main benefit of acoustic-prosodic features is in sentences with disfluencies and that attachment errors are most improved.

1 Introduction

While constituent parsing has become a relatively mature technology for written text, parser performance on conversational speech lags behind results on text. Speech poses challenges for parsing. First, transcripts may contain errors and lack punctuation. Joint speech recognition and parsing is useful for dealing with word errors and sentence segmentation (Kahn and Ostendorf, 2012). Second, 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 that are common in spontaneous speech. Some of these phenomena can be handled in standard grammars. Disfluencies typically require extensions of the model, and different approaches have been explored in both constituent parsing (Charniak and Johnson, 2001; Johnson and Charniak, 2004) and dependency parsing (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014).

In addition to these challenges, speech also carries helpful extra information – beyond the words – associated with the prosodic structure of an utterance and encoded via variation in timing and intonation. Studies show that speakers pause in locations that are correlated with syntactic structure (Grosjean et al., 1979), and listeners are able to 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. This study focuses on this last challenge, aiming to incorporate prosodic cues in a state-of-the-art neural parser.

Researchers have explored a number of approaches for incorporating prosody in parsing. A challenge of using prosodic features is that multiple acoustic cues interact to signal prosodic structure, including pauses, duration lengthening, fundamental frequency modulation, and even spectral shape. These cues also vary with the phonetic segment, emphasis, emotion and speaker, so feature extraction has typically involved experimenting with multiple different time windows and normalization techniques. Researchers have dealt with this by explicitly predicting prosodic structure. The work proposed here takes advantage of advances in neural networks to automatically learn a good feature representation for parsing without the need for

[*]Equal Contribution.

explicitly representing prosodic constituents. To narrow the scope of this work and facilitate error analysis, our experiments use hand transcripts and known sentence segmentation.

Our work offers the following contributions: We present a neural encoder-decoder model for parsing conversational speech that provides a framework for directly integrating acoustic-prosodic features with text; we demonstrate that significant improvements in parsing performance are obtained without requiring hand-annotated prosodic structure; and we provide analyses that show that combining multiple types of prosodic cues lead to gains for specific types of ambiguities.

2 Related Work

Related work on parsing conversational speech has mainly addressed two problems: i) handling disfluencies and ii) integrating prosodic cues.

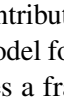
One major challenge of parsing conversational speech is the presence of disfluencies, which are grammatical (and prosodic) interruptions. Disfluencies include repetitions ('I am + I am'), repairs ('I am + we are'), and restarts ('What I + Today is the...'), where the '+' corresponds to an interruption point. Repairs often involve parallel grammatical constructions, but they can be more complex, involving hedging, clarifications, etc. One solution for handling dislfuencies is to first automatically detect them, remove the edited words (reparandum), and then parse the cleaned-up text. Disfluency detection is an active area of research (Georgila, 2009; Qian and Liu, 2013; Ferguson et al., 2015; Zayats et al., 2015, 2016). While disfluency detection has improved greatly, it still does not achieve high accuracy for a broad range of tasks, so researchers have emphasized joint parsing and disfluency detection. Such models have been explored for syntactic constituents (Charniak and Johnson, 2001; Kahn et al., 2005), or dependencies (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014).

Researchers have been exploring different approaches to incorporating prosody in parsing since the 1990s. One study used quantized acoustic-prosodic cues as tokens in parsing, similar to punctuation, and observed a degradation in performance (Gregory et al., 2004). They hypothesize that these tokens are problematic because they break n-gram dependencies. Because multiple prosodic cues interact to signal prosodic structure, most studies have leveraged symbolic prosodic boundary markers (prosodic breaks) based on the ToBI prosodic annotation system (Silverman et al., 1992). In some studies, automatically predicted prosodic breaks are incorporated directly (Hale et al., 2006; Dreyer and Shafran, 2007; Huang and Harper, 2010). Another study uses prosodic break posteriors in parsing (Kahn et al., 2005). In either case, the use of symbolic prosodic breaks requires availability of a hand-annotated corpus for training break detection models. In addition, the parsing approaches that are most successful in using prosody leverage N-best hypothesis rescoring, limiting the impact of prosodic features to a small number of hypotheses.

Recently, attention-enabled encoder-decoder models (Bahdanau et al., 2015) have gained traction for constituency parsing, with Vinyals et al. (2015) achieving state-of-the-art results for the Wall Street Journal (WSJ) corpus using an ensemble. Performance has since been improved by another ensemble of encoder-decoder models trained in a multi-task setting (Luong et al., 2016). Our models are based on this attention-based approach.

Original parse tree

```
(S (FRAG (INTJ (UH uh))
   (PP (IN about)
       (NP (PRP yourself) ))))
```



Encoder input:
uh about yourself

Decoder gold standard parse:

```
(S (FRAG (INTJ XX) (PP XX (NP XX))))
```

Figure 1: 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.

3 Task and Model Description

We next describe our encoder-attention-decoder model that 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 3.3.

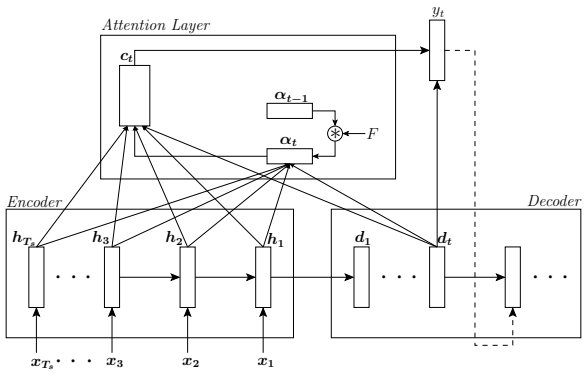


Figure 2: An attention-enabled encoder-decoder model reading the input features $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{T_s}$, where $\boldsymbol{x}_i = [\boldsymbol{e}_i\ \boldsymbol{\phi}_i\ \boldsymbol{s}_i]$ is composed of word embeddings $\boldsymbol{e}_i$, manually defined prosodic features $\boldsymbol{\phi}_i$, and learned (CNN-based) features $\boldsymbol{s}_i$. The encoder reads the input in *reverse* order and the decoder outputs the linearized parse sequence $y_1, \cdots, y_t, \cdots$.

3.1 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, with pre-terminals also normalized as "XX". Figure 1 illustrates how we convert standard treebank parses into encoder inputs and gold standard linearized parses.

3.2 Encoder-Attention-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 input feature sequence[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 $\boldsymbol{y_{<t}}$.

The attention mechanism used by Vinyals et al. (2015) computes the context vector $\boldsymbol{c}_t$ as follows:

$$\begin{aligned} u_{it} &= \boldsymbol{v}^\top \tanh(\boldsymbol{W}_1\boldsymbol{h}_i + \boldsymbol{W}_2\boldsymbol{d}_t + \boldsymbol{b}_a) \\ \boldsymbol{\alpha}_t &= \text{softmax}(\boldsymbol{u}_t) \\ \boldsymbol{c}_t &= \sum_{i=1}^{T_s} \alpha_{ti}\boldsymbol{h}_i \end{aligned}$$

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 can be crucial since disfluent structures can have duplicate words/phrases that may "confuse" the attention mechanism.[3]

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:

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

[1] As in Vinyals et al. (2015), the input sequence is processed in reverse order, as shown in Figure 2.

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

[3] This phenomenon has been observed in encoder-decoder models for speech recognition (Chorowski et al., 2014).

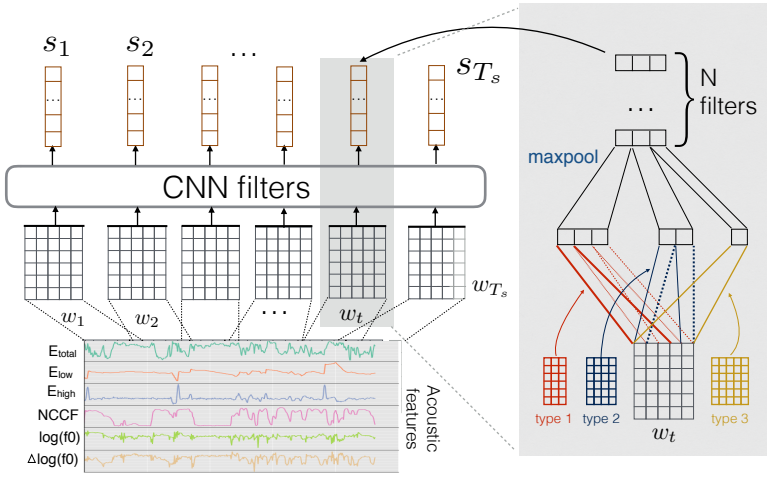


Figure 3: Detailed illustration of acoustic-prosodic feature learning module. CNN features are computed from input energy and pitch features; here the CNN filter parameters are $m = 3$ and $w = [3, 4, 5]$.

where $\boldsymbol{W}_f$ is another learnable parameter matrix.

Finally, the decoder hidden 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}.

3.3 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 also use 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 the features are different in nature (word-level measurements vs. time series), they are integrated with the encoder-decoder using different mechanisms.

All features are extracted from transcriptions that are time-aligned at the word level. In a small number of cases, the time alignment for a particular word boundary is missing, mainly due to tokenization. For example, contractions, such as *don't* in the original transcript, are labeled as separated words, *do* and *n't*, for the parser and the internal word boundary time is missing. In many cases, these internal times are estimated, but in a few cases estimation was difficult. For the roughly 1% of sentences where time alignments are missing, we simply backoff to the text-based parser.

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: off (no pause), not-available, $0 < p \leq 0.05$ s, 0.05 s $< p \leq 0.2$ s, $0.2 < p \leq 1$ s, and $p > 1$ s. This bucketing scheme for pause duration is motivated by the distribution of pause durations in our data and the assumption that longer pause length differences are not useful.

Word-final duration lengthening is a strong cue to prosodic phrase boundaries (Wightman et al., 1992). The **word duration** feature δ_i is a real-valued scalar, equal to 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.

For **fundamental frequency (f0) and energy (E) contours,** we use a CNN to automatically learn a mapping from the time series to a word-level vector. The contour features are extracted with 25-ms frames with 10-ms hops using Kaldi (Povey et al., 2011), motivated by the success of the associated f0 features in speech recognition (Ghahremani et al., 2014). 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 simple mechanism to capture articulatory strengthening at prosodic constituent onsets (Fourgeron and Keating, 1997).

Figure 3 summarizes the feature learning approach. The f0 and E features are processed at the word level: each sequence of frames corresponding to a 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.

4 Experiments

4.1 Dataset

Our core corpus is Switchboard-NXT (Calhoun et al., 2010), a subset of the Switchboard corpus (Godfrey and Holliman, 1993). Switchboard I – Release 2 (Godfrey and Holliman, 1993) is a collection of about 2,400 telephone conversations between strangers; 650 such conversations were later hand-annotated with syntactic parses as part of the Penn Treebank Release 3 dataset (Marcus et al., 1999), and 642 of these were 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 3 set, with a few manual corrections from the NXT release. This core dataset consists of 100K sentences, totaling 1M tokens forming a vocabulary of 13.5K words. We follow the sentence boundaries defined by the parsed data available [4].

We follow the data split defined by Charniak and Johnson (2001), as well as related work done on Switchboard (Johnson and Charniak, 2004; Kahn et al., 2005; Gregory et al., 2004; Honnibal and Johnson, 2014)[5]: 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. The overall data statistics are shown in Table 1.

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

Table 1: Data statistics.

4.2 Evaluation Metric

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.[6] Transcribed speech, however, includes disfluencies, with speech repairs (labeled under "EDITED" nodes in Switchboard parse trees) being particularly problematic for statistical parsers, as explained by Charniak and Johnson (2001). For this reason, previous work has often explicitly detected EDIT regions in conjunction with parsing or as a preprocessing step.

In our context, we are not addressing the disfluency detection problem. Therefore, besides the fact that we are using a larger training data set, our results are not comparable with those of previous work on Switchboard constituent parsing, which involved removing and re-inserting EDIT nodes during evaluation according to a set of rules (Charniak and Johnson, 2001; Kahn et al., 2005). Nevertheless, acknowledging the fact that we are evaluating parsing of transcribed speech, we also report *flattened-edit parseval* F1 scores ("flat-F1"), which is *parseval* computed on trees with "flattened" edit nodes, where the structure under edit nodes has been eliminated so that all leaves are immediate children.

4.3 Model Parameters and Training Details

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 convolution filters of width 40

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

[5]Part of our data preprocessing pipeline uses `https://github.com/syllog1sm/swbd_tools`

[6]`http://nlp.cs.nyu.edu/evalb/`

Model	F1	flat-F1
Berkeley	85.41	85.91
content-only attention	83.33	83.20
***content+location* attention**	**87.85**	**87.68**

Table 2: Scores of text-only models on the dev set.

Model	fluent	disfluent
content-only attention	90.86	79.94
***content+location* attention**	**92.07**	**85.95**

Table 3: F1-score of different attention models on fluent (2044) vs. disfluent (3725) sentences.

each. We use 512-dimensional embedding vectors to represent words and linearized parsing symbols, such as "(S".

A relatively small 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 size combinations $\in \{[10, 25, 50], [5, 10, 25, 50]\}$, and number of filters per filter size $N \in \{16, 32, 64, 128\}$[7]. These filter size combinations are motivated by the fact that the average word length in our dataset is 25 frames, so intuitively the different filter sizes are capturing f0 and energy phenomena on various levels: $w = 5, 10$ for sub-word, $w = 25$ for word, and $w = 50$ for word and context.

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 w.r.t. 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). We use TensorFlow (Abadi et al., 2015) to implement all models.

4.4 Inference

For inference, we use a greedy decoder to generate the linearized parse sequence. 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. As shown in Figure 1, we also use a post-processing step that merges the original sentence tokens with the decoder output to obtain the standard constituent tree representation.[8]

5 Results

5.1 Text-only Results

We first show our text-only model results (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 model proposed by 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 2 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%. Interestingly, flat-F1 scores for both encoder-decoder models is lower than their corresponding F1 scores. This suggests that the encoder-decoder models do well on predicting the internal structure of EDIT nodes. The reverse is true for the Berkeley parser: Flattening the EDIT nodes results in a better score.

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 3. From the table, it is clear that most of the gains for *content+location* attention are from disfluent sentences. The reason may be that disfluent sentences can have duplicate words or phrases, 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 our "text-only" model.

[7] So when we use filter sizes [10, 25, 50] with 16 of each type, we have a total of 48 filters. Also, note that the filter sizes are actually 6×10, 6×25, etc., but since the feature dimension is fixed (6 in our case), we specify only the filter width.

[8] 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.

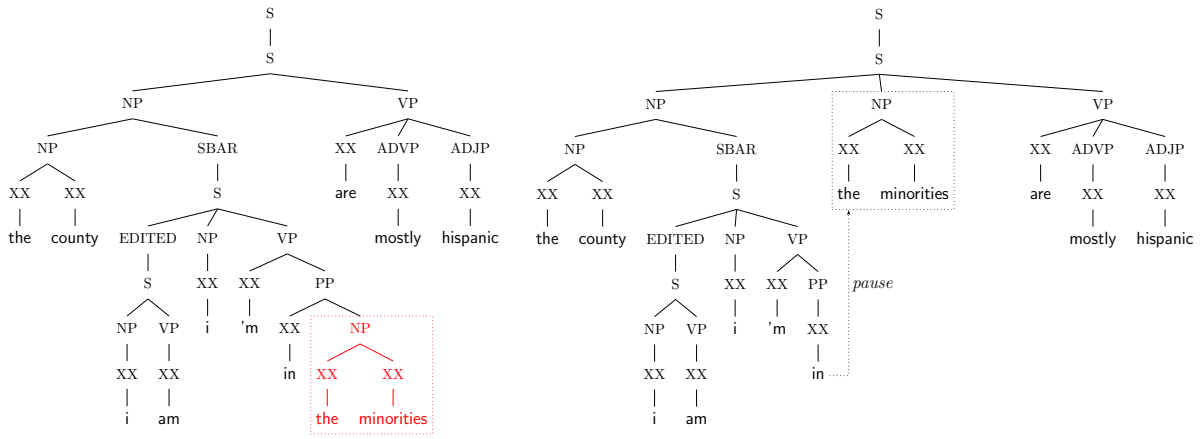


Figure 4: An example sentence from development data – *the county i am i 'm in [pause] the minorities are mostly hispanic*. The text-only parser (on the left), which is oblivious to prosodic cues, makes a PP Attachment error. The prosody-enhanced parser (on the right) uses the pause indicator to correctly predict a constituent change after the word *in*.

Model	F1	flat-F1
Berkeley	85.41	85.64
text-only	87.85	87.53
text + p	88.37	88.07
text + δ	88.04	87.69
text + p + δ	88.21	87.90
text + f0/E-CNN	88.52	88.21
text + p + f0/E-CNN	88.45	88.24
text + δ + f0/E-CNN	88.44	88.14
text + p + δ + f0/E-CNN	**88.59**	**88.27**

Table 4: F1 scores on the dev set.

Model	F1	flat-F1
Berkeley	85.87	85.91
text-only	87.99	87.68
text + p + δ	88.06	87.70
text + f0/E-CNN*	88.44	88.17
text + p + δ + f0/E-CNN*	**88.50**	**88.20**

Table 5: F1 scores on the test set. Models marked with * have *statistically significant* gains over the text-only baseline with p-value < 0.02.

5.2 Adding Acoustic-Prosodic Features

We extend our text-only 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).

There are $2^3 - 1 = 7$ model configurations corresponding to the possible combinations of the acoustic-prosodic features. The results of the 7 model configurations and the text-only models on our dev set are presented in Table 4. [9] 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 guaranteed to be better than their subsets, suggesting negative interaction among features (e.g. *text+p+wd*). The best model, *text* + p + δ + *f0/E-CNN*, uses all three types of features and has a gain of 0.7%, over the already-strong text-only baseline.

Table 5 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 *text* + *f0/E-CNN*, and *text* + p + δ + *f0/E-CNN* models at p-value < 0.02. The p-values are estimated using a bootstrap test (Efron and Tibshirani, 1993) that simulates 10^5 random test draws. Our best-performing model, *text* + p + δ + *f0/E-CNN*, will henceforth be referred to as "best model."

6 Analysis

We first study performance differences between our best model and the text-only model for varying sentence lengths, shown in Figure 5. Both models do worse on longer sentences, which is not surprising since the corresponding parse trees tend to be more

[9] For about 1% of the sentences in both dev and test, no time-alignments were available. Thus, we don't have acoustic-prosodic features in such cases, and we have to back-off to the text-only model during evaluation.

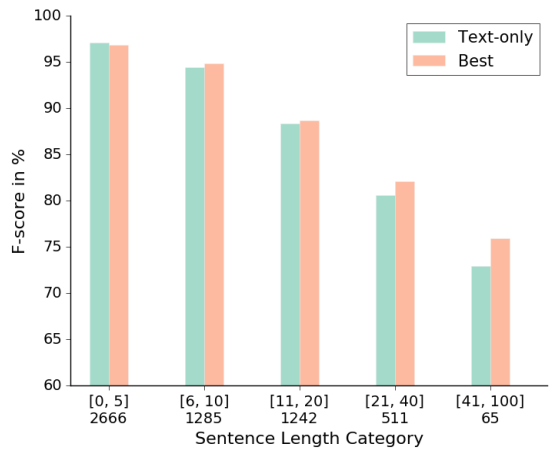


Figure 5: F1 scores of the text-only model and our best model as a function of sentence length. Acoustic-prosodic information helps more as sentence length increases.

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]

complex. The performance difference between our best model and the text-only model increases with sentence length. This may also be expected, since longer sentences are more likely to have multiple prosodic phrases and disfluencies.

Because sentence boundaries are given, and so many sentences in spontaneous speech are short, there is a possibility that the benefit from prosody is mainly related to disfluencies. Table 6 presents parse scores on the subset of fluent and disfluent sentences, suggesting that this may be the case.

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.

This analysis shows that, by including only pause information, we see the 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. The text

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

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 ≥ 100 for the text-only model).

+ p model and the best model differ quite a bit in the types of error reductions they provide. A more detailed analysis of what information each type of acoustic-prosodic feature is capturing would be an interesting topic for future work.

Figure 4 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) are those where the learned f0/E features are valuable in avoiding NP attachment errors in cases where the audio reveals a prominent word at the constituent boundary, even though there is no pause at that word.

7 Conclusion

We have presented an attention-based encoder-decoder model for parsing conversational sentences, obtaining strong results when parsing the text transcriptions and further improved results when including word-level acoustic-prosodic features. Unlike recent prior work, we do not use an explicit disfluency detection step, and we automatically learn mappings of f0 and energy contours using a CNN that is jointly trained with the attention model. 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. Further analysis may be helpful for revealing more specific contrasts, understanding how fine-grained duration would impact the findings, and discovering which aspects of prosody are learned by the CNN. The significant improvement

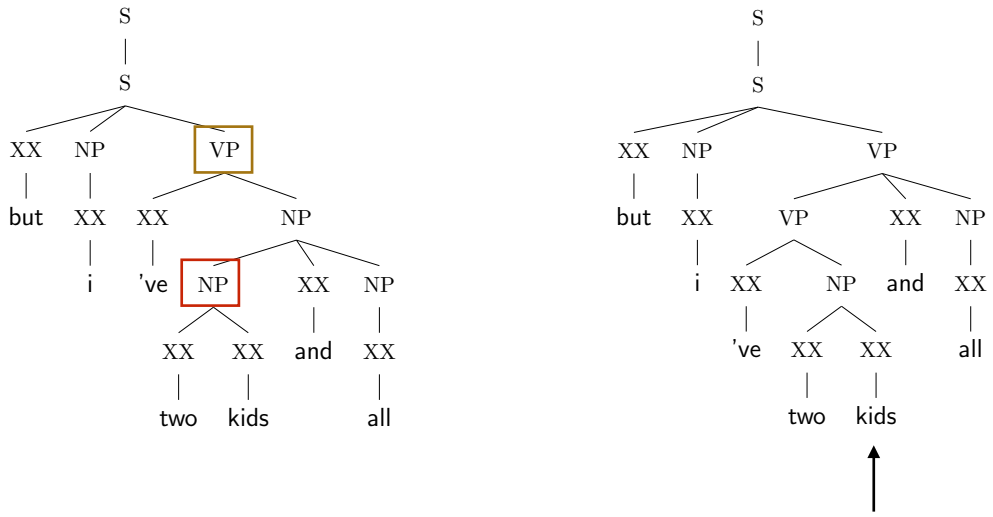


Figure 6: An example sentence from development data – *but i 've two kids and all*. Even though there are no pauses between all words, the word *kids* is lengthened in the audio sample, helping the prosody-enhanced parser (right) to recognize a major syntactic boundary, avoiding the NP Attachment error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

obtained when using only automatically learned features suggests that they may also be useful in other related problems, such as dialog act recognition. In future work we would like to explore direct speech parsing jointly with speech recognition.

Acknowledgement

We 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.

8 Appendix

8.1 Tree Examples

In figures 6, 8, 9, and 10, 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.

8.2 Miscellany

Figure 7 shows the distribution of pause duration in our training data. Our pause buckets of $0 < p \leq 0.05$ 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 figure.

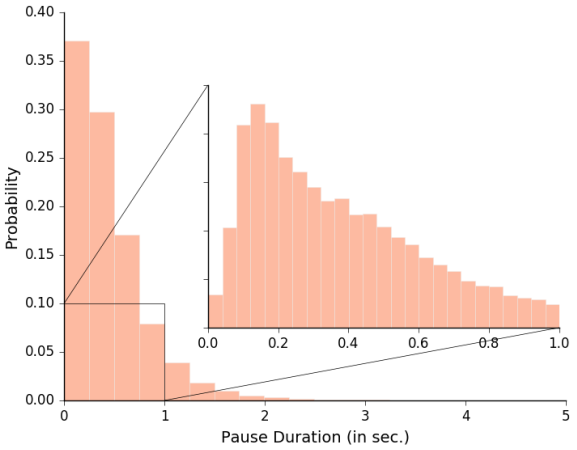


Figure 7: Histogram of inter-word pause durations in our training set. As expected, most of the pauses are less than 1 second. Further binning of pause durations ≤ 1 second reveals that the plot peaks around 0.2 seconds and continuously decays from there on. In some very rare cases, pauses of 5+ seconds occur within a sentence.

Table 8 shows the comprehensive error counts in all error categories in both the fluent and disfluent subsets.

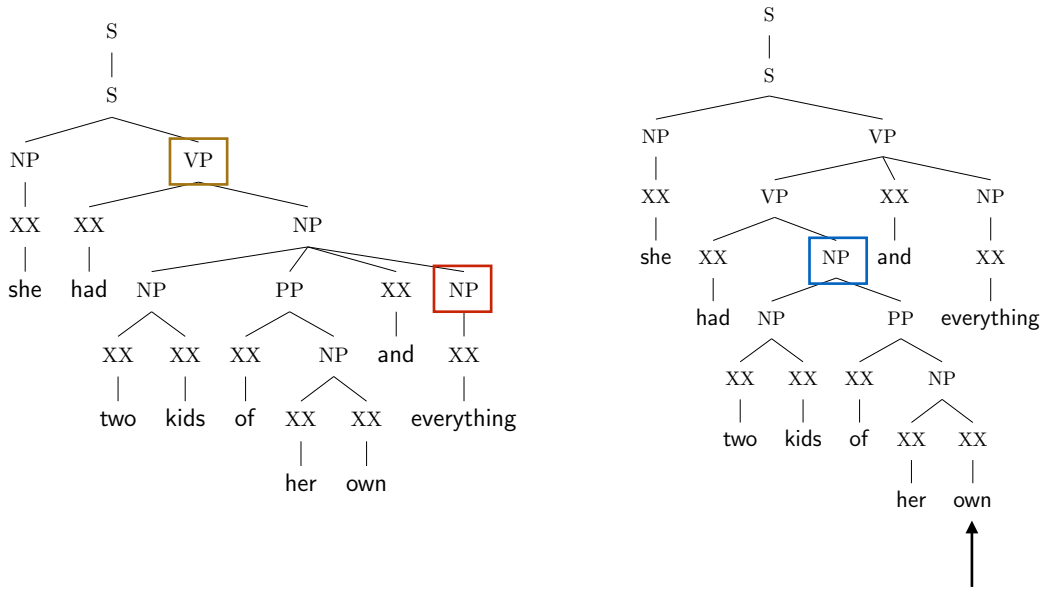


Figure 8: An example sentence from development data – *she had two kids of her own and everything*. There were no pauses between all words in this sentence, the audio sample showed that the word *own* was both lengthened and raised in intonation, giving the prosody-enhanced parser (right) a signal that *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*"

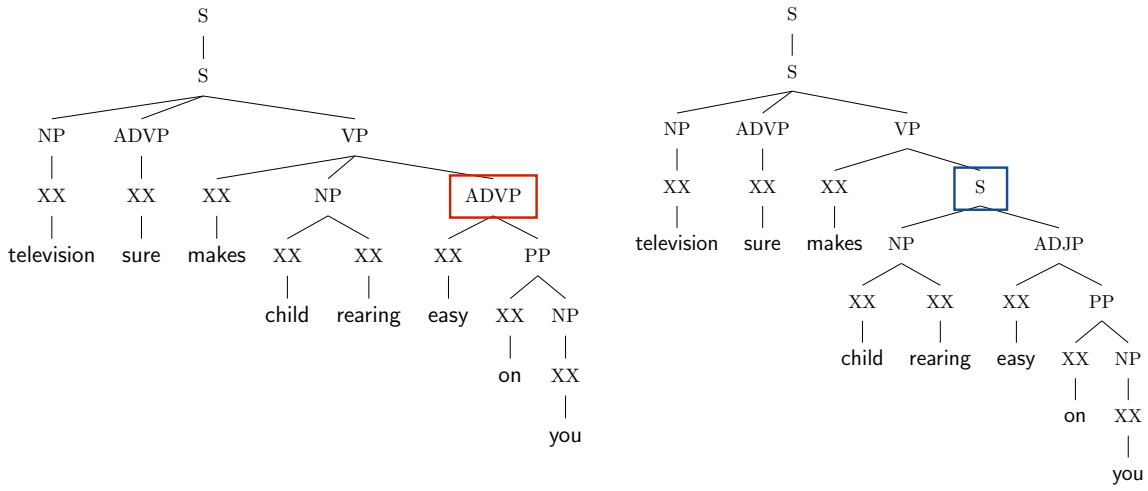


Figure 9: An example sentence from development data – *television sure makes child rearing easy on you*. This is an example where our prosody-enhanced parser (left) did worse than the text-only parser (right), which made no errors. The error type illustrated here is Different Label and Modifier Attachment. In the first iteration, the analyzer identifies a Different Label error (ADVP node), and in the second pass identifies the Modifier Attachment error.

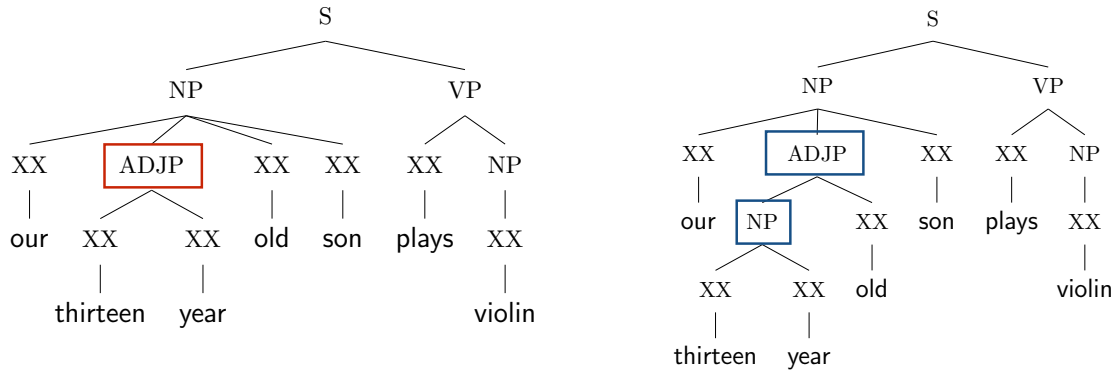


Figure 10: An example sentence from development data – *our thirteen year old son plays violin*. This is an example where our prosody-enhanced parser (left) did worse than the text-only parser (right), which made no errors. The error type illustrated here is Different Label and Modifier Attachment. The analyzer algorithm identifies node ADJP as Different Label in the first iteration, and in the second iteration identifies a Modifier Attachment error with the whole *thirteen year* constituent.

	Fluent Subset			Disfluent Subset		
Error Type	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 8: Parse error counts comparison on the fluent (2029 sentences) and disfluent (3689 sentences) subsets of the development set across three parsers. Presented here is a subset of the most frequent error types.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

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, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First results. *CoRR* abs/1412.1602.

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

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

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency Detection with a Semi-Markov Model and Prosodic Features. In *Proc. NAACL*.

Cécile Fourgeron and Patricia A. Keating. 1997. Articulatory strengthening at edges of prosodic domains. *Journal of the Acoustical Society of America* 101(6):3728–3740.

Kallirroi Georgila. 2009. Using Integer Linear Programming for Detecting Speech Disfluencies. In *Proc. NAACL*.

Pegah Ghahremani, Bagher BabaAli, Daniel Povey, Korbinian Riedhammer, Jan Trmal, and Sanjeev Khudanpur. 2014. A pitch extraction algorithm tuned for automatic speech recognition. In *Proc. ICASSP*.

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* .

Zhongqiang Huang and Mary Harper. 2010. Appropriately Handled Prosodic Breaks Help PCFG Parsing. In *Proc. NAACL*.

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* .

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*.

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

Mitchell P. Marcus, Beatrice Santorini, Mary A. Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. Technical report, Linguistic Data Consortium.

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. International Conference on Frontiers in Handwriting Recognition (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*.

Xian Qian and Yang Liu. 2013. Disfluency Detection Using Multi-step Stacked Learning. In *Proc. NAACL*.

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.

Kim Silverman, Mary Beckman, John Pitrelli, Mari Ostendorf, Colin Wightman, Patti Price, Janet Pierrehumbert, and Julia Hirschberg. 1992. Tobi: A Standard for Labeling English Prosody. In *Proc. ICSLP*.

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

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).

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*.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2015. Unediting: Detecting Disfluencies Without Careful Transcripts. In *Proc. NAACL*.