Prosodic segmentation for parsing spoken dialogue

Elizabeth Nielsen Mark Steedman Sharon Goldwater
School of Informatics
University of Edinburgh, UK
e.k.nielsen@sms.ed.ac.uk
{steedman, sgwater}@inf.ed.ac.uk

Abstract

Parsing spoken dialogue poses unique difficulties, including disfluencies and unmarked boundaries between sentence-like units. Previous work has shown that prosody can help with parsing disfluent speech (Tran et al., 2018), but has assumed that the input to the parser is already segmented into sentence-like units (SUs), which isn't true in existing speech applications. We investigate how prosody affects a parser that receives an entire dialogue turn as input (a *turn-based model*), instead of gold standard pre-segmented SUs (an *SU-based model*). In experiments on the English Switchboard corpus, we find that when using transcripts alone, the turn-based model has trouble segmenting SUs, leading to worse parse performance than the SU-based model. However, prosody can effectively replace gold standard SU boundaries: with prosody, the turn-based model performs as well as the SU-based model (90.79 vs. 90.65 F1 score, respectively), despite performing two tasks (SU segmentation and parsing) rather than one (parsing alone). Analysis shows that pitch and intensity features are the most important for this corpus, since they allow the model to correctly distinguish an SU boundary from a speech disfluency — a distinction that the model otherwise struggles to make.

1 Introduction

Parsing spoken dialogue poses unique difficulties: spontaneous speech is full of disfluencies, including false starts, repetitions, and filled pauses. In addition, speech transcripts lack punctuation, which would otherwise help signal the boundaries of sentence-like units (SUs).[1] Because of these difficulties, current parsers struggle to accurately parse English speech transcripts, even when they handle other English text well. However, research has shown that prosody can help with at least one of these problems, improving parsing performance for speech that contains disfluencies (Tran et al., 2018, 2019). In this work, we hypothesize that incorporating prosodic features from the speech signal can actually help with both of these problems: not only parsing disfluent speech, but also parsing speech that isn't segmented into SUs.

Other researchers have augmented parsers with prosodic features, but always with the assumption that the parser has access to gold SU boundaries, which cannot be assumed in a deployed speech application. For example, Gregory et al. (2004); Kahn et al. (2005) and Hale et al. (2006) incorporated prosody into statistical parsers or parse rerankers, with mixed results. More recently, Tran et al. (2018) and Tran et al. (2019) found that prosody improved an end-to-end neural parser, with the most significant gains in disfluent sentences. Parsing without access to gold SU boundaries is much more difficult: Kahn and Ostendorf (2012) showed that parsing quality depends on the quality of the sentence segmentation. Furthermore, finding SU boundaries is not as simple as finding long pauses in speech, as we demonstrate below.

We hypothesize that access to prosodic features will help an English parser that has to both parse and correctly identify SU boundaries (which we call *SU segmentation*). We test this hypothesis by inputting entire dialog turns to a neural parser without gold SU boundaries. We call this the *turn-based* model, and compare it to an *SU-based* model, which assumes gold SU boundaries and parses one SU at a time. We use turns as our input unit because they resemble the input a dialog agent would receive from a user. Following Tran et al. (2019) and others, we use a human-generated gold transcript instead of an automatic speech recognition

[1]We follow Kahn et al. (2004) in using the term 'sentence-like units' rather than 'sentences' throughout, since conversational speech doesn't always consist of syntactically complete sentences.

(ASR) transcript; we plan to use ASR output in future work.

We build on the work of Tran et al. (2018) and Tran et al. (2019), considering two different experimental conditions for each model: inputting text features only and inputting both text and prosodic features. Using the Switchboard corpus of English conversational dialogue, we find that when only transcripts are used, the turn-based parser performs considerably worse than the SU-based parser, which is not surprising given that it needs to perform two tasks instead of one. However, when prosodic features are included, there is no difference in performance between the turn-based and SU-based models, and both models outperform the text-only counterparts.

Our primary contributions are:

- We show that a parser that has access to prosody can perform both SU segmentation and parsing as well as a model that only has to parse.
- We show that one difficultly for the prosody-free turn-based model is that it confuses speech disfluencies with SU boundaries, as illustrated in Figure 1. Further analysis indicates that adding pitch and intensity features can help the model to disambiguate the two while pause and duration features do not.

2 Background: prosody and syntax

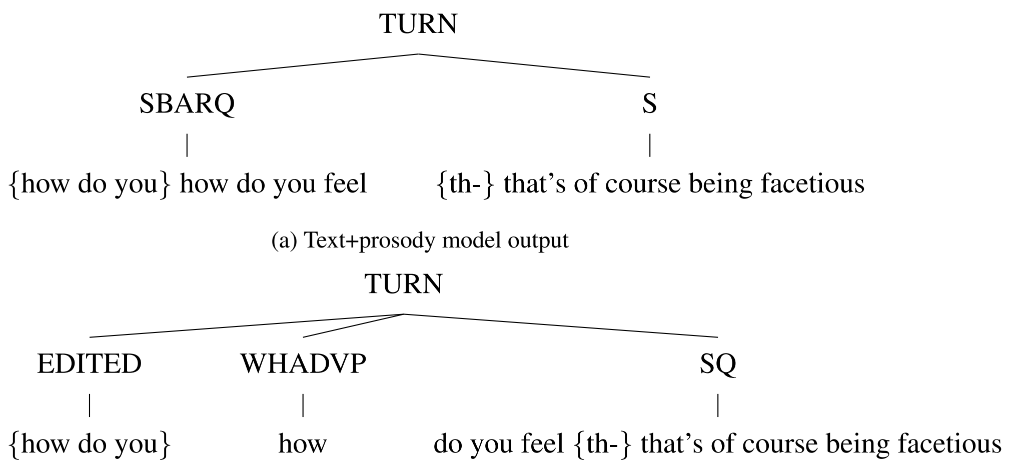
Prosodic signals divide speech into units (Pierrehumbert, 1980). The location and type of these prosodic units are determined by information structure (Steedman, 2000), disfluencies (Shriberg, 2001), and to some extent, syntax (Cutler et al., 1997). Some psycholinguistic research shows that in experimental conditions, speakers can use prosody to predict syntax – for example, that English speakers can use prosody to determine where to attach a modifier or prepositional phrase, or how to correctly group coordinands (e.g., Kjelgaard and Speer (1999); Speer et al. (1996); Warren et al. (1995)). However, Cutler et al. (1997) argues that English speakers often "fail to exploit" this prosodic information even when it is present, so it isn't actually a signal for syntax in practice. Many computational linguists have experimented with this possible link between syntax and prosody by incorporating prosody into syntactic parsers (e.g., Noeth et al. (2000); Gregory et al. (2004); Kahn et al. (2005); Tran et al. (2018)). These models have had mixed success: For example, Gregory et al. (2004) found that prosody was at best a neutral addition to their model, while Kahn et al. (2005) found that prosody helped rerank PCFG output.

One possible reason that prosody is only somewhat effective in previous research is that prosodic units below the level of the SU do not always coincide with traditional syntactic constituents (Selkirk, 1995, 1984).[2] In fact, the only prosodic boundaries that consistently coincide with syntactic boundaries are the prosodic boundaries at the ends of SUs (Wagner and Watson, 2010). The prosodic boundaries at the end of SUs are more distinctive (i.e., tending to correspond to longer pauses and more distinctive pitch and intensity variations) and less likely appear in any other location. These features make prosody a reliable signal for SU boundaries, even though it is an unreliable signal for syntactic structure below the SU level.

Some researchers have used this correlation between prosody and SU boundaries to help in SU boundary detection. Examples of SU segmentation models that found prosodic cues were important include Gotoh and Renals (2000); Kolář et al. (2006); Kahn et al. (2004); Kahn and Ostendorf (2012), who all used traditional statistical models (e.g., HMMs, finite state machines, and decision trees), and Xu et al. (2014), who used a neural model. Kahn et al. (2004) and Kahn and Ostendorf (2012) also looked at downstream parsing accuracy on the same corpus we use. Like us, Kahn and Ostendorf (2012) don't use gold SU boundaries, but direct comparison is impossible because they use ASR output instead of human transcriptions and a different metric for parse performance (SParseval; Roark et al. (2006)). However, they show that having access to gold SU boundaries increases the SParseval score from 78.5 to 82.3, which shows that parsing without gold SU boundaries is difficult.

However, in some research areas, prosody is less frequently used for SU detection. Some ASR corpora and applications segment at relatively arbitrary boundaries such as long silences or even regular intervals (e.g., Jain et al. (2020)). Other applications, such as speech translation, do require syntactically coherent input, but even there, systems targeting SUs have often used only textual features (Sridhar et al., 2013; Wan et al., 2020).

[2] We refer here to traditional constituency parsing; CCG (Steedman and Baldridge, 2011) proposes different syntactic constituents that coincide with prosodic units.


TURN
SBARQ
S
{how do you} how do you feel
{th-} that's of course being facetious
TURN
EDITED
WHADVP
SQ
{how do you}
how
do you feel {th-} that's of course being facetious

(a) Text+prosody model output

(b) Text-only model output

Figure 1: A portion of a turn that contains both disfluencies (shown in curly braces) and an SU boundary. A simplified version of the text+prosody model output is shown in (a), which matches the gold SU boundaries. The text-only model incorrectly places an SU boundary after a disfluency (shown in (b)).

Systems for restoring punctuation from ASR output must identify SU boundaries to correctly insert sentence-final punctuation, but these systems are typically evaluated on rehearsed monologues (such as TED talks) or read speech, which largely lack disfluencies (e.g., Federico et al. (2012)). Here, we show that prosody is primarily helpful for distinguishing SU boundaries from disfluencies, so although some of these systems have used prosody (e.g., Tilk and Alumäe (2016)), text-only systems are very competitive (e.g., Che et al. (2016), Alam et al. (2020)).

Even when SU boundaries are already known, other research in parsing conversational speech has shown that prosody helps identify and correctly handle disfluencies. Tran et al. (2018) found that prosody only modestly affects parsing of fluent SUs, but has a marked effect on disfluent SUs. This accords with other previous work that has found that prosody is helpful in disfluency detection (Zayats and Ostendorf, 2019) We discuss the relationship between prosody and disfluencies in greater detail in Section 6, including how prosody helps the model not to confuse disfluencies and SU boundaries, as shown in Figure 1 above.

3 Task and data

We use the American English corpus Switchboard NXT (henceforth SWBD-NXT) (Calhoun et al., 2010). We choose this corpus mainly so we can compare performance with Tran et al. (2018) and Tran et al. (2019), as well as other earlier probabilistic models such as Kahn et al. (2005). SWBD-NXT comprises 642 dialogues between strangers conducted by telephone. These dialogues are transcribed and hand-annotated with Penn Treebank-style constituency parses. We preprocess the transcripts to remove punctuation and lower-case all letters, making the input more like an ASR transcript as would be used in a deployed application.

The transcript divides the corpus into *SUs* and *turns*. Since these SUs may be sentences or other syntactically independent units such as sentence fragments, we use the generic term 'sentence-like unit' (SU). A turn is a contiguous span of speech by a single speaker. Turns are hand-annotated in SWBD-NXT, but for a deployed dialog agent, a turn is simply whatever contiguous input the user gives. Not all turns in the SWBD-NXT contain more than one SU: of a total 60.1k turns, 35.8k consist of a single SU. The remaining 24.3k contain more than one SU; the majority (52.4 percent) of these contain just two SUs. The average number of SUs per turn is 1.82.

We follow the general approach of Tran et al. (2018), but where they parse a single SU at a time, we give our parser a single dialog turn at a time for our turn-based model. The model returns constituency parses for the turn in the form of Penn Treebank (PTB)-style trees. In order to keep the output in the form of valid PTB trees, we add a top-level constituent, labelled TURN, to all turns, however many SUs they consist of. This example shows how the two sentences in (1) would be fused into a single turn in (2):

(1) Separate SUs:
  a. (S (NP Kim) (VP sings))
  b. (S (NP Sidney) (VP dances))

(2) Merged into a single turn:
  a. (TURN (S (NP Kim) (VP sings)) (S (NP Sidney) (VP dances)))

Of course, using turns instead of SUs leads to longer inputs. We experiment with a pipeline approach (first segmenting turns into SUs, then parsing) as well as an end-to-end approach. In the end-to-end approach, we can't handle extremely long inputs since these longer sequences lead to high memory usage for transformers. We still want to capture the model's behavior on generally longer inputs, so we filter out two problematically long turns from the training set (out of 49,294 turns). We do not have to remove any turns from the development or test sets. This leaves the maximum turn length at 270 tokens. We also remove any turns for which some or all speech features are missing from the corpus.

3.1 Feature extraction

From the speech signal, we extract features for pauses between words, word duration, pitch, and intensity. We largely follow the feature extraction procedure outlined in Tran et al. (2018) and Tran et al. (2019), which we summarize here, noting any deviations from or additions to their procedure.

Pause features are extracted from the time-aligned transcript. Each word's pause feature corresponds to the pause that follows it. Each pause is categorized into one of six bins by length in seconds: $p > 1$, $0.2 < p \leq 1$, $0.05 < p \leq 0.2$, $0 < p \leq 0.05$, $p \leq 0$ (see below), and pauses where we are missing time-aligned data. Following Tran et al. (2018), the model learns 32-dimensional embeddings for each pause category.

Since we use turns instead of SUs, we have to determine how to handle pauses at the beginnings and endings of turns. We decide to calculate pauses based on all words in the transcript, not just the words for a single speaker at a time. This means that at a turn boundary, we calculate the pause as the time between the end of one speaker's turn and the beginning of the other speaker's turn. If one speaker interrupts another, the pause duration has a negative value. We place these negative-valued pauses in the same bin as pauses with length 0.

Duration features are also extracted from the time-aligned transcript. We are interested in the relative lengthening or shortening of word tokens, so we normalize the raw duration of each token. Following the code base for Tran et al. (2019), we perform two different types of normalization. In the first case, we normalize the token's raw duration by the mean duration of every instance of that word type. In the second, we normalize the token's raw duration by the maximum duration of any word in the input unit (SU or turn). These two normalization methods result in two duration features for each word token, which are concatenated and input to the model.

Pitch features (or more accurately, F0 features) are extracted from the speech signal using Kaldi (Povey et al., 2011). These are extracted from 25ms frames every 10ms. Three pitch features are extracted: warped Normalized Cross Correlation Function (NCCF), log-pitch with mean subtraction over a 1.5-second window weighted by Probability of Voicing (POV); and the estimated derivative of the raw log pitch. For further details on these features, see Ghahremani et al. (2014).

Intensity features are also extracted from the speech signal using the same software and frame rate as we use for pitch features. Starting with 40-dimensional mel-frequency filterbank features, we calculate three features: (1) the log of the total energy, normalized by the maximum total energy for the speaker over the course of the dialog; (2) the log of the total energy in the lower half of the 40 mel-frequency bands, normalized by the total energy; and (3) the log of the total energy in the upper half of the 40 mel-frequency bands, normalized by the total energy.

For training, development, and testing, we use the split described in Charniak and Johnson (2001), which is a standard split for experiments on SWBD-NXT (e.g., Kahn et al. (2005); Tran et al. (2018)). The training set makes up 90 percent of the data, and the development and testing sets make up 5 percent each.

4 Model

We use the parser described in Tran et al. (2019), directly extending the code base described in their paper.[3] The model is a neural end-to-end constituency

[3]Original: https://github.com/trangham283/prosody_nlp; our extended code: https://github.com/ekayen/prosody_nlp

parser based on Kitaev and Klein (2018)'s text-only parser, with a transformer-based encoder and a chart-style decoder based on Stern et al. (2017) and Gaddy et al. (2018). This encoder-decoder is augmented with a CNN on the input side that handles prosodic features (Tran et al., 2019). For further description of the model and hyperparameters, see Appendices A.1 and A.2.

The text is encoded using 300-dimensional GloVe embeddings (Pennington et al., 2014).[4] Of the four types of prosodic features described in Section 3, pause and duration features are already token-level. However, pitch and intensity features are extracted from the speech signal at the frame level. In order to map from these frame-level features to a token-level representation, the pitch and intensity features pass through a CNN, and are then concatenated with the token-level pause and duration features.

We follow Tran et al. (2019) in training each model 10 times with different random seeds. For the development set, we report the *mean* of these 10 models' performance. We then select the *median* model by development set performance, and use it to calculate test set results. For any further experiments, such as those discussed in Section 6, we use the random seed for this median model. Each model is trained for 50 epochs and use the epoch with highest development set performance.

In addition to this end-to-end approach, we also report results for a pipeline approach. For the pipeline, we first segment the speech into SUs using a modified version of the parser architecture: We keep the encoder the same, but we change the decoder so that it only does sequence labelling, and we frame the SU segmentation task as a sequence labelling task. We then use the SU-based parser to parse the resulting SUs. We report the model's performance with and without prosodic features during the segmentation and parsing steps.

5 Results

We compare the turn-based F1 performance of our parser to a replication of the SU-based performance described in Tran et al. (2018) and Tran et al. (2019). Table 1 shows the development and test set results.[5] We find that the turn-based model benefits significantly from prosody. The turn-based model performs equivalently well to the SU-based model, despite doing two tasks instead of one. The SU-based model also improves by 0.36 in F1 score on the test set with the addition of prosody. Note that while prosody has a considerably larger effect on the turn-based model than on the SU based model, the exact size of this change will depend on the corpus. For example, in a corpus with very few multi-SU turns, the performance change in the turn-based model might not be as large. However, our results suggest that prosody helps when a model needs to both detect SU boundaries and parse SUs.

The biggest difference between the SU- and turn-based models' performance on this corpus is in the text-only scenario, where the turn-based parser is substantially worse. This is expected for a few reasons. First, the text-only turn-based parser encounters longer inputs. Longer inputs tend to lead to more parse errors simply because there are more ways to parse a longer string. Table 2 shows this correspondence between length and performance. The median length of turns in the development set is 9 tokens, while the median length of SUs is 6 tokens. Longer strings are also more likely to contain the things that make parsing difficult, namely disfluencies and SU boundaries.

The turn-based parser's task is also more com-

	SU-based	**Turn-based**
Test set:		
Text only	90.29	86.56
Text+prosody	90.65	90.79
Dev. set:		
Text only	90.31	86.08
Text+prosody	90.90	90.84

Table 1: Test and development set F1 of the turn-based model compared to the SU-based model. Dev. set scores are the mean over 10 random seeds. For the test set, we use the model that has the median dev. set performance out of 10 randomly seeded models.

	Input length (# tokens)			
	1	**2–8**	**9–22**	**23–255**
Text only	98.36	93.0[illegible]	89.2[illegible]	84.30
Text+pros.	99.18	94.9[illegible]	92.74	89.80
Δ	0.82	[illegible].91	[illegible].52	5.5

Table 2: F1 performance of the text-only and text+prosody turn-based models on inputs of various lengths in the development set. The inputs are divided into bins of approximately equal size by token length.

[4] See Appendix A.3 for results using BERT embeddings.

[5] We use PyEvalb to evaluate our parser's performance, though we modify it so that it behaves identically to Evalb: https://github.com/ekayen/PYEVALB

		Segmentation			Parsing
		Precision	**Recall**	**F1**	**F1**
Pipeline	Text only	78.84	68.61	73.31	82.73
	Text+prosody	99.96	99.45	99.71	90.89
E2E	Text only	55.01	75.78	63.74	86.09
	Text+prosody	99.41	99.41	99.41	90.90

Table 3: Development set performance of the pipeline model on segmentation and parsing as compared to the end-to-end model. (Results are from single models rather than an average as in Table 1.)

plex: it has to perform both SU segmentation and parsing, rather than parsing alone. This gives the turn-based parser novel ways to make errors by splitting a turn into the wrong number of SUs. However, prosody brings the turn-based parser up to the level of the SU-based parser, even though the turn-based model's task is more complex. Table 5 shows how the text-only parser significantly overestimates the number of SU boundaries. Without prosody, the model achieves an F1 score of 63.74 on SU prediction on the development set, compared to 99.41 with prosody (see Table 3). The most comparable work on SWBD is Kahn and Ostendorf (2012), who achieved 78 F1 using a hidden-event model, where we use a much more powerful transformer model; however, their model used ASR transcripts as input, so these scores aren't directly comparable.

We also test the pipeline model described in Section 4, which first segments turns into SUs and then parses them, both with and without prosody. We train just one segmentation model with the same random seed as the median development set model. We report the development set performance on segmentation (measured by segmentation F1 (Makhoul et al., 200[illegible]) and parse F1 in Table 3.

The text+prosody pipeline model achieves an F1 score of 99.71, which is statistically indistinguishable from the end-to-end text+prosody model. In both cases, we see that the addition of prosody boosts SU segmentation accuracy to near-perfect levels, which explains why the parser performance is similar (and much better than without prosody).

Comparing the two text-only models reveals a more interesting pattern: while the pipeline model achieves much better segmentation F1, its parsing performance is worse. This is unexpected, as parsing and segmentation performance are usually correlated. This effect seems to arise because the two models err in different directions on segmentation: The pipeline model under-segments turns (corresponding to higher segmentation precision), while the end-to-end over-segments (higher recall, substantially lower precision). When it over-segments, the end-to-end text-only model often splits a word or short constituent off of an otherwise well-formed SU subtree; by contrast, the pipeline model tends to leave two or more SUs combined and then to generate many SU-internal parsing errors. These SU-internal parsing errors include more coordination errors as well as VP, NP and clause attachment errors than the end-to-end model.[6] However, the pipeline model does as well as the end-to-end model at PP attachment and modifier attachment.

Overall, these results show that a pipeline model can be as effective at parsing as an end-to-end one, but that including prosody is even more important for a pipeline model. Since we care about parsing performance and the end-to-end text-only model does much better at parsing, we use the end-to-end model for all remaining analyses.

5.1 Error types

We use the Berkeley Parser Analyser (Kummerfeld et al., 2012) to determine what types of errors each of the SU-based and end-to-end turn-based models makes. Figure 2 summarizes the output of the Analyser. Overall, the SU-based parser shows only small effects from prosody, but the turn-based model does significantly worse on certain error types without prosody. Even for the turn-based model, prosody only affects error types that have to do with the shape of the tree. The *different label* category shows errors where two identically shaped trees have different constituent labels, and prosody has no effect on these.

For the turn-based model, poor SU segmentation by the text-only model explains some of the differences between the text+prosody and text-only models. Since 68.8 percent of SUs are clauses (i.e.,

[6] We use the Berkeley Parser Analyser to analyze types of parse error (Kummerfeld et al., 2012).

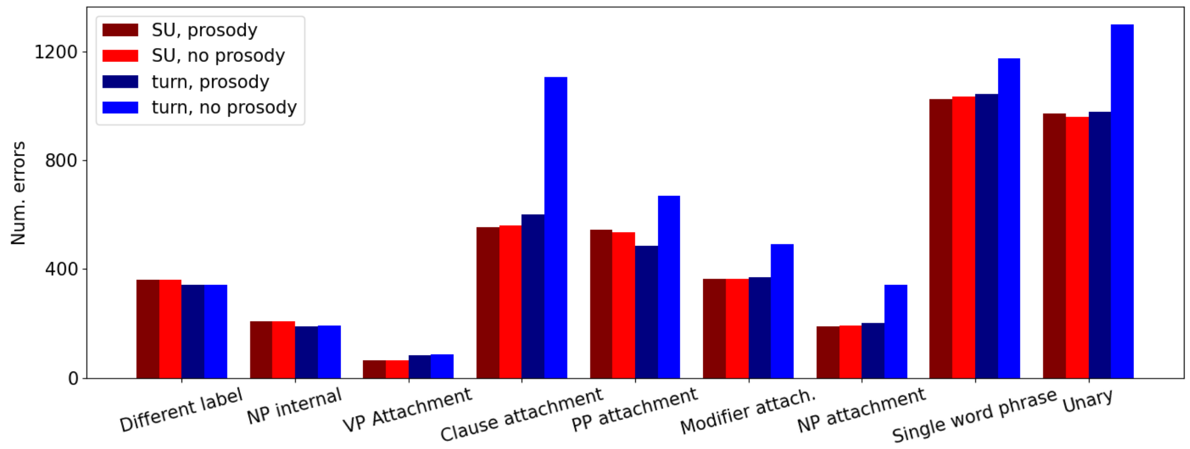


Figure 2: Prevalence of various error types in the development set output, given four different experimental conditions: SU-based, with and without prosody; and turn-based, with and without prosody. Error types are classified by the Berkeley Parser Analyzer (Kummerfeld et al., 2012).

they have a top node of type S, SBAR, SQ or SINV), an incorrect SU segmentation is usually classed as a clause attachment error. An example of this kind of attachment error can be seen in Appendix A.4. However, prosody also affects the turn-based model's rate of NP, PP, and modifier attachment errors. Since these attachment errors are not as common in the text-only SU-based model, it seems likely that they are caused by a cascade effect from errors in top-level SU segmentation. Prosody also affects the turn-based model's rate of *unary* errors (which are errors "involving unary productions that are not linked to a nearby error such as a matching extra or missing node") and *single word phrase* errors (which are "a range of node errors that span a single word" but which are not related to other errors) (Kummerfeld et al., 2012). Finally, very modest differences are seen for two rare error types: *NP-internal* and *VP attachment* errors.

5.2 Effect of disfluencies

	[illegible]	Fluent	Disfluent
Text only	86.09	89.89	84.25
Text+all prosody	90.90	93.63	89.58
Δ from prosody	4.81	3.74	5.33
Pitch only	90.71	93.46	89.37
Intensity only	90.29	93.30	88.83
Duration only	86.24	89.94	84.44
Pause only	86.21	90.09	84.32

Table 4: F1 for the text and text+prosody turn-based models when tested on the entire development set, the subset of the development set consisting of only fluent turns, and the subset of all disfluent turns.

Our turn-based model performs worse overall on disfluent turns than on fluent turns, which was also true of Tran et al. (2018)'s SU-based model. Prosody also leads to a greater gain in F1 for disfluent turns than for fluent turns. These differences in performance are shown in Table 4. The lower performance on disfluent sentences may be at least partially attributable to length differences: the median length of turns with disfluencies is 28 tokens, compared to 3 tokens for fluent turns, where we define a disfluent turn as any turn containing the constituent tag EDITED. As discussed in Section 5, longer input generally leads to more parser errors, meaning that disfluent sentences are more likely to cause parser errors. However, there are other reasons disfluencies are difficult for the turn-based model, as discussed in the following section.

6 Distinguishing disfluencies and SU boundaries

One effect of disfluencies is that the text-only model tends to confuse certain kinds of disfluencies for SU boundaries, as illustrated in Figure 1. Table 5 shows that the text+prosody model largely avoids this confusion, and indeed can do so almost as well using only pitch or intensity features. However, models using only pause or duration features are not good at distinguishing disfluencies from SU boundaries and predict boundaries too often. These results largely concur with previous work describing the similarities and differences between prosodic features of disfluencies and SU boundaries (Shriberg, 2001; Wagner and Watson, 2010). In this section, we examine each of the features

Features	Total predicted bound.	Predicted bound. at disf.
Gold	2552	2
All	2552	4
Pitch	2590	17
Intensity	2647	20
Duration	3437	204
Pause	3648	225
None	3516	208

Table 5: The total number of SU boundaries predicted on the dev. set as compared to the number of SU boundaries predicted to fall at what are actually interruption points within disfluencies. The first line shows the target for both values. We give results for a model with all four prosodic features, models with only one prosodic feature at a time, and a model with no prosodic features.

more closely with respect to this previous work and our results, highlighting where our results do (and do not) accord with expectations.

The disfluencies that are relevant to this discussion include repetitions and restarts. Examples of these from SWBD-NXT are shown here, with bracketing added for clarity:

(3) **Spurious repetition:** it [may] may be at this point
Restart: [but it's] but I think it's relatively unimportant

In these examples, the text in square brackets is called the *reparandum*, which is immediately followed by the *interruption point*. Disfluencies in SWBD-NXT are marked in the constituency parse annotation, where the reparandum is marked as a constituent with the label EDITED. The interruption point is the right edge of this constituent.

Our analysis draws on the work of Shriberg (2001), who described the prosodic features of the interruption point and the reparandum based on an analysis of three English conversational and task-based dialogue corpora — the Switchboard Corpus (which we use a subset of), ATIS (Hirschman, 1992), and AMEX (Kowtko and Price, 1989).

Pauses. Although pauses may be the most intuitive potential cue to SU boundaries, previous work suggests that long pauses also characterize interruption points (Wagner and Watson, 2010; Shriberg, 2001). Indeed, our analysis shows that longer pauses ($> 0.05s$) are over-represented in both locations. If pause types were distributed uniformly, 16 percent of both SU boundaries and interruption points would have a longer pause. Instead, we find that 33 percent of SUs boundaries and 37 percent of interruption points have such pauses. This explains why the pause-only model tends to confuse SU boundaries and interruption points.

Duration. Shriberg (2001) found that both interruptions and SU boundaries are associated with lengthening of the immediately preceding syllable. Lengthening before the interruption point may occur even if there are no other prosodic cues to the disfluency, and can be "far greater" than at SU boundaries (Shriberg, 2001, 161). This type of lengthening is captured by our first duration feature, which measures the token duration normalized by the mean duration for its word type. Like Shriberg (2001), we find that words preceding SU boundaries are lengthened on average (normalized duration: 1.18), and those preceding interruption points even more so (normalized duration: 1.41). In principle, this extra lengthening could help the duration-only model distinguish SU boundaries from interruptions, but in practice the model is nearly as bad at distinguishing them as the text-only model.

The second duration feature is the token length normalized by the maximum length of any token in the input, to normalize for speaking rate. Initially, this feature looks helpful: SU-final words have mean value of 0.86, while words directly before the interruption point have a mean of 0.50. However, the feature mainly captures the number of phones in a word, since words with fewer phones — including English function words — tend to have shorter normalized duration. It turns out that function words occur more often before interruption points than before SU boundaries: using NLTK's stopwords as a heuristic for function words, only 21.9 percent of development set SUs end in a function word, while the word before an interrutption point is a function word 51.6 percent of the time (Bird and Klein, 2009). Since the second duration feature captures a lexical distinction that is already signalled in the text, it cannot help the duration-only model outperform the text-only model.

Pitch. Based on previous work, our finding that pitch features are useful is not a surprise: the pitch contour before an interruption point is generally "flat or slowly falling" (Shriberg, 2001, 161), while SU boundaries are characterized by a *boundary tone*, generally corresponding to a fall or rise. Our model may be able to learn such temporal patterns, but even just looking at static pitch features reveals differences between boundaries and interruptions for two of the three features. In particular, the mean warped NCCF value for pre-interruption point words is significantly higher than the value for SU-final words ($p < 0.001$), though somewhat lower than the overall average value across the development set. Meanwhile, the log-pitch with POV-weighted mean subtraction is significantly lower at interruption points than at SU boundaries ($p < 0.01$). These differences allow the pitch-only model to distinguish SU boundaries and interruption points much better than the pause- or duration-only models can (see Table 5). Of these two pitch features, log-pitch is a more direct indicator of fundamental frequency (F0), which suggests that average perceived pitch is likely lower before disfluencies than before SU boundaries. There could be several reasons for this difference. For example, it could be that the "flat or slowly falling" tone of disfluencies that Shriberg (2001) describes has a lower average value than SU boundaries which can have either a fall *or* a rise (e.g., for certain kinds of questions). However, examining pitch features across the whole corpus obscures more subtle distinctions such as different types of pitch contours.

Intensity. We find that intensity features alone are enough to distinguish SU boundaries from interruption points, which is interesting because intensity has not been previously identified as an important cue: Shriberg (2001) doesn't note any particularly distinctive intensity features of the reparandum or interruption point, and work by Liu et al. (2006) on the Switchboard Corpus suggests that SU boundaries are correlated to lower intensity in some speakers, but that this isn't consistent across speakers. The three intensity features correspond to overall energy, energy in the lower half of frequencies, and energy in the higher frequencies. SU-final words have a significantly higher mean value for lower-frequency intensity than all other words ($p < 0.001$), while words before the interruption point do not. This systematic difference in one intensity feature seems to be part of how intensity features allow the model to consistently tell SU boundaries apart from disfluencies.

Overall performance. Given our claim that the main issue facing the text-only turn-based parser is distinguishing disfluencies from SU boundaries, it is not surprising that the two features that do best at this, pitch and intensity, also yield the highest overall performance. Results are shown in Table 6.

Features		F1
All features		90.90
Only	Pitch	90.71 (ns)
	Intensity	90.29 (*)
	Duration	86.24 (*)
	Pause	86.21 (*)
No prosodic features		86.09 (*)

Table 6: Results of ablation testing, measured by F1 score on the dev. set. Asterisks indicate a statistically significant difference ($p < 0.001$) from the model with all features. The first row shows with all features; the next four rows show the result with one feature at a time; the final row shows the result with no prosody.

7 Conclusion

Our experiments show that parsing English speech transcriptions without gold SU boundaries is difficult for our parser: its F1 score drops by about 4 percentage points compared to a model with gold SU boundaries. Incorrect SU segmentation causes a large part of this damage, though other errors in tree construction also play a role. We show that we can undo this damage by giving our parser prosodic information. Importantly, prosody helps by allowing the parser to distinguish disfluencies from SU boundaries. These results argue for giving prosodic information to parsers in deployed applications, where no SU boundary annotations are available, including dialog agents.

Furthermore, our experiments show that even limited prosodic features help a great deal: for our English data, pitch information alone is not significantly worse than pitch, intensity, pause, and word duration information combined. This means that incorporating the right kind of prosodic information can potentially lead to significant gains.

Acknowledgments

We are very grateful to Trang Tran for help answering questions related to her code and to Mari Ostendorf for conversations that helped inspire this paper. We would like to thank Korin Richmond, the ACL reviewers, and members of the AGORA research group at the University of Edinburgh for their feedback. This work was supported by funding from Huawei and the project SEMANTAX, which received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 742137).

References

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. Punctuation restoration using transformer models for high-and low-resource languages. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 132–142, Online. Association for Computational Linguistics.

Edward Loper Bird, Steven and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Sasha Calhoun, Jean Carletta, Jason 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*, 44:387–419.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Xiaoyin Che, Cheng Wang, Haojin Yang, and Christoph Meinel. 2016. Punctuation prediction for unsegmented transcript based on word vector. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Anne Cutler, Delphine Dahan, and Wilma van Donselaar. 1997. Prosody in the comprehension of spoken language: A literature review. *Language and Speech*, 40(2):141–201.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcello Federico, Sebastian Stüker, Luisa Bentivogli, Michael Paul, Mauro Cettolo, Teresa Herrmann, Jan Niehues, and Giovanni Moretti. 2012. The iwslt 2011 evaluation campaign on automatic talk translation. In *International Conference on Language Resources and Evaluation (LREC)*, pages 3543–3550.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur. 2014. A pitch extraction algorithm tuned for automatic speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2494–2498.

Yoshihiko Gotoh and Steve Renals. 2000. Sentence boundary detection in broadcast speech transcripts. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*.

Michelle Gregory, Mark Johnson, and Eugene Charniak. 2004. Sentence-internal prosody does not help parsing the way punctuation does. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistic*.

John Hale, Izhak Shafran, Lisa Yung, Bonnie J. Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover, and Robin Stewart. 2006. PCFGs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 161–168. Association for Computational Linguistics.

Lynette Hirschman. 1992. Multi-site data collection for a spoken language corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, page 7–14, USA. Association for Computational Linguistics.

Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf. 2020. Contextual RNN-T for Open Domain ASR. In *Proc. Interspeech 2020*, pages 11–15.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, page 233–240, USA. Association for Computational Linguistics.

Jeremy G. Kahn and Mari Ostendorf. 2012. Joint reranking of parsing and word recognition with automatic segmentation. *Computer Speech and Language*, 26(1):1 – 19.

Jeremy G. Kahn, Mari Ostendorf, and Ciprian Chelba. 2004. Parsing conversational speech using enhanced segmentation. In *Proceedings of HLT-NAACL 2004*, pages 125–128, Boston, Massachusetts, USA. Association for Computational Linguistics.

Heejin Kim, Tae jin Yoon, Jennifer Cole, and Mark Hasegawa-johnson. 2006. Acoustic differentiation of l- and l-l% in Switchboard and radio news speech. In *Proceedings of Speech Prosody 2006*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Margaret M. Kjelgaard and Shari R. Speer. 1999. Prosodic facilitation and interference in the resolution of temporary syntactic closure ambiguity. *Journal of Memory and Language*, 40(2):153 – 194.

Jáchym Kolář, Elizabeth Shriberg, and Yang Liu. 2006. Using prosody for automatic sentence segmentation of multi-party meetings. In *International Conference on Text, Speech and Dialogue*, pages 629–636. Springer.

Jacqueline C. Kowtko and Patti J. Price. 1989. Data collection and analysis in the air travel planning domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15-18, 1989*.

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 *Proceedings of the 2012 Joint Conference on EMNLP and CoNLL*, pages 1048–1059, Jeju Island, South Korea.

John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. 2000. Performance measures for information extraction. *Proceedings of DARPA Broadcast News Workshop*.

Elmar Noeth, Anton Batliner, Andreas Kießling, Ralf Kompe, and Heinrich Niemann. 2000. Verbmobil: The use of prosody in the linguistic components of a speech understanding system. *IEEE Transactions on Speech and Audio processing*, 8(5):519–532.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Janet Breckenridge Pierrehumbert. 1980. *The phonology and phonetics of English intonation*. Ph.D. thesis, Massachusetts Institute of Technology.

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 *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

Brian Roark, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy Kahn, Yang Liu, Mari Ostendorf, John Hale, Anna Krasnyanskaya, Matthew Lease, Izhak Shafran, Matthew Snover, Robin Stewart, and Lisa Yung. 2006. SParseval: Evaluation metrics for parsing speech. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*.

Elisabeth Selkirk. 1984. *Phonology and Syntax*. MIT Press, Cambridge, MA.

Elisabeth Selkirk. 1995. Sentence prosody: Intonation, stress, and phrasing. *The handbook of phonological theory*, 1:550–569.

Elizabeth Shriberg. 2001. To 'errrr' is human: Ecology and acoustics of speech disfluencies. *Journal of the International Phonetic Association*, 31:153 – 169.

Shari Speer, Margaret Kjelgaard, and Kathryn Dobroth. 1996. The influence of prosodic structure on the resolution of temporary syntactic closure ambiguities. *Journal of psycholinguistic research*, 25:249–71.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 230–238, Atlanta, Georgia. Association for Computational Linguistics.

Mark Steedman. 2000. Information structure and the syntax-phonology interface. *Linguistic inquiry*, 31(4):649–689.

Mark Steedman and Jason Baldridge. 2011. Combinatory Categorial Grammar. In Robert Borsley and Kersti Börjars, editors, *Non-Transformational Syntax: A Guide to Current Models*, pages 181–224. Blackwell, Oxford.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech 2016*, pages 3047–3051.

Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. 2018. Parsing speech: a neural approach to integrating lexical and acoustic-prosodic information. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, Volume 1 (Long Papers)*, pages 69–81, New Orleans, Louisiana. Association for Computational Linguistics.

Trang Tran, Jiahong Yuan, Yang Liu, and Mari Ostendorf. 2019. On the Role of Style in Parsing Speech with Neural Models. In *Proc. Interspeech 2019*, pages 4190–4194.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Michael Wagner and Duane G. Watson. 2010. Experimental and theoretical advances in prosody: A review. *Language and Cognitive Processes*, 25(7-9):905–945.

David Wan, Zhengping Jiang, Chris Kedzie, Elsbeth Turcan, Peter Bell, and Kathy McKeown. 2020. Subtitles to segmentation: Improving low-resource speech-to-TextTranslation pipelines. In *Proceedings of the workshop on Cross-Language Search and Summarization of Text and Speech*, pages 68–73, Marseille, France. European Language Resources Association.

Paul Warren, Esther Grabe, and Francis Nolan. 1995. Prosody, phonology and parsing in closure ambiguities. *Language and Cognitive Processes*, 10(5):457–486.

Chenglin Xu, Lei Xie, Guangpu Huang, Xiong Xiao, Eng Siong Chng, and Haizhou Li. 2014. A deep neural network approach for sentence boundary detection in broadcast news. In *INTERSPEECH-2014*, pages 2887–2891.

Vicky Zayats and Mari Ostendorf. 2019. Giving attention to the unexpected: Using prosody innovations in disfluency detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, Volume 1*, pages 86–95, Minneapolis, Minnesota. Association for Computational Linguistics.

A Appendices

A.1 Model description

The parser is an encoder-decoder model that takes both speech and text inputs. In this appendix, we describe the three main model components: the CNN that processes the continuous speech inputs before they reach the encoder, the transformer-based encoder, and the chart-style decoder.

A.1.1 The speech-processing CNN

Of the four prosodic features, pause and duration are already discrete at the token level. Pitch and intensity, however, are extracted from frames every 10 ms in the original speech signal. If a given token is shorter than a fixed number of frames, some frames of left and right context are included; frames from longer tokens are subsampled to reduce their frame length. These two frame-based features features have a different dimensionality than the token-level input and they are untenably long for a sequence model or transformer. The CNN solves both these problems by producing a fixed-length representation for each feature at the token level. This representation can be concatenated with the other token-level features and input to the encoder.

For a speech input with f frames, the raw features input to the CNN have dimensions $6 \times f$, where 6 is the number of total features for each frame (3 pitch features and 3 intensity features). Several filters of different sizes then perform one-dimensional convolution of the input. These different filters allow the CNN to integrate information on various time scales. We apply N of each of these m filters, for a total of mN filters. We use the hyperparameters described by Tran et al. (2018): N = 32 filters of widths w = [5, 10, 25, 50], for a total of mN = 128 filters. The output of each filter is then max-pooled, which converts the features for a given token to a uniform dimension.

These CNN-processed features are then concatenated with the token-level prosodic features (pause and duration) and the text embedding of the token, and then input to the encoder. The CNN is trained along with the encoder-decoder model.

A.1.2 The encoder

The encoder is a standard transformer with eight attention heads based on the work of Kitaev and Klein (2018). For each word of input x_i, the transformer encoder produces a representation of the forward context, $\overrightarrow{y_i}$, and the backward context $\overleftarrow{y_i}$. We represent a given span between indices i and j by subtracting the forward representations and backward representations and concatenating the results:

$$v_{(i,j)} = [\overrightarrow{y_j} - \overrightarrow{y_i}; \overleftarrow{y_j} - \overleftarrow{y_i}]$$

The next section explains how we use this span representation $v_{(i,j)}$ to generate scores for constituents in a tree.

A.1.3 The decoder

The decoder is a chart-style span-based decoder. Its goal is to output the correct tree T for an input $x_1, ..., x_n$. Each tree's score $S(T)$ is simply the sum of the scores of its constituents, where each constituent is defined by a start index i, an end index j, and a label l.

$$S_{tree}(T) = \sum_{i,j,label \in T} S_{label}(i,j,l) + S_{span}(i,j)$$

As this formula for tree score shows, each constituent's score is made up of a label score and span score. Conceptually, the span score corresponds to the probability that a constituent exists that exactly covers span (i, j) in the input; the label score reflects the probability that the span (i, j) has a given constituent label (e.g., S, NP). The decoder must have a way of determining the label score and span score for each constituent.

The label scores are generated by passing the span representation $v_{(i,j)}$ through a two-layer feed-forward network like the feed-forward networs Vaswani et al. (2017) use:

$$FFN(x) = W_2(relu(W_1x + b_1)) + b_2$$

Following Kitaev and Klein (2018), we also include a layer normalization step ($LNorm$). This feed-forward network produces a vector for each span $S_{label}(i, j)$ whose size is the number of possible labels:

$$S_{label}(i, j) = M_2(relu(LNorm(M_1v_{(i,j)})+c_1))+c_2$$

The lth element of this vector is the score for the label l:

$$S_{label}(i, j, l) = [S_{label}(i, j,)]_l$$

We also need to calculate the span score, but calculating the score for all spans (i, j) would be prohibitively inefficient. Instead, Kitaev and Klein (2018), following the approach of Stern et al. (2017) and Gaddy et al. (2018), use a dynamic programming strategy based on the CKY algorithm. The score for a span (i, j) is calculated in terms of the scores of its subspans, which allows span scores to be built up recursively from the stored scores of smaller spans. A given span (i, j) can be split at any internal point into two subspans, (i, k) and (k, j). Each of these possible splits (i, k, j) is assigned a score, calculated by summing the span scores of the subspans:

$$S_{split}(i, k, j) = S_{span}(i, k) + S_{span}(k, j)$$

Then, to find the best score for this span (i, j), we find the label and split that maximize the following sum:

$$S_{best}(i, j) = \max_{l,k}[S_{label}(i, j, l) + S_{split}(i, k, j)]$$

All spans are recursively split into subspans, eventually arriving at single-word spans. Since there are no splits possible for a single-word span, the score for a single word span is simply that word's best label score:

$$S_{best}(i, i + 1) = \max_{l}[S_{label}(i, i + 1, l)]$$

This method requires that the grammar be in Chomsky-Normal form, which the model achieves by collapsing strings of unary rules and using dummy nodes to make n-ary rules into binary rules.

With this method of generating tree scores from span representations, we can then define the hinge loss for our predicted tree $\hat{T}$ compared to the gold tree $T*$, where Δ represents the Hamming loss on labeled spans:

$$\begin{aligned} Loss(\hat{T}, T*) = & \\ & \max[0, \max_{T}[\Delta(\hat{T}, T*) + S_{tree}(\hat{T})] \\ & \qquad - S_{tree}(T*)] \end{aligned}$$

We then use this loss function to train our encoder-decoder, including the CNN input module for speech.

A.2 Model training details

We used the hyperparameters specified in (Tran et al., 2019)'s code base, documented in Table 7. Each model was trained for 50 epochs on a single Nvidia GTX 1080 GPU, which took approximately 7 hours per model. The text-only models have approximately 23M trainable parameters each, while the text+prosody models have approximately 20M trainable parameters.

Hyperparameter	Value
Epochs	50
Text embedding dim.	300
Max. seq. length	270
Dropout	0.3
Num. layers	4
Num. heads	8
Model dim.	1536
Key/value dim.	96

Table 7: Model hyperparameters. Note that the maximum sequence length for the SU-based model is 200 tokens.

A.3 Incorporating BERT

We include here the results for both the SU- and turn-based parsers when given BERT embeddings (Devlin et al., 2019) in place of GloVE embeddings (Pennington et al., 2014). We train one model for each experimental condition, using the random seed we used to generate the results shown in Table 1. We see in Table 8 that BERT improves the

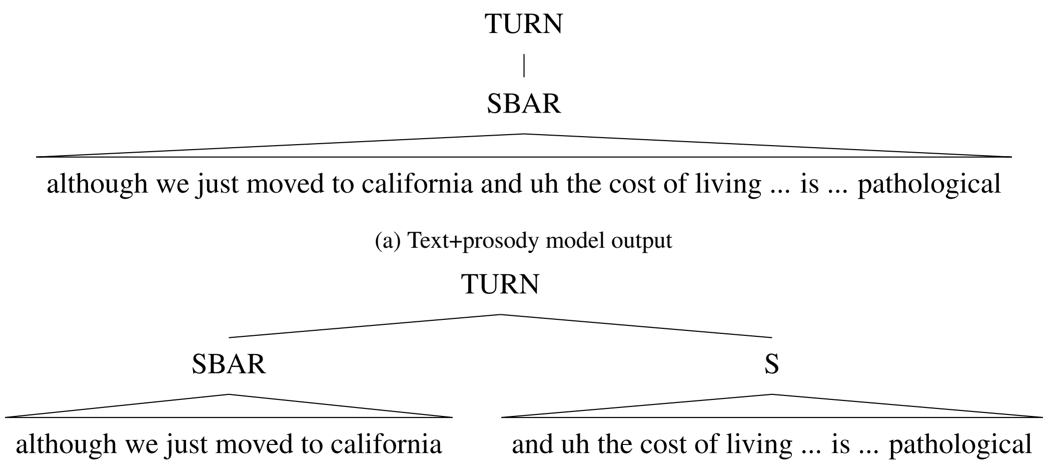

TURN
SBAR
although we just moved to california and uh the cost of living ... is ... pathological
(a) Text+prosody model output
TURN
SBAR
S
although we just moved to california
and uh the cost of living ... is ... pathological
(b) Text-only model output

Figure 3: An example of a clause attachment error. The tree shown in (a) is correctly parsed as a single SU by the text+prosody model, whereas the text-only model incorrectly segments this into two SUs, as shown in (b). This example is taken from the development set and slightly simplified for space (shown by ellipses).

performance in all experimental conditions. The SU-based text+prosody parser does outperform the turn-based parser by a statistically significant margin, though this result was obtained on just one model instead of 10 randomly seeded models. However, the turn-based parser's performance remains quite close to the SU-based parser's despite having a more difficult task to perform, and otherwise the basic pattern from the GloVE results holds here.

	SU-based	Turn-based
Text only	91.90	88.[illegible]
Text+prosody	92.77	92.12

Table 8: Development set F1 when using BERT embeddings, comparing the turn-based model to the SU-based model.

A.4 Clause attachment illustration

Figure 3 illustrates an example of an error classified as a clause attachment error by the Berkeley Parser Analyser (Kummerfeld et al., 2012).