INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Prosodic features improve sentence segmentation and parsing in English

Elizabeth Nielsen *Mark Steedman* *Sharon Goldwater*

School of Informatics
University of Edinburgh, UK
e.nielsen@ed.ac.uk, {steedman, sgwater}@inf.ed.ac.uk

Abstract

Parsing spoken dialogue presents challenges that parsing text does not, including a lack of clear sentence boundaries. We know from previous work that prosody helps in parsing single sentences [1], but we want to show the effect of prosody on parsing speech that isn't segmented into sentences. In experiments on the English Switchboard corpus, we find prosody helps our model both with parsing and with accurately identifying sentence boundaries. However, we find that the best-performing parser is not necessarily the parser that produces the best sentence segmentation performance. We suggest that the best parses instead come from modelling sentence boundaries jointly with other syntactic boundaries.

1. Introduction

Parsing spoken dialogue poses unique difficulties, including speech disfluencies and a lack of defined sentence boundaries. Because of these difficulties, current parsers struggle to accurately parse English speech transcripts, even when they handle other English text well. Research has shown that prosody can improve parsing performance for speech that is already divided into sentence-like units (SUs)[1, 2].[1]

In this work, we hypothesize that prosodic features from the speech signal will help with parsing speech that *isn't* segmented into SUs, by improving the parser's ability to find SU boundaries. We test this hypothesis by inputting entire dialog turns to a neural parser without SU boundaries. These turns resemble the input a dialog agent would receive from a user. We try two approaches: an end-to-end model that jointly segments and parses input, and a pipeline model that first segments and then parses the input. To our knowledge, there hasn't been previous research on combining SU segmentation and parsing into a single task. Following Tran et al.[1, 2] (henceforth, T18 and T19), we consider two experimental conditions for each model: inputting text features only, and inputting both text and prosodic features extracted directly from the audio signal. We also follow them in using the Switchboard corpus of English conversational dialogue [4].

Although overall parse scores are lower for parsers that don't have access to gold standard SU boundaries, our main hypothesis holds: that parsers using both text and prosodic features are more accurate than those using text alone. Unsurprisingly, the end-to-end model performs parsing better than the pipeline model because it doesn't suffer from error propagation. We expected to find that gains in parsing quality would come primarily because models with access to prosody would perform SU segmentation better. We do find that prosody helps all models improve their SU segmentation. However, the pipeline model produces much better segmentation scores than the end-to-end model, and yet it still does worse at parsing. In Section 5, we discuss why segmentation and parsing quality do not always correlate in this task. However, even though the best parses and segmentations are not always produced by the same model, all models perform better at both tasks with prosodic information.

Our primary contributions are:

- We build an end-to-end model that jointly performs SU segmentation and parsing.
- We show that prosodic features are helpful for both SU segmentation and parsing, whether using an end-to-end or pipeline model.
- We show that an end-to-end model performs parsing better than a pipeline model, specifically because the end-to-end model is able to model SU boundaries jointly with other constituent boundaries.

2. Background: prosody and syntax

Prosodic signals divide speech into units [5]. The location and type of these prosodic units are determined by information structure [6], disfluencies [7], and to some extent, syntax [8]. Some psycholinguistic research shows that in experimental conditions, speakers can use prosody to predict syntax (e.g., [9]). However, Cutler et al. [8] argue 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, which improves performance in some cases, but not all (e.g., [10, 11, 12, 1]).

Prosody's mixed record may be because units below the SU don't always coincide with traditional syntactic constituents [13, 14]. In fact, the only prosodic boundaries that consistently coincide with syntactic boundaries are the ends of SUs [15]. Prosodic boundaries at the end of SUs are more distinctive, with longer pauses and more distinctive pitch and intensity variations, making prosody a reliable signal for SU boundaries, but less so for lower-level syntactic structure.

Some researchers have used prosody to help in SU boundary detection. Examples of SU segmentation models that benefit from prosody include [16, 17, 3, 18], who all used traditional statistical models (e.g., HMMs, finite state machines, and decision trees), and [19], who used a neural model.

[1] We follow Kahn et al. (2004) [3] in using the term 'sentence-like units' rather than 'sentences' throughout, since conversational speech doesn't always consist of syntactically complete sentences.

10.21437/Interspeech.2023-952

3. Task and model

3.1. Task

In order to test our hypothesis that prosody is helpful for both SU segmentation and parsing, we design a model to parse whole dialog turns, rather than just one SU at a time. Our parser follows the general design of T19, but because it parses turns, it must perform both SU segmentation and parsing. We approach this task in two ways: an end-to-end model (SU-segmentation and parsing done jointly) and a pipeline model (SU-segmentation done before parsing). Both models return constituency parses for each turn in the form of Penn Treebank (PTB)-style trees. In order to keep the output in the form of valid PTB trees for the end-to-end-model, we add a top-level constituent, labelled TURN, to all turns, however many SUs they consist of. As we discuss in Section 6, this innovation allows the end-to-end model to treat SUs in the same way that it treats other syntactic units.

3.2. Model architecture

For both the end-to-end model and the pipeline parser, we use T19's parser, extending the code base described in their paper.[2] The model is a neural constituency parser based on [20]'s text-only parser, with a transformer-based encoder and a chart-style decoder based on [21] and [22]. The text is encoded using 300-dimensional GloVe embeddings [23].

Acoustic features corresponding to pitch, intensity, and pause and word duration are added to the text input by concatenating them directly with each token embedding. While the pause and word duration features are token-level in their raw form, the pitch and intensity features are extracted from frames every 10 ms in the speech signal (see Section 4). In order to be able to concatenate all acoustic features with token embeddings, T19 add a CNN to produce fixed-length representations for the pitch and intensity features at the token level. Several CNN filters of different sizes perform one-dimensional convolution of the pitch and intensity features for each token, allowing the CNN to integrate information on various time scales (see Table 1 for exact filter sizes). The output of each filter is then max-pooled to create a fixed-dimension representation of pitch and intensity features for each token. All of the prosodic features (pitch, intensity, word and pause duration) are then concatenated with the text embedding for the corresponding token, and then input to the encoder. The CNN is trained along with the encoder-decoder model.

For the pipeline, we first segment into SUs, and then parse the resulting SUs. For segmentation, we use a modified version of the parser: With the same encoder, we change the decoder to only do sequence labeling, marking tokens as either SU-internal or SU-final. We then parse the predicted SUs. Rather than using a parser that was trained on gold SUs, we train a parser on the SUs that the segmenter predicted on the train set. This allows the model to learn to produce the parses on imperfectly segmented SUs and leads to better parsing scores. To reduce error propagation in the pipeline model, when we construct these parses for training, we insert a node labeled BLANK dominating any incorrectly predicted SUs. We remove any BLANK nodes the parser predicts in post-processing, essentially allowing the parser to fail to predict an SU where the segmentation step predicts one. However, the error propagation issues remain largely unchanged by this procedure.

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

4. Experimental set-up

4.1. Data

We use the American English corpus Switchboard NXT (henceforth SWBD-NXT) [4] to allow us to compare performance with T18 and T19 [1, 2]. This is a relatively small corpus compared to many datasets used today, but it remains the largest speech corpus with hand-annotated constituency parses. While other corpora with hand-annotated *dependency* parses exist (e.g., UD corpora such as [24] and [25]), these are all significantly smaller than Switchboard NXT. SWBD-NXT comprises 642 telephone dialogues between strangers, totalling 55k dialog turns. For training, development, and testing, we use the split described in [26], which is a standard split for experiments on SWBD-NXT (e.g., [12, 1]). The training set makes up almost 90 percent of the data (49k turns), and the development and testing sets make up slightly more than 5 percent each (3k turns). These dialogues are transcribed and hand-annotated with Penn Treebank-style constituency parses, and have no punctuation.

Not all turns in the SWBD-NXT contain multiple SUs: of a total 60.1k turns, 35.8k consist of a single SU. The average number of SUs per turn is 1.82. To avoid memory problems from too-long inputs, we filter out two problematically long turns from the training set (out of 50k 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 of the audio is missing.

4.2. Acoustic features

We extract acoustic features for pitch, intensity, pause duration, and word duration from the audio, largely following T18's feature extraction procedure, noting any devations below. The alignment of tokens to audio is annotated in the Switchboard NXT corpus. We extract pitch and intensity features from the speech signal with Kaldi [27], using 25ms frames every 10ms. We extract three pitch features: 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) [28]. For intensity features, we start with 40-dimensional mel-frequency filterbanks and calculate the log of the total energy, normalized by the maximum total energy for the speaker over the course of the dialog. We also calculate this value for the upper half and lower halves of the 40 mel-frequency bands.

Pause and word duration features are based on token timestamp annotations. Each word's pause feature corresponds to the pause that follows it, which we categorize 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 T18, the model learns 32-dimensional embeddings for each pause category. Unlike T18, we opt 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. This introduces negative-valued pauses, where one speaker interrupts the other, and so we choose to place these negative-valued pauses in the same bin as pauses with length 0.

We normalize word duration features, since we are interested in the relative lengthening or shortening of word tokens, rather than their raw length. Following the code base for T18, we perform two different types of normalization: normalizing

each token's raw duration by the mean duration of every occurrence of that word type; and normalizing the token's raw duration by the maximum duration of any word in the input unit.

We also considered using features for voice quality (e.g., harmonics-to-noise ratio, zero-crossing rate) since creaky voice has been observed to mark sentence boundaries in some varieties of American English [29]. However, we found no benefit from using these voicing features, so we omit them in the model we report on here.

4.3. Training

Unless stated otherwise, we train each model on five random seeds, and report the mean of each metric. To determine the statistical significance of differences in performance, we use bootstrap resampling [30], resampling 10^5 times.

We use the hyperparameters specified in T19's code base, documented in Table 1. Each model is trained for 50 epochs on a single Nvidia GTX 1080 GPU, which takes 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.

4.4. Evaluation

We report two metrics for both the pipeline and end-to-end models: parse and SU segmentation F1 scores. Parse F1 is calculated on the whole turn using a Python implementation of EVALB (link to be included in non-anonymous version). We don't count TURN constituents, so that turn-based and SU-based parse scores are comparable. The SU segmentation F1 score is calculated on all turn-medial SU boundaries; turn-final SU boundaries are not counted. In order to calculate the SU segmentation F1 score for the end-to-end model, we consider every node that is a direct child of the tree's top TURN node to be an SU. That is, SUs are just one kind of syntactic constituent, differentiated only by their location in the tree.

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
Num. CNN filters	32
CNN filter widths	5, 10, 25, 50

Table 1: *Model hyperparameters, based on T19.*

5. Results and discussion

Experiments with the end-to-end model show that prosody shows a statistically significant effect on parsing performance, though this effect is small (see Table 2). In the pipeline model, the effect of prosody is larger: The difference in parse F1 score from adding prosody is 0.94 for the pipeline model, where it is only 0.52 for the end-to-end model. However, the pipeline model's parse F1 score is lower than the end-to-end model's.

We make one modification to the end-to-end model as well in order to improve its segmentation performance. In the end-to-end model, poor segmentation performance is often caused when the model predicts many more children for the top node than it ought. Since we consider each direct child of the top TURN node to be an SU predicted by the end-to-end model, these highly branching nodes lower segmentation scores. The parser actually creates these multiply branching nodes by predicting a series of binary branching DUMMY nodes, which are collapsed in post-processing into a single n-ary node. In order to discourage oversegmentation, we experiment with weighting DUMMY nodes with a greater loss penalty. We found that adding a weight of 0.5 for each DUMMY node led to improved parse and SU segmentation quality. While weights higher than 0.5 continue to improve segmentation quality, particularly segmentation precision, as expected, they begin to harm parse quality.

However, even with this modification (which is included in all reported results), the end-to-end model underperforms the pipeline model on segmentation, which is surprising, given that it parses better than the pipeline model. The end-to-end model's higher parse performance is simple enough to explain in isolation: In the pipeline model, errors propagate from the segmentation step to the parsing step; this is impossible in the end-to-end model. But the end-to-end model's much lower segmentation score complicates the error propagation account.

We can explain this discrepancy by examining the kinds of errors each model makes. First, the end-to-end model tends not to predict top nodes for each gold SU, instead connecting lower nodes in the tree directly to the TURN node, leading to oversegmentation, as shown in Figure 1a. The pipeline model tends instead to undersegment. This tendency is shown in the end-to-end and the pipeline models' similar segmentation recall, compared to the end-to-end model's very low precision, shown in Table 3. The examples in Figure 1 show the effect this has on segmentation: The end-to-end example has a segmentation F1 score 57.1%, where the pipeline has a score of 66.7%.

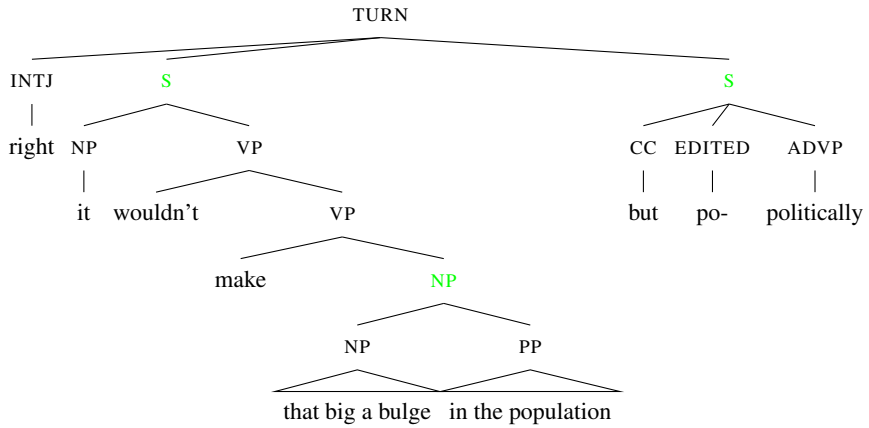
However, when scoring parses, the end-to-end model is penalized much less. Both models omit three nodes from the tree in Figure 1. However, the pipeline model also predicts sev-

	Gold SUs	E2E	Pipeline
Dev. set:			
Text only	90.31	85.70	84.34
Text+prosody	90.90	86.21	85.28
Test set:			
Text only	90.29	86.03	84.68
Text+prosody	90.65	86.55	85.62

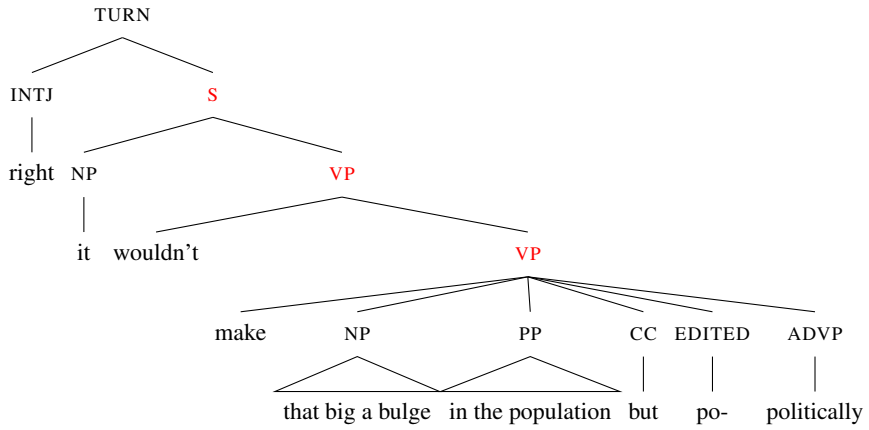
Table 2: *Development and test set parsing F1 score of the end-to-end and pipeline models (and for comparison, a model that receives gold standard SUs as input). Results averaged over 5 random seeds.*

		E2E	Pipeline
Dev. set:			
Text only	F1	66.32	63.74
Text+prosody	F1	72.95	77.38
	Prec	69.46	79.44
	Rec	76.92	75.69
Test set:			
Text only	F1	71.01	66.98
Text+prosody	F1	72.94	77.38

Table 3: *Segmentation F1 score of the turn-based models compared to the SU-based model, with precision and recall given for some cases, averaged over 5 random seeds.*



(a) *Gold parse. The nodes shown in green are omitted by both the end-to-end and pipeline models.*



(b) *Parse predicted by the pipeline model. The nodes shown in red are incorrect.*

Figure 1: *Comparison of gold parse to the pipeline's predicted tree. The end-to-end model's tree isn't shown because the only differences between it and the gold parse is the omission of the three nodes that are highlighted in green in the gold parse. This example has been edited for length and clarity, and part-of-speech tags have been omitted.*

eral nodes high up in the tree that the end-to-end model does not. This leads to a much lower parse F1 score for the pipeline model: 69.2%, compared to the end-to-end model's 88.0%.

In fact, the pipeline's better segmentation seems to actively *worsen* its parse score. The pipeline's undersegmentation comes from predicting nodes that dominate entire predicted SUs — like the S node in Figure 1b. Because the S node erroneously spans the entire turn, it is more likely that its VP daughter will also span too many nodes, as it does in this example.

The phenomena in this example affect performance on the whole development set, as shown by the interaction of parse precision and segmentation accuracy. On examples where the SU boundaries are incorrectly predicted, as in Figure 1, the pipeline model predicts many more incorrect nodes, and its parse precision declines by 5.22 percentage points on the development set (from 85.72% to 80.50%). By comparison, the end-to-end model's parse precision is less affected by segmentation quality – it only drops by 3.96 percentage points (from 86.20% to 82.24%).

These results suggest that the end-to-end model's superior parsing ability comes from the fact that it is able to model all syntactic units, including SUs, in the same way. The pipeline model has to treat SUs as being logically prior to and distinct from all sub-SU units, which leads to the error propagation described above. By modeling SUs and other syntactic constituents similarly, the end-to-end model is able to propose the sub-SU nodes that lead to the best parses overall, without being bound to a certain SU segmentation.

6. Conclusion

Previous work has shown that prosody improves parse quality. In this work, we show that prosody improves parse and SU segmentation quality simultaneously. We show that parse and SU segmentation score are not necessarily correlated: A pipeline model does SU segmentation better, but an end-to-end model produces better parses. We propose that this is because the end-to-end parser models SU boundaries the same way as other syntactic boundaries. By treating SUs as just another kind of syntactic unit, our model is able to take advantage of prosody to produce better parses overall.

7. Acknowledgements

This work was supported in part by ERC Advanced Fellowship GA 742137 SEMANTAX and the University of Edinburgh Huawei Laboratory.

8. References

[1] T. Tran, S. Toshniwal, M. Bansal, K. Gimpel, K. Livescu, and M. Ostendorf, "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)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 69–81. [Online]. Available: https://www.aclweb.org/anthology/N18-1007

[2] T. Tran, J. Yuan, Y. Liu, and M. Ostendorf, "On the Role of Style in Parsing Speech with Neural Models," in *Proc. Interspeech 2019*, 2019, pp. 4190–4194. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-3122

[3] J. G. Kahn, M. Ostendorf, and C. Chelba, "Parsing conversational speech using enhanced segmentation," in *Proceedings of HLT-NAACL 2004*. Boston, Massachusetts, USA: Association for Computational Linguistics, May 2 - May 7 2004, pp. 125–128. [Online]. Available: https://www.aclweb.org/anthology/N04-4032

[4] S. Calhoun, J. Carletta, J. Brenier, N. Mayo, D. Jurafsky, M. Steedman, and D. Beaver, "The NXT-format Switchboard Corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue," *Language Resources and Evaluation*, vol. 44, pp. 387–419, 12 2010.

[5] J. B. Pierrehumbert, "The phonology and phonetics of english intonation," Ph.D. dissertation, Massachusetts Institute of Technology, 1980.

[6] M. Steedman, "Information structure and the syntax-phonology interface," *Linguistic inquiry*, vol. 31, no. 4, pp. 649–689, 2000.

[7] E. Shriberg, "To 'errrr' is human: Ecology and acoustics of speech disfluencies," *Journal of the International Phonetic Association*, vol. 31, pp. 153 – 169, 06 2001.

[8] A. Cutler, D. Dahan, and W. van Donselaar, "Prosody in the comprehension of spoken language: A literature review," *Language and Speech*, vol. 40, no. 2, pp. 141–201, 1997.

[9] M. M. Kjelgaard and S. R. Speer, "Prosodic facilitation and interference in the resolution of temporary syntactic closure ambiguity," *Journal of Memory and Language*, vol. 40, no. 2, pp. 153 – 194, 1999.

[10] E. Noeth, A. Batliner, A. Kießling, R. Kompe, and H. Niemann, "Verbmobil: The use of prosody in the linguistic components of a speech understanding system," *IEEE Transactions on Speech and Audio processing*, vol. 8, no. 5, pp. 519–532, 2000.

[11] M. Gregory, M. Johnson, and E. Charniak, "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*, 04 2004.

[12] J. G. Kahn, M. Lease, E. Charniak, M. Johnson, and M. Ostendorf, "Effective use of prosody in parsing conversational speech," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. USA: Association for Computational Linguistics, 2005, p. 233–240. [Online]. Available: https://doi.org/10.3115/1220575.1220605

[13] E. Selkirk, "Sentence prosody: Intonation, stress, and phrasing," *The handbook of phonological theory*, vol. 1, pp. 550–569, 1995.

[14] ——, *Phonology and Syntax*. Cambridge, MA: MIT Press, 1984.

[15] M. Wagner and D. G. Watson, "Experimental and theoretical advances in prosody: A review," *Language and Cognitive Processes*, vol. 25, no. 7-9, pp. 905–945, 2010.

[16] Y. Gotoh and S. Renals, "Sentence boundary detection in broadcast speech transcripts," in *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[17] J. Kolář, E. Shriberg, and Y. Liu, "Using prosody for automatic sentence segmentation of multi-party meetings," in *International Conference on Text, Speech and Dialogue*. Springer, 2006, pp. 629–636.

[18] J. G. Kahn and M. Ostendorf, "Joint reranking of parsing and word recognition with automatic segmentation," *Computer Speech and Language*, vol. 26, no. 1, pp. 1 – 19, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0885230811000209

[19] C. Xu, L. Xie, G. Huang, X. Xiao, E. S. Chng, and H. Li, "A deep neural network approach for sentence boundary detection in broadcast news," in *INTERSPEECH-2014*, 2014, pp. 2887–2891.

[20] N. Kitaev and D. Klein, "Constituency parsing with a self-attentive encoder," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 2676–2686. [Online]. Available: https://www.aclweb.org/anthology/P18-1249

[21] M. Stern, J. Andreas, and D. Klein, "A minimal span-based neural constituency parser," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 818–827. [Online]. Available: https://www.aclweb.org/anthology/P17-1076

[22] D. Gaddy, M. Stern, and D. Klein, "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)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 999–1010. [Online]. Available: https://www.aclweb.org/anthology/N18-1091

[23] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[24] T. Wong, K. Gerdes, H. Leung, and J. Lee, "Quantitative comparative syntax on the Cantonese-Mandarin parallel dependency treebank," in *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*. Pisa,Italy: Linköping University Electronic Press, Sep. 2017, pp. 266–275. [Online]. Available: https://aclanthology.org/W17-6530

[25] K. Dobrovoljc and J. Nivre, "The Universal Dependencies treebank of spoken Slovenian," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 1566–1573. [Online]. Available: https://aclanthology.org/L16-1248

[26] E. Charniak and M. Johnson, "Edit detection and parsing for transcribed speech," in *Second Meeting of the North American Chapter of the Association for Computational Linguistics*, 2001. [Online]. Available: https://www.aclweb.org/anthology/N01-1016

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

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

[29] L. Wolk, N. B. Abdelli-Beruh, and D. Slavin, "Habitual use of vocal fry in young adult female speakers," *Journal of Voice*, vol. 26, pp. E111–E116, 2011.

[30] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.