An Empirical Comparison of Unsupervised Constituency Parsing Methods*

Jun Li[◇], **Yifan Cao**[◇], **Jiong Cai**[◇], **Yong Jiang**[†], and **Kewei Tu**[◇]
[◇]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
[◇]Shanghai Engineering Research Center of Intelligent Vision and Imaging, Shanghai, China
[†]DAMO Academy, Alibaba Group
{lijun2, caoyf, caijiong, tukw}@shanghaitech.edu.cn
yongjiang.jy@alibaba-inc.com

Abstract

Unsupervised constituency parsing aims to learn a constituency parser from a training corpus without parse tree annotations. While many methods have been proposed to tackle the problem, including statistical and neural methods, their experimental results are often not directly comparable due to discrepancies in datasets, data preprocessing, lexicalization, and evaluation metrics. In this paper, we first examine experimental settings used in previous work and propose to standardize the settings for better comparability between methods. We then empirically compare several existing methods, including decade-old and newly proposed ones, under the standardized settings on English and Japanese, two languages with different branching tendencies. We find that recent models do not show a clear advantage over decade-old models in our experiments. We hope our work can provide new insights into existing methods and facilitate future empirical evaluation of unsupervised constituency parsing.

1 Introduction

Unsupervised constituency parsing, a task in the area of grammar induction, aims to learn a constituency parser from a training corpus without parse tree annotations. While research on unsupervised constituency parsing has a long history (Carroll and Charniak, 1992; Pereira and Schabes, 1992; Stolcke and Omohundro, 1994), recently there is a resurgence of interest in this task and several approaches based on neural networks have been proposed that achieve impressive performance (Shen et al., 2018; Drozdov et al., 2019; Shen et al., 2019; Kim et al., 2019b,a; Jin et al., 2019).

With the recent development in research of unsupervised constituency parsing, however, the problem of lacking a unified experimental setting begins to emerge, which makes empirical comparison between different approaches difficult. First of all, although almost all previous approaches are evaluated on the Penn Treebank (Marcus and Marcinkiewicz, 1993), they differ in how they preprocess the training data, with respect to the sentence length limit, punctuation removal, vocabulary pruning, and so on. For example, non-neural methods such as Constituent Context Model (CCM) (Klein and Manning, 2002) are trained on short sentences, while modern neural based methods such as Parsing-Reading-Predict Network (PRPN) (Shen et al., 2018; Htut et al., 2018) do not impose any limit on sentence length.

Furthermore, existing approaches also differ in their evaluation metrics, with respect to the methods of computing averages, counting trivial spans, and so on. The evaluation results of the same approach using different metrics can differ significantly in some cases. Unfortunately, we have seen more than one paper that directly compares approaches evaluated with different metrics.

In this paper, we propose three standardized experimental settings with respect to data preprocessing, post-processing, evaluation metrics, and tuning. We then empirically compare five existing methods under the standardized settings, including two decade-old methods and three recently proposed neural methods. We run our experiments on English and Japanese, two languages with different branching tendencies. Interestingly, the overall experimental results show that the recent methods do not show a clear advantage over the decade-old methods.

We hope our empirical comparison could provide new insights into the relative strength and weakness of existing methods and our standard-

* This work was supported by the National Natural Science Foundation of China (61976139). Kewei Tu is the corresponding author.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3278–3283

ized experimental settings could facilitate future evaluation of unsupervised constituency parsing. Our pre/post-processing and evaluation source code can be found at https://github.com/i-lijun/UnsupConstParseEval.

2 Experimental Setup

2.1 Models

We choose to evaluate five models under our experimental setup: PRPN[1] (Shen et al., 2018), URNNG[2] (Kim et al., 2019b), CCM[3] (Klein and Manning, 2002), CCL[4] (Seginer, 2007), DIORA[5] (Drozdov et al., 2019). We use the open source implementation of each model, which we make sure can reproduce the results in the original papers.

PRPN is a neural-based model designed for language modeling by leveraging latent syntactic structures. It calculates syntactic distances between words of a sentence which can be used to obtain an unlabeled parse tree. Note that as a constituency parser, PRPN is incomplete (Dyer et al., 2019).

URNNG is an unsupervised version of the supervised neural parser RNNG (Dyer et al., 2016). It uses a chart parser to approximate the posterior of the original RNNG.

DIORA is a recursive autoencoder using the inside-outside algorithm to compute scores and representations of spans in the input sentence. It is the only model in our comparison that uses external word embedding (in our experiments, we use ELMo (Peters et al., 2018) for English and fastText (Grave et al., 2018) for Japanese).

CCM is a generative distributive model, the parameters of which are updated with the EM algorithm. It is the only model in our comparison that uses the gold Part-of-Speech tags as input.

CCL is an incremental parser, which uses a representation for syntactic structures similar to dependency links.

In addition to these models, we note that there are several other models that achieve good results on unsupervised constituency parsing, such as UML-DOP (Bod, 2006), UPParse (Ponvert et al., 2011), feature CCM (Golland et al., 2012), Depth-Bounded PCFG (Jin et al., 2018), and Compound PCFG (Kim et al., 2019a). However, because of limited time and computational resource, as well as a lack of open source implementations for some of the models, we do not evaluate them in our experiments.

2.2 Datasets and Preprocessing

We use two corpora in our evaluation: the English Penn Treebank (PTB) (Marcus and Marcinkiewicz, 1993) and the Japanese Keyaki Treebank (KTB) (Butler et al., 2012). We pick KTB in addition to PTB for the purpose of checking the generalizability of existing models on left-branching languages. For PTB, we follow the standard split, using section 02-21 for training, 22 for validation and 23 for testing. For KTB, we shuffle the corpus and use 80% of the sentences for training, 10% for validation and 10% for testing.

Many previous approaches learn from training sentences of length ≤ 10, but recent models based on language modeling often use a length limit of 40 or set no length limit at all. We experiment with both length ≤ 10 and length ≤ 40. We do not impose any length limit on test sentences.

Previous models also have different ways to deal with punctuation. Although Jones (1994) and Spitkovsky et al. (2011) point out that careful treatment of punctuation may be helpful in unsupervised parsing, many previous models choose to remove punctuation and some recent models treat punctuation as normal words. Only a few models such as CCL (Seginer, 2007) make special treatment of punctuation. We experiment with two settings for length 40, one with punctuation and one without.

To reduce the vocabulary size, we replace all the numerals with a <num>token and words that appear only once with <unk>.

2.3 Post-processing

The parses output by CCL do not contain punctuation even when it is trained with punctuation, so it cannot be evaluated properly using a test set with punctuation. In addition, although the right branching baseline is a very strong baseline when punctuation is removed, its evaluation score becomes very low if punctuation is included because of its treatment of trailing punctuation. So we extend the post-processing method used in (Drozdov et al., 2019) to either add back punctuation marks or modify their connections in a parse tree: for a trailing punctuation mark, we manually attach it to

[1] https://github.com/yikangshen/PRPN
[2] https://github.com/harvardnlp/urnng
[3] https://github.com/davidswelt/dmvccm
[4] https://github.com/DrDub/cclparser
[5] https://github.com/iesl/diora

Train	ptb_len10_nopunct			ptb_len40_nopunct			ptb_len40_punct		
Metric	micro	macro	evalb	micro	macro	evalb	micro	macro	evalb
Evaluated on test sentences with length ≤ 10.									
PRPN	31.29 ± 4.49	37.29 ± 5.04	44.72 ± 3.59	**56.98** ± 3.66	**58.79** ± 2.85	**65.23** ± 2.92	38.07 (52.17) ± 3.94 (± 3.08)	33.75 (46.1) ± 3.33 (± 2.75)	51.56 (60.59) ± 3.08 (± 1.94)
URNNG	50.77 ± 1.11	53.67 ± 0.83	60.41 ± 0.89	51.43 ± 0.00	54.20 ± 0.00	60.94 ± 0.00	**47.95** (49.07) ± 0.00 (± 0.00)	**41.65** (44.61) ± 0.00 (± 0.00)	**59.34** (59.78) ± 0.00 (± 0.00)
DIORA	31.55 ± 2.50	37.90 ± 2.13	44.93 ± 2.00	50.26 ± 0.72	52.92 ± 0.68	59.86 ± 0.58	42.66 (47.13) ± 0.98 (± 1.92)	37.77 (41.37) ± 0.84 (± 1.30)	55.15 (57.87) ± 0.77 (± 1.36)
CCL	28.31	36.61	33.55	53.67	57.45	53.67	n/a **(62.39)**	n/a (52.33)	n/a (62.00)
CCM	**62.97**	**63.35**	**70.14**	50.29	53.73	60.03	1.04 (54.30)	4.30 (54.68)	22.70 (58.02)
LBranch	13.32	22.39	30.37	13.32	22.39	30.37	11.73 (13.79)	14.08 (24.31)	30.98 (35.66)
RBranch	51.43	54.20	60.79	51.43	54.20	60.79	1.03 (56.80)	4.30 **(56.19)**	22.63 **(67.74)**
UBound	83.20	78.74	86.64	83.20	78.74	86.64	68.19	56.85	75.15
Evaluated on all test sentences.									
PRPN	18.08 ± 3.66	21.73 ± 3.69	22.85 ± 3.45	41.99 ± 4.05	45.50 ± 3.73	45.36 ± 3.82	33.25 (42.17) ± 3.20 (± 1.82)	33.92 (43.55) ± 3.27 (± 1.95)	36.85 (44.43) ± 3.03 (± 1.60)
URNNG	34.62 ± 2.19	38.58 ± 1.65	38.43 ± 2.07	35.88 ± 0.00	39.58 ± 0.00	39.62 ± 0.00	36.7 (36.72) ± 0.00 (± 0.00)	38.44 (38.84) ± 0.00 (± 0.00)	40.11 (40.03) ± 0.00 (± 0.00)
DIORA	20.44 ± 1.53	23.72 ± 1.66	25.08 ± 1.44	**46.27** ± 0.31	**47.81** ± 0.33	**49.39** ± 0.29	**41.48** (46.94) ± 0.43 (± 1.59)	**41.56** (46.73) ± 0.37 (± 1.50)	**44.63** (**49.38**) ± 0.41 (± 1.44)
CCL	19.08	21.56	18.68	37.41	41.67	37.98	n/a **(49.70)**	n/a **(51.51)**	n/a (47.46)
CCM	**49.54**	**52.60**	**52.48**	40.90	43.62	44.34	0.09 (33.15)	0.54 (36.88)	5.48 (35.65)
LBranch	6.00	8.98	11.49	6.00	8.98	11.49	4.88 (5.55)	6.36 (8.30)	10.01 (11.07)
RBranch	35.88	39.58	39.61	35.88	39.58	39.61	0.07 (35.54)	0.52 (38.98)	5.45 (39.3)
UBound	84.41	83.32	85.34	84.41	83.32	85.34	77.76	75.06	78.96

Table 1: Experimental results on PTB. The column headings show the training setups and the evaluation metrics. The presence or removal of punctuation in a test set is kept consistent with the corresponding training setup. Scores in parentheses are obtained using the post-processing method of section 2.3. For models sensitive to random seeds (PRPN, URNNG and DIORA), we report the means and standard deviations from five runs. LBranch and RBranch represent the left and right branching baselines. UBound represents the score upper bound that a binary tree parser can achieve.

the root of the constituency parse tree; for a punctuation mark inside the sentence, we attach it to the lowest common ancestor of its two adjacent words in the parse tree. Note that the above procedure will produce non-binary parse trees.

2.4 Evaluation Metrics

The performance of a constituency parser is often evaluated with F1 scores. However, two ways of averaging F1 scores over multiple test sentences are available, i.e., micro average and macro average. In micro average, all the span predictions are aggregated together and then compared with the gold spans to get the precision and recall. In contrast, macro average is obtained by calculating the F1 score for each individual sentence and then take an average over all the sentences.

We use both metrics in our experiments. Note that when computing F1 scores, we remove trivial spans, i.e., single-word spans and whole-sentence spans, and we calculate duplicate constituents only once.

We additionally use the standard PARSEVAL metric computed by the Evalb program[6]. Although Evalb calculates the micro average F1 score, it differs from our micro average metric in that it will count the whole sentence spans and duplicated spans are calculated and not removed.

2.5 Tuning and Model Selection

To maintain the unsupervised nature of our experiments, we avoid the common practice of using gold parses of the validation set for hyperparameter tuning. CCM and CCL do not expose any hyperparameter for tuning. We tune PRPN and URNNG based on their perplexity on the validation set. DIORA does not provide a metric that can be used for tuning, so we do not tune it.

We tune PRPN and URNNG with the same time budget of 5 days on a GPU cluster with TITAN V GPUs. We use Bayesian optimization[7] to automatically tune these models. We set the ranges of hyperparameter values around the default values provided in the original papers.

[6] https://nlp.cs.nyu.edu/evalb/

[7] https://github.com/fmfn/BayesianOptimization

Train	ktb_len10_nopunct			ktb_len40_nopunct			ktb_len40_punct		
Metric	micro	macro	evalb	micro	macro	evalb	micro	macro	evalb
Evaluated on test sentences with length ≤ 10.									
PRPN	10.18 ± 2.75	23.72 ± 2.17	30.48 ± 2.12	14.29 ± 11.95	26.80 ± 9.56	33.67 ± 9.25	8.09 (9.27) ± 1.12 (± 1.32)	20.47 (23.95) ± 0.98 (± 1.15)	29.61 (29.95) ± 0.86 (± 0.89)
URNNG	1.37 ± 0.00	16.60 ± 0.00	23.43 ± 0.00	1.93 ± 0.00	17.13 ± 0.00	23.86 ± 0.00	2.71 (1.89) ± 0.00 (± 0.00)	16.25 (17.91) ± 0.00 (± 0.00)	25.39 (24.74) ± 0.00 (± 0.00)
DIORA	21.96 ± 6.59	32.37 ± 5.35	39.60 ± 5.10	**34.69** ± 6.51	**42.20** ± 5.00	**49.45** ± 5.04	**27.00** (**27.34**) ± 3.82 (± 4.51)	**34.68** (35.86) ± 2.95 (± 3.69)	**44.10** (43.24) ± 2.92 (± 3.15)
CCL	18.49	30.31	32.28	2.74	18.43	27.47	n/a (13.93)	n/a (27.85)	n/a (36.90)
CCM	**24.69**	**36.32**	**41.72**	32.67	41.97	47.89	3.44 (3.45)	16.47 (18.93)	26.05 (25.82)
LBranch	23.86	34.69	41.07	23.86	34.69	41.07	20.10 (25.46)	29.98 (**36.37**)	38.81 (**45.61**)
RBranch	1.37	16.60	23.67	1.37	16.60	23.67	2.12 (1.29)	15.68 (17.52)	25.05 (27.97)
UBound	57.68	60.82	67.25	57.68	60.82	67.25	49.62	52.86	61.41
Evaluated on all test sentences.									
PRPN	8.01 ± 1.19	13.92 ± 1.28	15.61 ± 1.09	11.11 ± 8.06	17.25 ± 8.82	18.45 ± 7.39	5.83 (7.15) ± 0.71 (± 0.77)	10.16 (12.17) ± 0.78 (± 0.88)	13.1 (14.07) ± 0.65 (± 0.67)
URNNG	0.24 ± 0.00	6.44 ± 0.00	8.47 ± 0.00	0.68 ± 0.00	6.94 ± 0.00	8.87 ± 0.00	0.33 (0.26) ± 0.00 (± 0.00)	5.08 (5.6) ± 0.00 (± 0.00)	8.01 (7.95) ± 0.00 (± 0.00)
DIORA	**14.95** ± 3.22	**21.90** ± 4.19	**21.97** ± 2.95	**29.94** ± 3.16	**35.06** ± 4.04	**35.72** ± 2.90	**24.22** (**23.48**) ± 4.32 (± 4.45)	**28.09** (**28.08**) ± 3.88 (± 4.18)	**30.06** (**28.98**) ± 3.98 (± 4.03)
CCL	12.62	19.43	18.03	1.20	7.69	12.60	n/a (8.63)	n/a (14.18)	n/a (18.44)
CCM	12.21	21.70	19.46	20.21	28.60	26.80	1.33 (1.42)	5.91 (6.78)	8.94 (8.98)
LBranch	11.15	20.62	18.49	11.15	20.62	18.49	9.63 (10.77)	16.77 (19.66)	16.60 (18.26)
RBranch	0.22	6.43	8.46	0.22	6.43	8.46	0.20 (0.17)	4.83 (5.45)	7.89 (8.54)
UBound	64.38	62.52	67.32	64.38	62.52	67.32	59.40	56.44	62.53

Table 2: Experimental results on KTB.

3 Experimental Results

We list the experimental results of all the models and the left/right-branching baselines for PTB and KTB in Table 1 and Table 2 respectively. Since all the models except CCL produce binary parse trees, we also show the score upper bound that a binary tree parser can achieve, which is computed by binarizing the gold trees and calculating their scores against the original gold trees.

Note that our results can be very different from those reported in the original papers of these models because of different experimental setups. For example, the original CCM paper reports an F1 score of 71.9 on PTB, but we report 62.97. This is because the original CCM experiment uses the whole WSJ corpus (with length ≤ 10) for both training and test, which is very different from our setup.

Also note that for the left and right branching baselines and the binary upper bound, the scores for "length 10 no punct" and "length 40 no punct" are the same, because these baselines do not require training and are evaluated on the same test sets.

Overall Comparison There is no universal winner for all the settings but there is clear winners for specific settings. On PTB, it is surprising to see that each model is the winner of at least one setting. Right-branching is a very strong baseline and with post-processing it outperforms all the models in some settings of "ptb_len40_punct". On KTB, DIORA is the winner in most of the settings, while CCM has a strong performance on "ktb_len10_nopunct". Left-branching is a strong baseline especially when evaluated on sentences with length ≤ 10.

Although CCM and DIORA achieve the best overall performance, we note that they both utilize additional resources. CCM uses gold POS tags and DIORA uses pretrained word embedding. Our preliminary experiments on PTB show a significant drop in performance when we run CCM using words without gold POS tags, with the Evalb F1 score dropping from 70.14 to 57.29 when evaluated on length ≤ 10 under the "ptb_len10_nopunct" setting. DIORA also performs worse when pretrained word embedding is replaced by randomly initialized embedding, with the average Evalb F1 score dropping from 49.39 to 42.63 when evaluated on all sentences under the "ptb_len40_nopunct" setting.

Overall, we do not see a clear advantage of more recent neural models over traditional models. There are two factors that should be taken into account though. First, neural models are significantly slower and therefore may not have been sufficiently tuned because of the fixed tuning time budget. Second, the training data may still be too

small from the perspective of neural models.

Finally, we also note that our post-processing method for adding back punctuation almost always improves the score in PTB, sometimes by a large margin (e.g., for CCM and RBranch). On KTB, however, it sometimes decreases the score. This may be caused by different annotation standards for punctuation in the two treebanks.

Impact of Experimental Settings Different experimental settings lead to remarkable difference in the evaluation scores of the same model. Different evaluation metrics also produce very different scores. With the same output parses, they can sometimes differ more than 20 F1 points.

Running Time Traditional models such as CCM and CCL are fast, taking only several minutes. On the other hand, neural models take hours or even days to train. Apart from training, the inference stage is also very fast for traditional models but slow for neural models. Considering their close F1 scores, we believe at least in the scenario of limited data and computational resources, traditional models are preferred to neural models.

Comments on Individual Models We find that CCM when trained with length ≤ 10 sentences is very competitive. On PTB, it even outperforms all the other models that are trained on length 40 data with no punctuation. However, CCM cannot handle punctuation very well without post-processing.

URNNG seems to degrade to mostly right-branching in many settings (thus having very low standard deviations). This is possibly due to two reasons: 1) URNNG takes a lot of time to train and is therefore only lightly tuned because of the tuning time budget; 2) in the original paper, URNNG is trained with punctuation but evaluated without punctuation, which is quite different from our settings.

PRPN has a strong performance on PTB when trained with long sentences. However, we note that PRPN has a right-branching bias during inference (Dyer et al., 2019). If we switch its inference bias to left-branching, the performance drops significantly (for more than 10 points). Because of its right-branching bias, PRPN does not perform well on KTB.

4 Discussion

We make the following recommendations for future experiments on unsupervised constituency parsing. For the sentence length limit, we think one can set any limit on the training data, but should report evaluation results on both length ≤ 10 and all-length test data. For the evaluation metrics, since small details in implementing micro and macro average will lead to nontrivial differences, we suggest using PARSEVAL which has publicly available implementation. For models sensitive to random seeds, we recommend reporting means and standard deviations from multiple runs. We also recommend evaluation on treebanks of both left-branching and right-branching languages, such as PTB and KTB.

References

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 865–872, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alastair Butler, Zhu Hong, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, and Zhen Zhou. 2012. Keyaki treebank: phrase structure with functional information for japanese. In *Proceedings of Text Annotation Workshop*.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University, Providence, RI, USA.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *North American Association for Computational Linguistics*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL*.

Chris Dyer, Gábor Melis, and Phil Blunsom. 2019. A critical analysis of biased parsers in unsupervised parsing. *arXiv preprint arXiv:1909.09428*.

Dave Golland, John DeNero, and Jakob Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 17–22, Stroudsburg, PA, USA. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4998–5003, Brussels, Belgium. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, Lane Schwartz, and William Schuler. 2019. Unsupervised learning of PCFGs with normalizing flow. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452, Florence, Italy. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy A. Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised grammar induction with depth-bounded pcfg. *Transactions of the Association for Computational Linguistics*, 6:211–224.

Bernard E. M. Jones. 1994. Exploring the role of punctuation in parsing natural text. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.

Yoon Kim, Alexander M. Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of NAACL*.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Mitchell P Marcus and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2).

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics*, ACL '92, pages 128–135, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1077–1086. Association for Computational Linguistics.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 19–28, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by bayesian model merging. In *International Colloquium on Grammatical Inference*, pages 106–118. Springer.