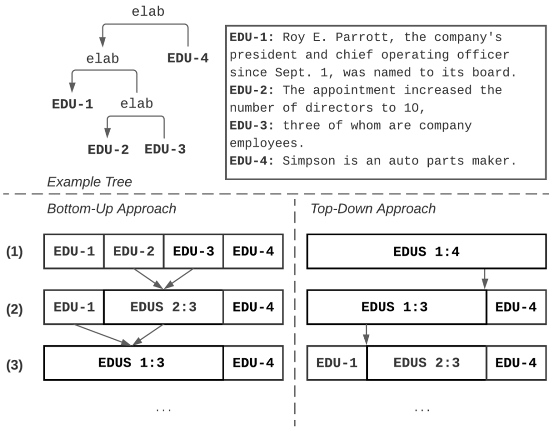
Easy-First Bottom-Up Discourse Parsing via Sequence Labelling

Andrew Shen♠♡ **Fajri Koto**♠ **Jey Han Lau**♠ **Timothy Baldwin**♠♢
♠ School of Computing and Information Systems, The University of Melbourne
♡ School of Computer Science, Carnegie Mellon University
♢ Department of Natural Language Processing, MBZUAI
ashen3@cs.cmu.edu, fajri.koto91@gmail.com
jeyhan.lau@gmail.com, tbaldwin@unimelb.edu.au

Abstract

We propose a novel unconstrained bottom-up approach for rhetorical discourse parsing based on sequence labelling of adjacent pairs of discourse units (DUs), based on the framework of Koto et al. (2021). We describe the unique training requirements of an unconstrained parser, and explore two different training procedures: (1) fixed left-to-right; and (2) random order in tree construction. Additionally, we introduce a novel dynamic oracle for unconstrained bottom-up parsing. Our proposed parser achieves competitive results for bottom-up rhetorical discourse parsing.



EDU-1: Roy E. Parrott, the company's president and chief operating officer since Sept. 1, was named to its board.
EDU-2: The appointment increased the number of directors to 10,
EDU-3: three of whom are company employees.
EDU-4: Simpson is an auto parts maker.

Figure 1: An example discourse tree (`elab` = elaboration, "←" means Nucleus–Satellite relation). For this tree, we show the parsing states of the bottom-up (left) and top-down (right) approaches.

1 Introduction

Discourse analysis aims to explain the relationship of texts beyond sentence boundaries, and has been modelled based on Rhetorical Structure Theory (RST: Mann and Thompson (1988)). In the RST framework, texts are modelled as a labelled hierarchy of discourse units (DU), with elementary discourse units (EDU) being the smallest unit (see Figure 1).

Although there has been a move from bottom-up (Hernault et al., 2010; Ji and Eisenstein, 2014; Joty et al., 2015; Li et al., 2016; Yu et al., 2018; Mabona et al., 2019) to top-down approaches (Lin et al., 2019; Zhang et al., 2020; Nguyen et al., 2021; Koto et al., 2021), we argue that the bottom-up paradigm is conceptually intuitive as humans analyse the structure of documents incrementally based on elementary structures. Furthermore, in contemporaneous work, Yu et al. (2022) have shown that bottom-up parsers built on a language model pre-trained at the EDU level outperform top-down parsers trained comparably.

In this paper, we revisit the bottom-up approach and introduce a novel *unconstrained* bottom-up discourse parsing $\mathcal{O}(n^2)$ by adopting the sequence-labelling framework of Koto et al. (2021). *Unconstrained* means that we relax the fixed left-to-right direction of discourse tree construction, allowing us to make the easiest decisions first. Intuitively speaking, when it comes to making the harder decisions, the history of existing structures can be used to make more reliable predictions.

Goldberg and Elhadad (2010) introduced the non-directional easy-first algorithm to dependency parsing, which is a greedy, best-first parser, which relaxes the left-to-right order constraint of other bottom-up transition-based algorithms (Yu et al., 2018). Because the model is conditioned on existing parsed structures, we need to sample parsing trajectories to train the model, and compare two simple sampling methods: (1) left-to-right, and (2) random. To the best of our knowledge, we are the first to propose a bottom-up model for discourse parsing using the easy-first algorithm in a sequence labelling framework.

To summarize our contributions: (1) we propose a novel bottom-up context-sensitive parser; (2) we explore sampling methods for training a context-sensitive parser; and (3) we devise a novel dynamic oracle for our unconstrained bottom-up discourse

Proceedings of 3rd Workshop on Computational Approaches to Discourse, pages 35–41
October 16 – 17, 2022. ©2022 International Conference on Computational Linguistics

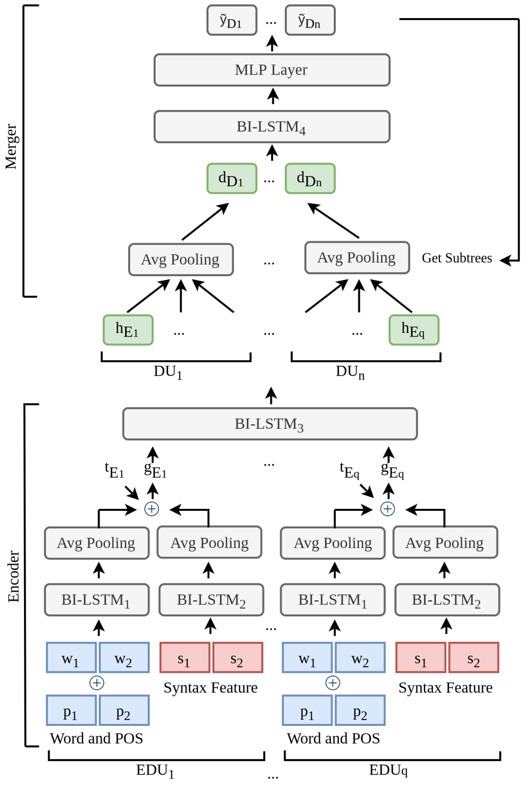


Figure 2: Architecture of the model

parser. We make the source code available online.[1]

2 Bottom-Up RST Parsing

We construct RST trees in a bottom-up fashion, starting with a sequence of EDUs and sequentially merging adjacent discourse units. At each stage, there are multiple merge points in the partially-parsed document that make up the gold discourse tree, and we define all such points to be gold merges. We impose no constraint on which gold merge needs to be executed first.

Following Koto et al. (2021), we frame the merging task as a sequence labeling problem. We train a merging model to assign a binary label $y \in \{0, 1\}$ to each discourse unit, where 1 indicates the unit and its right neighbour are subject to a gold merge. For each parse state, we train the model to label all gold merge points. At test time, we select the highest-probability merge point to construct the next parse state. We assign the discourse label and nuclearity relation separately with a second classifier after a merge is decided.

2.1 Model

Following Koto et al. (2021), our merging module consists of two blocks, as depicted in Figure 2. The first block is an EDU encoder. We use the hierarchical LSTM architecture of Yu et al. (2018), generating encodings with implicit syntax features. We obtain a suitable representation for each EDU text span $\{w_1, w_2, \ldots, w_m\}$ by using two Bi-LSTMs (Bi-LSTM$_1$ and Bi-LSTM$_2$). Bi-LSTM$_1$ is given the neural embedding of w_i concatenated with the part of speech embedding as input. Bi-LSTM$_2$ is given the syntax embedding s_i of each work as input. The syntax embedding comes from the syntax dependency parser from Dozat and Manning (2017). We also use an EDU type embedding t_{E_j} to distinguish EDUs at the end of a paragraph from other EDUs. The final EDU encoding g_{E_j} is the concatenation of the average output states for both Bi-LSTMs over the EDU and the EDU type embedding t_{E_j}:

$$\begin{aligned} x_i &= w_i \oplus p_i \\ \{a_1^w, .., a_p^w\} &= \text{Bi-LSTM}_1(\{x_1, .., x_p\}) \\ \{a_1^s, ..., a_p^s\} &= \text{Bi-LSTM}_2(\{s_1, .., s_p\}) \\ g_{E_j} &= \text{Avg-Pool}(\{a_1^w, .., a_p^w\}) \oplus \\ &\quad \text{Avg-Pool}(\{a_1^s, .., a_p^s\}) \oplus t_{E_j} \end{aligned}$$

Given a sequence of independent EDU encodings, we use a third Bi-LSTM (Bi-LSTM$_3$) to capture relationships between EDUs and produce a contextualized encoding h_{E_j}:

$$\{h_{E_1}, \ldots, h_{E_q}\} = \text{Bi-LSTM}_3(g_{E_1}, \ldots, g_{E_q})$$

The second block (the top half of Figure 2) is the merger, and deviates from Koto et al. (2021). The parse state consists of a sequence of discourse units, each of which is represented by averaging the encodings of the component EDUs:

$$d_{D_k} = \text{Avg}(h_{E_a}, \ldots, h_{E_b})$$

where D_k is a discourse unit with EDU span $E_{a:b}$.

We use a fourth Bi-LSTM (Bi-LSTM$_4$) to encode relationships between complex discourse units and assign a binary label to each merge.

$$\{d'_{D_1}, \ldots, d'_{D_n}\} = \text{Bi-LSTM}_4(d_{D_1}, \ldots, d_{D_n})$$

$$\hat{y}_{D_k} = \sigma(\text{MLP}(d'_{D_k}))$$

[1] https://github.com/Redrew/NeuralRST-Bottom-Up

Algorithm 1 Bottom-up Dynamic Oracle

```
function DynOracle($E, O, R$)
    # For training only
    # E is list of EDUs
    # O is gold order for merger
    # R is list of gold discourse labels based on O
    $q = \text{length}(E)$; $state = \{E_1, \ldots, E_q\}$
    while $\|state\| > 1$ do
        $id_{gold} = \text{oracleMerge}(state, O, R)$
        $id_{pred} = \text{predictMerge}(state)$
        $r_{pred1} = \text{predictLabel}(state, id_{gold})$
        $r_{pred2} = \text{predictLabel}(state, id_{pred})$
        if $\text{random}() > \alpha$ then
            $state = merge(state, id_{gold})$
            $r_{gold} = \text{oracleLabel}(state, id_{gold})$
            $L = Loss(id_{gold}, r_{gold}, id_{pred2}, r_{pred1})$
        else
            $state = merge(state, id_{pred})$
            $r_{oracle} = \text{oracleLabel}(state, id_{pred})$
            $L = Loss(id_{gold}, r_{oracle}, id_{pred1}, r_{pred1})$
        end if
    end while
end function
```

We predict the joint probability distribution of the nuclearity and discourse labels after a merge is chosen by feeding the encodings d'_{ind}, d'_{ind+1} of the selected discourse units into an MLP layer, where ind is the index of the left discourse unit chosen to be merged:

$$z_{nuc+dis} = \text{softmax}(\text{MLP}(d'_{ind}, d'_{ind+1}))$$

The final training loss of our model is the combination of the merging and nuclearity-discourse prediction loss: $\mathcal{L} = \mathcal{L}_{merge} + \mathcal{L}_{nuc+dis}$.

2.2 Merge Order in Training

Because the model evaluates each merge candidate in the context of all previously parsed structures in the document, different permutations of parse states with discourse units not part of the merge candidate can lead to different predictions for that merge candidate. We propose to sample parse sequences for training. We evaluate two different sampling schemes: (1) merging gold pairs left to right; and (2) merging gold pairs at random.

2.3 Dynamic Oracle

In the standard training regimen, the model is only trained on parse states constructed by a sequence of correct merges. However, at test time, the model will often see error parse states, created by an incorrect merge in its history. Because the model is never trained on error states, it will struggle to recover after it has made a mistake.

We address this problem by training our model with a dynamic oracle, first introduced by Goldberg and Nivre (2012) and adopted for discourse parsers (Yu et al., 2018; Koto et al., 2021). Given an error state, a dynamic oracle provides the next set of merge actions that will minimize deviation between the gold tree and the final tree. The dynamic oracle is described in Algorithm 1. At each merging step in training, with probability α we execute the predicted merge instead of the sampled gold merge. In this manner, we introduce error states to the training set and teach the model to predict the next set of oracle actions, so the parser chooses the best actions even after a mistake.

In a document with n EDUs, the oracle assigns a merge order to each $n-1$ cut separating adjacent EDUs. The merge order is defined as the earliest step discourse units to the left and right of the cut are merged in all possible gold merge sequences. If the merge order of a cut is lower than adjacent cuts, it is an oracle action to merge the two discourse units around the cut, because in such cases, other gold merges that involve the two discourse units must come after the oracle action.

3 Experiments

3.1 Data

Following previous studies (Koto et al., 2021; Yu et al., 2018), we focus on the English language and use the RST Discourse Treebank for our experiments, binarizing all discourse trees in a right-heavy manner. It contains 347 annotated documents for training and 38 documents for testing. Our development set consists of the same 35 documents as Koto et al. (2021) and Yu et al. (2018), taken from the training set. Consistent to previous works, we use the same 18 coarse-grained discourse relationships and use the gold EDU segments for discourse tree construction.

3.2 Set-Up

We use the standard Parseval metrics for RST parsing of Marcu (2000). Based on the recommendations of a recent replication study (Morey et al., 2017), we report micro-averaged F-1 scores on labeled attachment decisions (original Parseval) instead of macro-averaged F-1 scores (RST-Parseval). The Parseval metrics consist of: `Span`, `Nuclearity`, `Relation`, and `Full`.[2]

[2] `Span` evaluates the correctness of the predicted tree structure. `Nuclearity` evaluates the tree skeleton together with nuclearity indications. `Relation` evaluates the tree skeleton with the discourse relations. `Full` evaluates the tree skeleton along with nuclearity indications and discourse relations.

Merge Order	**Full**	**Bias**
Left Merge	47.3	12.6
Random Merge	51.8	0.8

Table 1: Sampling strategy results over the dev set, based on the `Full` metric (micro-averaged F-score on labeled attachment decisions) and Bias (depth difference between the left and right end of the tree).

Method	**S**	**N**	**R**	**F**
Bottom-Up:				
Feng and Hirst (2014)†	68.6	55.9	45.8	44.6
Ji and Eisenstein (2014)†	64.1	54.2	46.8	46.3
Surdeanu et al. (2015)†	65.3	54.2	45.1	44.2
Joty et al. (2015)	65.1	55.5	45.1	44.3
Hayashi et al. (2016)	65.1	54.6	44.7	44.1
Li et al. (2016)	64.5	54.0	38.1	36.6
Braud et al. (2017)	62.7	54.5	45.5	45.1
Yu et al. (2018) (static)‡	71.1	59.7	48.4	47.4
Yu et al. (2018) (dynamic)‡	71.4	60.3	49.2	48.1
Mabona et al. (2019)	67.1	57.4	45.5	45.0
Yu et al. (2022) (XLNet)	**76.4**	**66.1**	**54.5**	**53.5**
Top-Down:				
Zhang et al. (2020)	67.2	55.5	45.3	44.3
Nguyen et al. (2021)	67.1	57.4	45.5	45.0
Koto et al. (2021) (static)‡	72.7	61.7	50.5	49.4
Koto et al. (2021) (dynamic)‡	73.1	62.3	51.5	50.3
Our proposed Bottom-Up Method:				
Static‡	73.3	62.0	50.1	49.1
Dynamic‡	73.6	62.3	50.3	49.3

Table 2: Results over the test set calculated using micro-averaged F-1 on labeled attachment decisions (original Parseval). All metrics (S: `Span`, N: `Nuclearity`, R: `Relation`, F: `Full`) are averaged over three runs. "†" and "‡" denote that the model uses sentence and paragraph boundary features, respectively.

We adopt the hyperparameter settings used in Koto et al. (2021). `GloVe` embeddings (Pennington et al., 2014) are used to encode the words in each EDU. We use CoreNLP (Manning et al., 2014) to obtain POS tag, and initialize each POS encoding as a random vector. The embedding dimension of words, POS tags, EDU type and syntax features are 200, 200, 100 and 1200, respectively. The dimensionality of the Bi-LSTMs in the encoder is 256 and Bi-LSTM$_4$ in the merge classifier has a dimension of 128. We use batch size = 4, gradient accumulation = 2, learning rate = 0.001, dropout probability = 0.5, and optimizer = Adam (with epsilon of 1e-6). When training with a dynamic oracle, we activate the dynamic oracle after 50 epochs.

We tune the α value used in the dynamic oracle on the development set. We performed grid search on α values, each averaging the `Full` Parseval metric over three random seeds. For training with a dynamic oracle, we found that $\alpha = 0.8$ resulted in the best `Full` Parseval score.

We use a single Tesla V100 SXM2 32 GB with 4 CPU cores to run our experiments. A run with static oracle takes around 14 hours in run time.

3.3 Results

We present analysis of the sampling strategy in Table 1. All results are averaged over three runs with different random seeds on the development set, with a static oracle. We compare training with left-first state sequences and randomly-sampled state sequences, and find that the latter result in an absolute +4.5 improvement over training with left-first state sequences. As such, we use random sampling for the remainder of the paper.

We benchmark our parser against previous state-of-the-art RST parsers over the test set. The results are presented in Table 2 (original Parseval).

Training with a dynamic oracle improved results over a static oracle, with a Full score increase of +0.2. Even with a static oracle, our parser surpasses previous bottom-up parsers with a simple greedy algorithm, without the need for complex post-editing or a chart-parsing algorithm. The sequence labeling framework has the benefit of being conceptually simpler than transition parsers. Training with a dynamic oracle adds algorithmic complexity during training, but our inference procedure remains the same. Our parser is most comparable with the transition-based parser proposed by Yu et al. (2018), which shares the same LSTM-architecture as our work and also utilises implicit syntax features. Our results demonstrate that a parser with the context of the document structure outperforms parsers without structure context.

Compared to the top-down parser proposed by Koto et al. (2021) with the dynamic oracle, our results for Span and Nuclearity are superior or equivalent, but the relation classification results are slightly inferior, resulting in slightly lower results overall. It is important to note that, while noticeably superior to our approach, the methods of Yu et al. (2022) and Zhang et al. (2021) are heavily based on pre-trained LMs, where our method makes no use of pre-training, which we leave to future work.

3.4 Analysis

We perform bias analysis on discourse trees produced by models trained with left-first states against random states. We introduce a simple metric for detecting heaviness bias, by calculating the depth difference between the left-most and the right-most leaf nodes and subtracting the expected difference from the gold tree. A higher value indicates the predicted trees are more right-heavy than the gold trees.

$$d_i = \text{Depth}_{pred}(EDU_i) - \text{Depth}_{gold}(EDU_i)$$

$$b = d_n - d_1$$

When the parser is trained with left-first examples, $b = 12.6$ (Table 1), indicating a bias towards right-heavy trees. This is expected due to right merges being merged last in the training examples, thus creating an imbalance in the number of correct merges in the left and right sides of the tree in the training examples. On the other hand, when trained with random sampling, there is no such imbalance in the training dataset. And we see that there is no significant bias, with $b = 0.8$.

4 Conclusion

In this work, we adapted the sequence labeling framework to bottom-up RST parsing, introducing an easy-first parser conditioned on past decisions. We investigated methods to sample training examples for a non-directional parser, and proposed a dynamic oracle for our bottom-up parsing. We demonstrated that our parser achieves competitive results for bottom-up RST parsing.

Acknowledgements

We are grateful to the anonymous reviewers for their helpful feedback and suggestions. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at The University of Melbourne. This facility was established with the assistance of LIEF Grant LE170100200.

References

Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual RST discourse parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 292–304, Valencia, Spain. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 2016 International Conference on Learning Representations*, pages 1–8.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, Maryland. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India. The COLING 2012 Organizing Committee.

Grigorii Guz and Giuseppe Carenini. 2020. Coreference for discourse parsing: A neural approach. In *Proceedings of the First Workshop on Computational Approaches to Discourse*, pages 160–167, Online. Association for Computational Linguistics.

Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for RST discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136, Los Angeles. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. Hilda: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2015. CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Naoki Kobayashi, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. 2020. Top-down RST parsing utilizing granularity levels in documents. In *AAAI 2020 : The Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Fajri Koto, Jey Han Lau, and Timothy Baldwin. 2021. Top-down discourse parsing via sequence labelling.

In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 715–726, Online. Association for Computational Linguistics.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 362–371, Austin, Texas. Association for Computational Linguistics.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy. Association for Computational Linguistics.

Amandla Mabona, Laura Rimell, Stephen Clark, and Andreas Vlachos. 2019. Neural generative rhetorical structure parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2284–2295, Hong Kong, China. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text Interdisciplinary Journal for the Study of Discourse*, pages 243–281.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, USA.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on RST discourse parsing? a replication study of recent results on the RST-DT. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1319–1324, Copenhagen, Denmark. Association for Computational Linguistics.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2021. RST parsing from scratch. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1613–1625, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escárcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5, Denver, Colorado. Association for Computational Linguistics.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural RST parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 559–570, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural RST parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 559–570.

Nan Yu, Meishan Zhang, Guohong Fu, and Min Zhang. 2022. RST discourse parsing with second-stage EDU-level pre-training. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4269–4280, Dublin, Ireland. Association for Computational Linguistics.

Longyin Zhang, Fang Kong, and Guodong Zhou. 2021. Adversarial learning for discourse rhetorical structure parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3946–3957, Online. Association for Computational Linguistics.

Longyin Zhang, Yuqing Xing, Fang Kong, Peifeng Li, and Guodong Zhou. 2020. A top-down neural architecture towards text-level parsing of discourse rhetorical structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6386–6395, Online. Association for Computational Linguistics.

A Additional Results

We also report the results in Table 3 with the RST-Parseval Procedure. We include the reported results from Guz and Carenini (2020) as a reference. Their reported RST-Parseval scores beat other works, but uses the pre-trained language model SpanBERT.

A.1 Evaluation with RST-Parseval Procedure

Method	S	N	R	F
Bottom-Up				
Feng and Hirst (2014)*†	84.3	69.4	56.9	56.2
Ji and Eisenstein (2014)*†	82.0	68.2	57.8	57.6
Surdeanu et al. (2015)*†	82.6	67.1	55.4	54.9
Joty et al. (2015)*	82.6	68.3	55.8	54.4
Hayashi et al. (2016)*	82.6	66.6	54.6	54.3
Li et al. (2016)*	82.2	66.5	51.4	50.6
Braud et al. (2017)*	81.3	68.1	56.3	56.0
Yu et al. (2018) (1 run)*‡	85.5	73.1	60.2	59.9
Yu et al. (2018) (static)‡	85.8	72.6	59.5	59.0
Yu et al. (2018) (dynamic)‡	85.6	72.9	59.8	59.3
Our Work:				
Static ‡	86.7	73.2	60.5	60.0
Dynamic‡	86.8	73.6	60.6	60.1
Top-Down				
Kobayashi et al. (2020)*†‡	**87.0**	**74.6**	60.0	-
Koto et al. (2021) LSTM (static)‡	86.4	73.4	60.8	60.3
Koto et al. (2021) LSTM (dynamic)‡	86.6	73.7	**61.5**	**60.9**
Using Pretrained LM:				
Guz and Carenini (2020) (SpanBERT-CorefFeats)*†‡	88.1	76.1	63.6	-
Human	88.3	77.3	65.4	64.7

Table 3: Results over the test set calculated using micro-averaged F-1 on RST-Parseval. All metrics (S: `Span`, N: `Nuclearity`, R: `Relation`, F: `Full`) are averaged over three runs. "*" denotes reported performance. "†" and "‡" denote that the model uses sentence and paragraph boundary features, respectively.

A.2 Evaluation over Development Set

Method	S	N	R	F
Static	71.8	62.2	52.6	51.8
Dynamic	71.6	62.0	53.0	52.2

Table 4: Results over the development set calculated using micro-averaged F-1 on labeled attachment decisions (original Parseval). All metrics are averaged over three runs.