Bottom-Up Discourse Parsing via Sequence Labeling

Anonymous ACL submission

Abstract

We translate the sequence labeling framework, first introduced for top-down discourse parsing by Koto et al. (2021), to bottom-up discourse parsing. We introduce a novel parser that is not constrained by parsing direction (left-to-right or otherwise), and is conditioned on previous parsing decisions. We describe the unique training requirements of a (directionally) unconstrained parser and explore two different training procedures. Additionally, we introduce a novel dynamic oracle for unconstrained bottom-up parsing. Our proposed parser achieves state-of-the-art performance amongst bottom-up RST parsers.

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 hierarchy of discourse units (DU), with elementary discourse units (EDU) being the smallest unit (see Figure 1 as an example).

Most previous works have attempted to model the RST discourse tree based on the bottom-up paradigm (Ji and Eisenstein, 2014; Joty et al., 2015; Li et al., 2016; Yu et al., 2018). However, the bottom-up approaches used in the previous study have several drawbacks. First, tree construction of transition-based parser (Yu et al., 2018) is arguably less intuitive due to the requirement to transform the problem into such SHIFT and REDUCE actions. Secondly, CYK parsing is a greedy approach and requires computing all possible mergers efficiently.

In this work, we propose a conceptually simpler bottom-up discourse parsing framing it as a sequence labelling problem. We adopt this technique from the recent top-down discourse parsing (Koto et al., 2021). Our work is a novel bottom-up discourse parser that is not constrained to parsing in a particular direction, and parses conditioned on the context of past parsing decisions. Because the model is conditioned on the parsing history, we need to sample parsing trajectories to train the model. We compare two simple sample methods: (1) left to right, and (2) random. In addition, we introduce a dynamic oracle for unconstrained bottom-up parsing. Compared to previous bottom-up parsers, our method achieves the state of the art over the English RST Discourse Treebank. We make the source code available online [1].

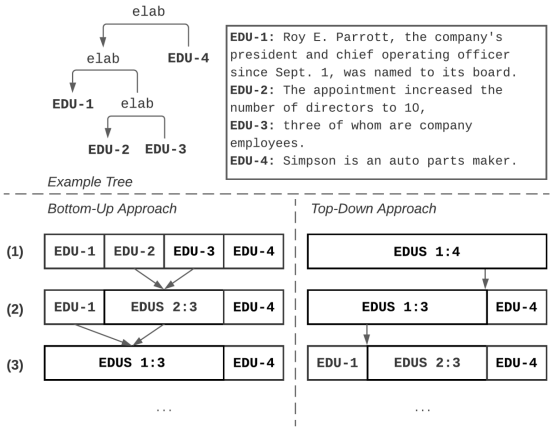

Figure 1: An example discourse tree (`evid` = evidence, `seq` = sequence). For this tree, we show the parsing states of the top-down (left) and bottom-up (right) approaches.

2 Bottom-Up RST Parsing

We construct RST trees in a bottom-up fashion: starting with a sequence of EDUs, the parser sequentially merges 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

[1] https://anonymous.4open.science/r/NeuralRST-BottomUp-C5DB

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 is part of 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 relation nuclearity separately with a second classifier after a merge is decided.

2.1 Parsing Context

We condition our parser on all discourse units in the text. Figure 1 demonstrates an example where the context of nearby discourse units is important. At a glance, it is difficult to decide whether EDU-2 is evidence for EDU-1 or EDU-3. With the context information that EDU-3 and EDU-4 are connected, it becomes more likely that EDU-2 is connected with EDU-1. A left-to-right parser (Ji and Eisenstein, 2014; Yu et al., 2018) that uses context-insensitive models must make a parsing decision without context. With no constraint on merge order, the parser can parse subtrees that it is most confident in first, and use the context of past decisions to inform difficult parsing decisions.

2.2 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 introduced in 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 is the MLP output from a bi-affine syntax dependency parser (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

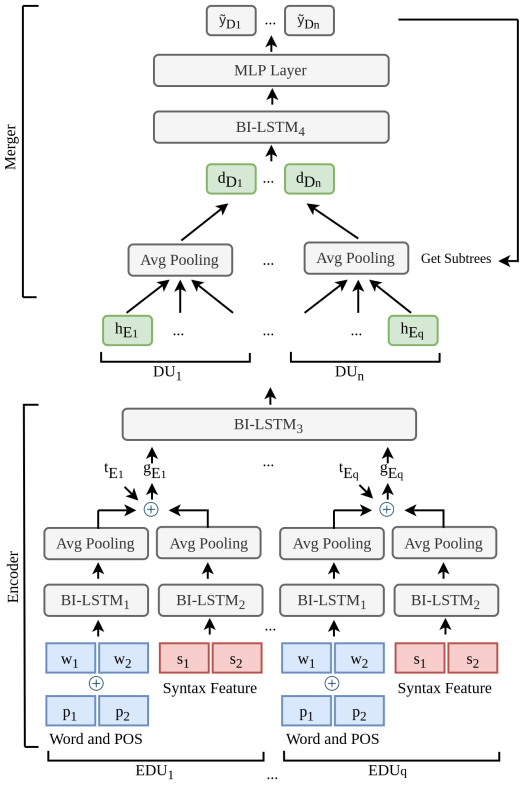

Figure 2: Architecture of the LSTM Model

EDU and the EDU type embedding t_{E_j}:

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

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). Our parse state consists of a sequence of discourse units, each of which is represented by averaging the encodings of the EDUs spanning the discourse unit:

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

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 discourse units and assign a binary label to each merge.

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

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

2.3 Nuclearity and Discourse Relation Prediction

Following Koto et al. (2021), we predict the nuclearity indication and discourse label after a merge is chosen. We feed the encodings d'_{ind}, d'_{ind+1} of the selected discourse units into a MLP layer to predict the nuclearity and discourse labels, where ind is the index of the left discourse unit chosen to be merged:

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

where $z_{nuc+dis}$ is the joint probability distribution over the nuclearity and discourse classes.

The final training loss of our model is the combination of the merging and nuclearity-discourse prediction loss:

$$\mathcal{L} = \mathcal{L}_{merge} + \mathcal{L}_{nuc+dis}$$

2.4 Merge Order in Training

The same RST tree can be constructed by different sequences of merges. Because each pair of merge candidates is evaluated in the context of the whole discourse unit sequence, different merge histories can lead to different predictions for the same merge candidate.

During training, it is intractable to train the model to predict each merge correctly with all possible merge histories. We propose to construct each training text using merges from the gold tree, and use the parse state in the merge sequence to train the parser.

We propose two methods for generating training sequences: (1) merging gold pairs left to right; and (2) merging gold pairs at random.

Training examples generated by executing gold merges left to right are unique for each gold tree, minimizing data variance. The parse states generated are equivalent to the states seen by a transition-based parser. However, the model will only see parse states created by a left to right parser. At test time, the parser is not limited to parsing in a left-to-right manner, and the model will see states outside of the training distribution.

Executing gold merges at random can generate different parse states even for the same gold tree. All gold merge sequences that constructs the gold tree are within the training data data distribution, at the cost of higher data variance.

We train our parser with both methods and analyze induced model biases.

2.5 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) for dependency parsing and adopted for training other types of 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 in the Appendix. 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.

The oracle assigns a merge order to each merge position in the document, defined as the earliest step the position is merged in all possible gold merge sequences. A position is an oracle action if the adjacent merge positions have a higher order. In such cases, the merge at the position precedes all merges containing the two discourse units in the merge.

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. We also use the same 18 coarse-grained discourse relationships.

3.2 Results

We use the Parseval metric for evaluating parser performance, described in the Appendix Section A. The hyperparameters used are also listed in the Appendix in Section B. We perform a feature selection study to find the best training procedure. Results

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

Table 1: Feature addition study over the development set to find the best configuration for our models. Presented results are the mean of the `Full` metric (micro-averaged F-score on labeled attachment decisions) and bias (depth difference between the left and right end of the tree) over the development set.

are presented in Table 1. All metrics are averaged over three random seeds on the development set, with a static oracle. We compared training with left-first state sequences and randomly-sampled state sequences; randomly-sampled state sequences resulted in a score of 51.8, a +4.5 improvement over training with left-first state sequences. We use random state sequences for the remainder of this section.

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) and Table 3 (RST-Parseval). The reported human performance is the score of human agreement from Joty et al. (2015) and Morey et al. (2017).

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 result demonstrates that a parser with context of 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.

3.3 Analysis

We perform bias analysis on discourse trees produced by models trained 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.

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

$$b = d_n - d_1$$

When the parser is trained with left-first examples, $b = 12.57$ (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 side and the right side of the tree in the training examples. On the other hand, when trained with random sampling, there is no significant bias, with $b = 0.8$.

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
Our Work:				
Static ‡	73.3	62.0	50.1	49.1
Dynamic‡	**73.6**	**62.3**	50.3	49.3
Top Down:				
Zhang et al. (2020)	67.2	55.5	45.3	44.3
Koto et al. (2021) LSTM (static)‡	72.7	61.7	50.5	49.4
Koto et al. (2021) LSTM (dynamic)‡	73.1	**62.3**	**51.5**	**50.3**
Human	78.7	66.8	57.1	55.0

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.

4 Conclusion

In this work, we adapted the sequence labeling framework to bottom-up RST parsing, introducing a parser conditioned on past decisions. We investigated methods to sample training examples for a context-sensitive parser, and proposed a dynamic oracle for our bottom-up parsing. We demonstrated that our parser achieves state-of-the-art results for bottom-up RST parsing, and is competitive with the state-of-the-art top-down parser.

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 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.

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.

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.

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.

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.

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 Evaluation

We use the standard Parseval metrics for RST parsing from (Marcu, 2000). Under 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. 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.

B Hyperparameters

We follow the hyperparameters used in (Koto et al., 2021). The GloVe embedding (Pennington et al., 2014) is used to word encodings in each EDU. We use CoreNLP (Manning et al., 2014) to tag POS and initialize each POS encoding as a random vector. The embedding dimension of words, POS tags, EDU type and syntax features are respectively 200, 200, 100 and 1200. 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, 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.

C Compute

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 and 32 hours in wall-clock time.

D Dynamic Oracle

Algorithm 1 Bottom-up Dynamic Oracle

function DYNORACLE(E, O, R)
  # For training only
  # E is list of EDUs
  # O is gold order for segmentation
  # 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 random() $> \alpha$ **then**
      $state = merge(state, id_{pred}$
      $r_{oracle} = \text{oracleLabel}(state, id_{pred})$
      $L = Loss(id_{gold}, r_{oracle}, id_{pred1}, r_{pred1})$
    else
      $state = merge(state, id_{gold}$
      $r_{gold} = \text{oracleLabel}(state, id_{gold})$
      $L = Loss(id_{gold}, r_{gold}, id_{pred2}, r_{pred1})$
    end if
  end while
end function

E Additional Results

E.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**
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.

E.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.