Improving Neural RST Parsing Model with Silver Agreement Subtrees

Naoki Kobayashi[1], **Tsutomu Hirao**[2], **Hidetaka Kamigaito**[1],
Manabu Okumura[1], **Masaaki Nagata**[2]
[1]Institute of Innovative Research, Tokyo Institute of Technology,
[2]NTT Communication Science Laboratories, NTT Corporation
{kobayasi@lr., kamigaito@lr., oku@}pi.titech.ac.jp
{tsutomu.hirao.kp, masaaki.nagata.et}@hco.ntt.co.jp

Abstract

Most of the previous Rhetorical Structure Theory (RST) parsing methods are based on supervised learning such as neural networks, that require an annotated corpus of sufficient size and quality. However, the RST Discourse Treebank (RST-DT), the benchmark corpus for RST parsing in English, is small due to the costly annotation of RST trees. The lack of large annotated training data causes poor performance especially in relation labeling. Therefore, we propose a method for improving neural RST parsing models by exploiting *silver data*, i.e., automatically annotated data. We create large-scale silver data from an unlabeled corpus by using a state-of-the-art RST parser. To obtain high-quality silver data, we extract *agreement subtrees* from RST trees for documents built using the RST parsers. We then pre-train a neural RST parser with the obtained silver data and fine-tune it on the RST-DT. Experimental results show that our method achieved the best micro-F1 scores for Nuclearity and Relation at 75.0 and 63.2, respectively. Furthermore, we obtained a remarkable gain in the Relation score, 3.0 points, against the previous state-of-the-art parser.

1 Introduction

Rhetorical Structure Theory (RST) (Mann and Thompson, 1987) is one of the most widely used theories for representing the discourse structure of a text as a tree. RST trees are a kind of constituent tree, whose leaves are Elementary Discourse Units (EDUs), i.e., clause-like units, and whose non-terminal nodes cover text spans consisting of either a sequence of EDUs or a single EDU. The label of a non-terminal node represents the attribution of a text span, i.e., *nucleus* (N) or *satellite* (S). A discourse relation is also assigned between two adjacent non-terminal nodes.

In most cases, RST parsers have been developed on the basis of supervised learning algorithms (Wang et al., 2017b; Yu et al., 2018; Kobayashi et al., 2020; Lin et al., 2019; Zhang et al., 2020), which require a high-quality annotated corpus of sufficient size. Generally, they train the following three components of the RST parsing: (1) structure prediction by splitting a text span consisting of contiguous EDUs into two smaller ones or merging two adjacent spans into a larger one, (2) nuclearity status prediction for two adjacent spans by solving a 3-class classification problem, and (3) relation label prediction for two adjacent spans by solving an 18-class classification problem (see Section 3.3 for details). However, it is costly to annotate RST trees for a huge collection of documents, and thus it is difficult to obtain a large amount of human-annotated data for RST parsing. As a result, research on RST parsing has focused on English, with the largest annotated corpus being the RST Discourse Treebank (RST-DT) (Carlson et al., 2001), although even this is still small with only 385 documents.[1]

Many RST parsing methods have recently been developed based on neural models (Ji and Eisenstein, 2014; Li et al., 2014a, 2016; Liu and Lapata, 2017; Braud et al., 2016, 2017). Among them, Kobayashi et al. (2020) is the current state-of-the-art system and is based on the neural top-down method. While its Span and Nuclearity scores achieved the highest level, its Relation score still has room for improvement. One of the reasons for its poor Relation score might be its small amount of training data for solving the 18-class classification problem.

Currently, we can refer to various studies on improving neural models for NLP tasks through acquiring large-scale synthetic training data, sometimes called *silver data*. Among them, one of the studies on Neural Machine Translation (NMT)

[1]We can find some exceptions for other languages such as Spanish (da Cunha et al., 2011) and German (Stede and Neumann, 2014).

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1600–1612
June 6–11, 2021.

(Sennrich et al., 2016) introduced a simple learning framework: first pre-train an NMT model with *silver data*, i.e., pseudo-parallel data generated by automatic back-translation, and then fine-tune it with *gold data*, i.e., real parallel data, to overcome the data sparseness problem. Since the frameworks successfully improved the NMT systems, it has become a standard approach.

Inspired by the above research, we propose a method for improving a student neural parser by exploiting large-scale *silver data*, thus generating RST trees using an automatic RST parser.[2] Specifically, we improve the state-of-the-art neural RST parser (Kobayashi et al., 2020), in terms of Relation, by employing another RST parser whose Relation score is also state-of-the-art (Wang et al., 2017b) as a *teacher* parser to generate the silver data. To yield high-quality silver data, we extract a collection of agreement subtrees (ASTs), which are common subtrees among multiple RST trees automatically parsed by the *teacher* parser with different seeds. Our method includes an efficient algorithm for extracting the agreement subtrees to handle large-scale data. We first pre-train the student parser by using the obtained *silver data*. We then fine-tune parameters of the parser on *gold data*, using the RST-DT. Experimental results on the RST-DT clearly indicate the effectiveness of our silver data. Our method obtained remarkable Nuclearity and Relation F_1 scores of 75.0 and 63.2, respectively.

2 Related Work

Early studies on RST parsing were based on traditional supervised learning methods with handcrafted features and the shift-reduce or CKY-like parsing algorithms (duVerle and Prendinger, 2009; Feng and Hirst, 2012; Joty et al., 2013, 2015; Feng and Hirst, 2014). Recently, Wang et al. (2017b) proposed a shift-reduce parser based on SVMs and achieved the current best results in classical statistical models on the RST-DT. The method first built nuclearity-labeled RST trees and then assigned relation labels between two adjacent spans consisting of a single or multiple EDUs.

Inspired by the success of neural networks in many NLP tasks, several neural network-based models have been proposed for RST parsing (Ji and Eisenstein, 2014; Li et al., 2014a, 2016; Liu and Lapata, 2017). Yu et al. (2018) proposed a shift-reduce parser based on neural networks and leveraged the information from their neural dependency parsing model within a sentence for RST parsing. The best Relation score on the RST-DT, i.e., F_1 of 60.2, was achieved with their method.

Recently, a top-down neural parser was proposed (Lin et al., 2019) for use only at the sentence-level. The method parses a tree in a depth-first manner with a pointer-generator network. Zhang et al. (2020) extended the method and applied it to document-level RST parsing. Kobayashi et al. (2020) proposed another top-down RST parsing method exploiting multiple granularity levels in a document and achieved the best Span and Nuclearity scores on the RST-DT, i.e., F_1 of 87.0 and 74.6, respectively.

Since the RST-DT, the largest treebank, contains only 385 documents, several studies have been conducted on overcoming the problem of a limited number of training data. Braud et al. (2016) leveraged multi-task learning not only with 13 related tasks as an auxiliary task but also for multiple views of discourse structures, such as Constituent, Nuclearity, and Relation. Braud et al. (2017) used multilingual RST discourse datasets that share the same underlying linguistic theory. Huber and Carenini (2019) adopted distant supervision with an auxiliary task of sentiment classification to create large-scale training data, i.e., they trained a two-stage RST parser (Wang et al., 2017a) with RST trees automatically built based on attention and sentiment scores from the Multiple-Instance Learning network, which was trained with a review dataset. However, these studies need other annotated corpora than the RST-DT, which means we still face the problem of being dependent on costly annotated corpora. Jiang et al. (2016) proposed a framework for enriching training data based on co-training to improve the performance for infrequent relation labels. However, the method failed to improve the overall Relation score, while they did not aim at improving the Span and Nuclearity scores.

Unsupervised RST parsing methods have also been proposed recently (Kobayashi et al., 2019; Nishida and Nakayama, 2020). Since they are unsupervised, they do not require any annotated corpora. However, they can predict only tree structures and

[2] Nguyen et al. (2020) proposed a similar approach in NMT and introduced a method named data diversification: it diversifies the training data by using multiple forward and backward translation models. We can find some weak supervision approaches for other discourse representation formalisms such as (Badene et al., 2019).

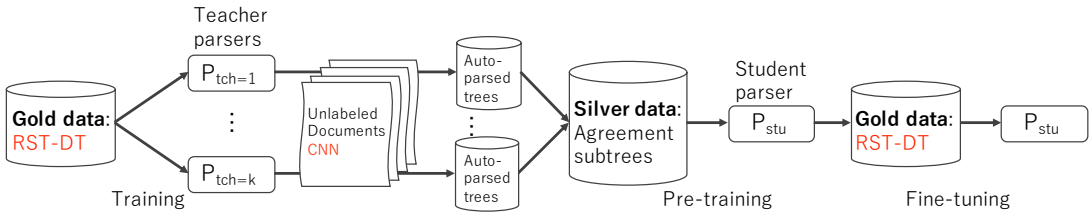

Figure 1: Overview of proposed method. In the subtree extraction step, the teacher RST parsers first annotate trees to unlabeled documents, and then the proposed subtree extraction method constructs large silver data. In the training step, the student parser is trained through pre-training and fine-tuning.

cannot predict nucleus and relation labels. Therefore, the predicted trees cannot be used for learning for predicting relation labels.

We should mention the relationship of our work with semi-supervised learning as a machine learning framework. First, the reason why we do not adopt self-training, where the student and teacher parsers are the same, but instead use two different parsers that rely on different parsing algorithms is that we can acquire instances that the student parser cannot correctly parse yet the teacher parser can parse as the training data. Second, using multiple different RST parsers in a semi-supervised manner in our work might seem reminiscent of co- or tri-training. While co- or tri-training is attractive, it is time consuming to repeat the step of alternately training multiple different neural network-based parsers many times. Thus, previous studies have focused on simplifying the repetition step in constituency and dependency parsing (McClosky et al., 2006; Yu et al., 2015; Pekar et al., 2014; Weiss et al., 2015; Li et al., 2014b).

We believe our method is similar to these simplified version as a semi-supervised framework with two different RST parsers.

3 Neural RST Parsing with Silver Data

3.1 Training Student Parser

Traditional semi-supervised learning frameworks, such as self-, co-, and tri-training, tend to iteratively train a student classifier with the training data that contains human-annotated (gold) data and iteratively added silver data. Since neural network-based models require a large amount of time for training, the iterative procedure is not suitable for training them. Furthermore, the training method may be affected by the bias problem in relation-label distribution because frequent labels in the original training data become yet more frequent in the future training data. For these reasons, we adopt a simple pre-training and fine-tuning strategy, which is inspired by the NMT research (Sennrich et al., 2016), to train a student RST parser.

Since early statistical RST parsing methods relied on handcrafted features, i.e., sentence-level features obtained from parse trees and document-level features, they require complete documents with complete sentences for their feature extraction. On the other hand, recent neural models do not necessarily need such features. Thus, we can exploit subtrees as training data for the neural networks.

Our method involves the following two steps: First, we extract a collection of ASTs from RST trees for each document in unlabeled data as the silver data. In this step, each document is first parsed using multiple *teacher* RST parsers with different seeds, trained with a gold dataset, the RST-DT. We then apply our algorithm for extracting the ASTs, which are common subtrees among multiple automatically parsed RST trees. In the second step, we pre-train the *student* RST parser with the collection of ASTs to complement the amount of training data. The parameters of the student parser are then fine-tuned on the RST-DT. Figure 1 shows an overview of our proposed method.

3.2 Extracting Agreement Subtrees

A good strategy for obtaining high-quality silver data is to get agreement among the results of multiple RST parsers. However, it is difficult to reach agreement for the entire RST trees at the document-level because their size is big. Thus, we believe we cannot collect enough silver data using agreement for the whole trees. On the other hand, we find that many subtrees agreed among multiple RST trees, even when the whole trees do not agree with each other. Accordingly, we extract ASTs as the silver data.

To create large-scale silver data, we need an efficient algorithm that extracts the ASTs, i.e., com-

Algorithm 1: Extracting Agreement Subtrees

```
   Input: trees
   Output: subtrees
1  AGREEMENT(root(tree))
2  subtrees ← FINDROOT(root(tree))
3  Function AGREEMENT(span):
4      if Len(span)=1 then
5          return True
6      else
7          if Count(span)=k then
8              S_c(span) ← True
9          else
10             S_c(span) ← False
11         S_l(span)←AGREEMENT(leftChild(span))
12         S_r(span)←AGREEMENT(rightChild(span))
13         S(span) ← S_c(span)∧S_l(span)∧S_r(span)
14         if S(span)=True then
15             return True
16         else
17             return False
18 Function FINDROOT(span):
19     subtrees ← list()
20     Function SUBFINDROOT(span):
21         if Len(span) < l_min then
22             return
23         else if Len(span) > l_max then
24             SUBFINDROOT(leftChild(span))
25             SUBFINDROOT(rightChild(span))
26         else //  l_min ≤ Len(span) ≤ l_max
27             if S(span)=True then
28                 subtrees.append(span)
29             else
30                 SUBFINDROOT(leftChild(span))
31                 SUBFINDROOT(rightChild(span))
32     SUBFINDROOT(span)
33     return subtrees
```

mon subtrees among multiple RST trees for a document. Note that we need to extract multiple maximal common subtrees among the RST trees. This requires a different algorithm from the maximum agreement subtree problem, which is well-known in bioinformatics (Deepak and Fernández-Baca, 2014). Thus, we develop the algorithm in Algorithm 1. This algorithm follows a tree-traversal algorithm and works with $O(n)$, where n indicates the number of nodes in an RST tree.

In the algorithm, a tree is represented as a fully-labeled nested span structure (see the example in Figure 2). The function AGREEMENT receives an arbitrary span as the input and returns a Boolean value indicating whether the subtree for the span is an AST. AGREEMENT first counts how many times the input span appears in the set of given RST trees and checks the status of the left and right children of the input span. Len() returns the length of the span and Count() returns the frequency of the fully-labeled span among the trees, which indicates how many trees agree on the subtree. The minimum and maximum values of Count() are 1 and k respectively, where k indicates the number of RST trees. The variables S_c, S_l, and S_r store the Boolean value for the input span and the left and right children of the span, respectively. Here, root, leftChild, and rightChild are functions for returning the root span and the left and right children spans, respectively. To obtain the status of each child, AGREEMENT calls itself with the child span. When the frequency of the input span is k, indicating all of the trees in the set agree on the span, and the status of the left and right children is True, indicating the left and right children are ASTs, the function returns True. Furthermore, the information regarding which subtrees are ASTs is stored in variable S during the execution of AGREEMENT.

The function FINDROOT returns the list of ASTs, based on the information in variable S, given by AGREEMENT. FINDROOT first checks the S(span), the Boolean value of the span. If it is True, the function appends the span, corresponding to the root node of an AST, to the output. Otherwise, it searches both left and right children for ASTs recursively. The function, therefore, lists all of the maximal ASTs in a depth-first fashion, based on the information in variable S.

In the algorithm, $l_{\min}$ and $l_{\max}$ are used to control the size of extracted ASTs. If the trees parsed using the multiple *teacher* parsers significantly differ from each other, the extracted ASTs tend to be small, which might become noise. To avoid such noise, we do not take into account subtrees with less than $l_{\min}$ EDUs. Excessively large subtrees are difficult to handle because they need a lot of time and space for training. Therefore, if the size of subtrees exceeds $l_{\max}$, the algorithm tries to find smaller ASTs from both their left and right children.

Initially, we call the function AGREEMENT with an arbitrary tree in multiple RST trees. We show an example of extracting ASTs in Figure 2. Assume the two trees at the left are from two RST parsers. The right part represents how the algorithm works with the top tree at the left as the input. In the figure, two subtrees consisting of spans (1,4) and (5,7) are extracted as ASTs since the frequency of

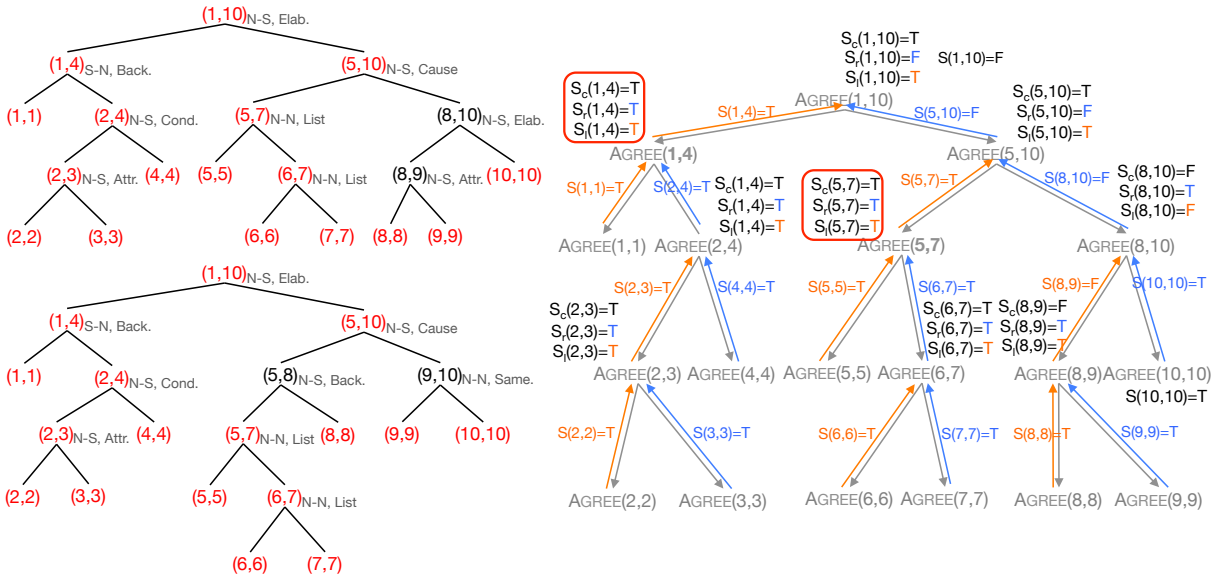

Figure 2: Example of extracting ASTs. Assume the left two trees are from two RST parsers. In the figure, red spans are shared by the two trees. The right part represents how the algorithm works with the top tree in the left as the input. Subtrees whose roots are spans (1,4) and (5,7) are extracted as ASTs. Note that we do not extract a subtree in extracted ASTs as an AST.

these two spans and all their descendant spans is 2, which is the number of given RST trees. Note that, while several spans, such as spans (2,3) and (6,7), are also common subtrees, we do not extract them since they are contained in either span (1,4) or (5,7).

3.3 Span-based Neural Top-down Parser as student Parser

As described in Section 3.1, the advantage of recent neural models is that they can utilize the annotation for partial documents, or subtrees, as training data. Among the neural models, the span-based neural top-down RST parsing method (Kobayashi et al., 2020) achieved the best Span and Nuclearity scores. Thus, we employ it as the student parser.

The method builds a tree by recursively splitting a text span into two smaller ones while predicting the nuclearity status and relation labels. As we explain below, the parser can be trained with arbitrary subtrees for spans consisting of EDUs.

Structure Prediction

For each position k in a span which consists of i-th EDU to j-th EDU, a scoring function, $s_{\text{split}}(i,j,k)$, is defined as follows:

$$s_{\text{split}}(i,j,k) = \mathbf{h}_{i:k}^{\top}\mathbf{W}_u\mathbf{h}_{k+1:j} + \mathbf{v}_{\ell}^{\top}\mathbf{h}_{i:k} + \mathbf{v}_r^{\top}\mathbf{h}_{k+1:j}, \quad (1)$$

where $\mathbf{W}_u$ is a weight matrix and $\mathbf{v}_\ell$ and $\mathbf{v}_r$ are weight vectors corresponding to the left and right spans, respectively. $\mathbf{h}_{i:k}$ and $\mathbf{h}_{k+1:j}$ are defined as follows: $\mathbf{h}_{i:k} = \text{MLP}_{\text{left}}(\mathbf{u}_{i:k})$, $\mathbf{h}_{k+1:j} = \text{MLP}_{\text{right}}(\mathbf{u}_{k+1:j})$, where MLP_* is the multi-layer perceptron. The vector representation of a span, $\mathbf{u}_{i:j}$, is obtained by feeding word embedding vectors into LSTMs. Then, the span is split at position k that maximizes Eq. (1):

$$\hat{k} = \text{argmax}_{k\in\{i,\ldots,j-1\}}[s_{\text{split}}(i,j,k)]. \quad (2)$$

Label Prediction

When splitting a span at position k, the score of the nuclearity status and relation labels for the two spans is defined as follows:

$$s_{\text{label}}(i,j,k,\ell) = \mathbf{W}_\ell \text{MLP}([\mathbf{u}_{i:k};\mathbf{u}_{k+1:j};\mathbf{u}_{1:i};\mathbf{u}_{j:n}]), \quad (3)$$

where $\mathbf{W}_\ell$ is a weight matrix and $\mathbf{u}_{1:i};\mathbf{u}_{j:n}$ are vector representation of left and right spans that appear outside the current focus. Then, the label that maximizes Eq. (3) is assigned to the spans:

$$\hat{\ell} = \underset{\ell\in\mathcal{L}}{\text{argmax}}[s_{\text{label}}(i,j,k,\ell)], \quad (4)$$

where $\mathcal{L}$ denotes a set of valid nuclearity status combinations, {N-S,S-N,N-N}, for predicting the nuclearity, and a set of relation labels, {Elaboration, Condition,. . .}, for predicting the relation. Accordingly, we solve a 3-class classification problem for the nuclearity labeling and an 18-class classification problem for the relation labeling. Note that the weight parameters $\mathbf{W}_\ell$ and MLP for the nuclearity and relation labeling are separately learned.

Parameter Optimization

All parameters, $\mathbf{W}_u$, $\mathbf{W}_\ell$, $\mathbf{v}_r$, $\mathbf{v}_\ell$, and the parameters for LSTMs are optimized by using margin-based learning. When the correct splitting position k^* and labels ℓ^* are given, loss functions for splitting and labeling are defined as follows:

$$\begin{aligned} &\max(0, 1+s_{\text{split}}(i,j,k^*)-s_{\text{split}}(i,j,\hat{k})), \\ &\max(0, 1+s_{\text{label}}(i,j,\hat{k},\ell^*)-s_{\text{label}}(i,j,\hat{k},\hat{\ell})). \end{aligned} \quad (5)$$

By minimizing the sum of the losses in each splitting point, the parameters are optimized.

3.4 Two-stage Parser as Teacher Parser

Since the student parser still has room for improvement in Relation, it is desirable to utilize another state-of-the-art parser based on a different parsing algorithm with a good Relation score. While the current best Relation score was achieved by NNDis-Parser (Yu et al., 2018), we cannot reproduce this score with their official code. Therefore, we employ the two-stage parser (Wang et al., 2017b), which obtained the second-best Relation score, as the *teacher* parser. This two-stage parser is based on a shift-reduce parsing algorithm and utilizes SVMs to determine actions to build trees. Since their SVMs are optimized by a dual coordinate descent method, we build multiple two-stage parser models with different seeds to obtain enough agreement between teacher parsers,[3] and create silver data by the agreement among the parsers.

4 Experiments

4.1 Datasets

We used the RST-DT to evaluate the performance of our student RST parser and compared it with state-of-the-art parsers. It is officially divided into 347 documents as the training dataset and 38 documents as the test dataset. Since there is no development dataset, we used a part of the training dataset, 40 documents, as the development dataset by following the previous study (Heilman and Sagae, 2015). By following conventional studies, we used gold EDU segmentation for the RST-DT. The training and development datasets were used as gold data to fine-tune our student parser.

To obtain silver data for pre-training, we used the CNN dataset (Hermann et al., 2015). To parse each document, we split sentences into EDUs by using the Neural EDU Segmenter (Wang et al., 2018)[4] and applied the two-stage parser.

Tree type	$l_{\min}$	# of trees	# of nodes
DT	-	91,536	8,162,114
ADT	-	2,142	57,940
AST	5	534,352	4,087,989
	6	387,636	3,501,125
	7	290,532	3,015,605
	8	223,101	2,611,019
	9	175,709	2,279,275
	10	140,384	1,996,675

Table 1: Number of trees and nodes in each type of silver data obtained from the CNN dataset. Note that we utilized only 91,536 documents that could go through the pre-processing with the CoreNLP toolkit.

4.2 Settings

$l_{\min}$ and $l_{\max}$ for AST extraction: Since the number of EDUs for a document in the RST-DT is from 7 to 240, we selected $l_{\min}$ with a range from 5 to 10 and set $l_{\max}$ to 240. Based on the results for the development dataset, $l_{\min}$ was fixed to 9 (see Appendix A for details).

Student Parser: We used the official code of the span-based neural top-down parsing method.[5] The dimension of the hidden layers was set to 500. We trained the model in 5 and 10 epochs for pre-training and fine-tuning, respectively. Other parameters of the model and an optimizer were the same as those used by Kobayashi et al. (2020) (see Appendix E for details).

Kobayashi et al. (2020) achieved the best results in the D2P2S2E setting, training the models in three levels of granularity, i.e., paragraph trees for documents, sentence trees for paragraphs, and EDU trees for sentences. This setting requires us to train many models corresponding to multiple granularity levels. To simplify this, we trained only the model for building an RST tree whose leaves are EDUs for a document, which corresponds to their D2E setting. In decoding, we split spans at sentence and paragraph boundaries to make the setting closer to D2P2S2E.

We also used ensemble decoding by following

[3] We could not adopt multiple different parser architectures, for example, the Two-stage parser and Span-based parser, because the agreement was low.

[4] https://github.com/PKU-TANGENT/NeuralEDUSeg

[5] https://github.com/nttcslab-nlp/Top-Down-RST-Parser

Model	Average				Ensemble			
	S	N	R	F	S	N	R	F
SBP	86.3	73.1	57.6	57.3	87.1	74.6	60.0	59.6
SBP+DT	**86.9**	74.1	61.8	61.0	**87.4**	74.7	62.7	61.7
SBP+ADT	86.6	73.5	59.5	58.8	86.9	74.3	60.5	59.7
SBP+AST	86.8	**74.7**	**62.5**	**61.8**	87.1	**75.0**	**63.2**	**62.6**

Table 2: Micro-averaged F_1 scores of span-based neural top-down parser with or without silver data on the test dataset of the RST-DT. S, N, R, and F represent Span, Nuclearity, Relation, and Full scores, respectively. The best score for each metric in the average and ensemble settings is indicated with **bold**.

Kobayashi et al. (2020). Since it takes a large amount of time to train multiple models in pre-training, we trained only a single model in the pre-training stage, while multiple models were trained in the fine-tuning stage with the pre-trained model as the initial state.

Teacher parser:[6] We used the official code of the two-stage parsing method[7] and re-trained it four times with different random seeds. A smaller value of k made reliability of the agreement lower since we could not exclude coincidentally agreed trees. On the other hand, a larger value required us more time to create silver data, while the reliability of the agreement is high. Thus, we set k to 4, that is a moderate number in terms of both the reliability of the agreement and the data creation time.

4.3 Evaluation Metrics

By following previous studies (Sagae and Lavie, 2005), we transformed RST-trees into right-heavy binary trees and evaluated system results with micro-averaged F_1 scores of Span, Nuclearity, Relation, and Full, based on RST-Parseval (Marcu, 2000). Span, Nuclearity, Relation, and Full were used to evaluate unlabeled, nuclearity-labeled, relation-labeled, and fully-labeled tree structures, respectively. Since Morey et al. (2017) made a suggestion to use a standard parseEval toolkit for evaluation, we also report the results using this in Appendix C.

4.4 Compared Methods

To demonstrate the effectiveness of our proposed method, we pre-trained the span-based neural top-down parser, i.e., our student parser, in various settings for creating the silver data and compared the performance after fine-tuning on the RST-DT. Table 1 summarizes the statistics of the different types of silver data. 'DT' denotes RST trees obtained by using a single two-stage parser. The number of RST trees is the same as that of documents in the CNN dataset. 'ADT' denotes agreement document-level RST trees, i.e., the cases in which the parsers built the same trees for the whole document. 'AST' denotes ASTs of RST trees obtained from the teacher parsers.

5 Results and Discussion

5.1 Different Methods for Constructing Silver Data

Table 2 shows the average and ensemble scores with five models for different types of silver data. In the table, SBP indicates the results obtained from the original span-based neural top-down parser, which means the parser was trained only with the RST-DT; this setting is without any silver data.

With AST as the silver data, performance in all metrics improved against the baseline. In most metrics, AST achieved the best scores. In particular, the gains in Relation and Full were impressive. DT and ADT, which consist of document-level RST trees, also outperformed the baseline. However, the gains against the baseline were smaller than those by AST. We believe this is related to the size and quality of the silver data. The number of trees and nodes in ADT is only 2,142 and 57,940, respectively, while AST has 175,709 trees and 2,279,275 nodes. Thus, a small number of silver data for pre-training is not effective. On the other hand, while DT has only 91,536 trees, the number of their nodes is huge, at about 8,000 K. The lower score of DT would come from unreliable parse trees contained in the silver data built by a single teacher parser. As described above, to pre-train

[6] We show the results for a case of using SBP as a teacher parser in Appendix B.

[7] https://github.com/yizhongw/StageDP

the student parser, we do not need to use the entire RST trees for documents. Thus, AST, with a large collection of RST subtrees, is more effective than the other approaches. Since the training time depends on the number of nodes contained in the data, SBP+AST can be learned in a quarter of the time required by SBP+DT. Consequently, AST has another advantage against DT.

Furthermore, the performance of averaging five models was greatly improved by pre-training with the silver data. The gains against the baseline were larger than those for 'Ensemble,' and the differences between their performances became small. The neural model tends to converge to a different local optimum solution by mini-batch training, so the convergence is not stable when the data size is small. Pre-training can improve this. This is another advantage of pre-training with silver data.

We also compare the results of our parser pre-trained with AST with and without fine-tuning in Appendix D.

5.2 Effect of Data Size

To investigate how the data size of AST for pre-training affects the performance, we show Span, Nuclearity, Relation, and Full scores while varying the size in Figure 3. Span scores showed only small gains even by increasing the amount of data because identifying splitting points for spans is a simple 2-class classification problem. On the other hand, identifying nuclearity and relation labels is a multi-class classification problem. Thus, we believe we need more training data than that for identifying splitting points. In particular, the Relation score could be improved with more silver data.

5.3 Detailed Analysis of Relation Labeling

To investigate the effectiveness of SBP+AST in more detail, we show Relation F_1 scores for relation labels with SBP, SBP+AST, and the two-stage parser in Figure 4. The results of SBP and SBP+AST were obtained from a five-model ensemble. In most relation labels, since the two-stage parser, the teacher parser, is comparable or superior to SBP, i.e., the student parser, the performance of SBP+AST can be improved. It finally outperformed the two-stage parser by introducing pre-training with silver data, even for less frequent relation labels. Furthermore, SBP+AST can correctly parse for some relation labels that the student parser cannot handle, by acquiring training instances with the help of the teacher parser.

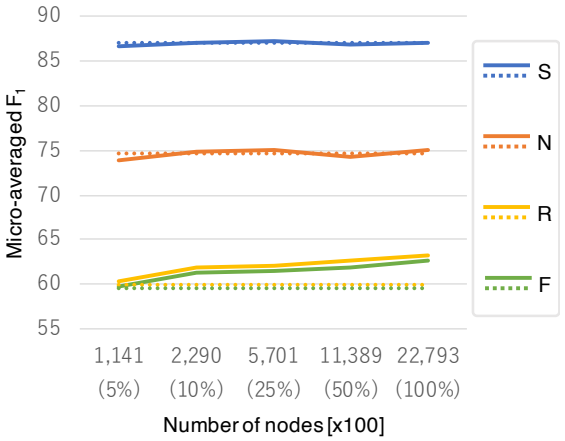

Figure 3: Results of changing the data size used for pre-training. The scores are from a five-model ensemble with SBP+AST. The solid lines represent the scores for SBP+AST by changing the data size. The dotted lines represent SBP, which does not depend on the size.

Model	S	N	R	F
Two-stage Parser	86.0	72.4	59.7	58.8
NNDisParser*	85.5	73.1	60.2	59.9
NNDisParser	85.9	72.5	59.5	58.9
SpanBasedParser	**87.1**	74.6	60.0	59.6
SBP+AST	**87.1**	**75.0**	**63.2**	**62.6**

Table 3: Comparison of state-of-the-art parsers. * indicates reported scores. The best score in each metric is indicated in **bold**. Our model is statistically significantly better than underlined scores at p-level $< \mathbf{0.01}$ in pairwise comparison.[8]

5.4 Comparison with state-of-the-art parsers

Finally, we compare our SBP+AST with the ensemble to current state-of-the-art parsers. Table 3 shows the micro-averaged F_1 scores. We used Paired Bootstrap Resampling (Koehn, 2004) for the significance test. We can see that our method achieved the best scores except for Span. The gains against the previous best scores were 0.4, 3.0, and 2.7 points for Nuclearity, Relation, and Full, respectively. In particular, the gains for Relation and Full are remarkable.

[8]Since the previous best scores for Relation and Full are reported scores, and we could not obtain the authors' data, we could not perform a significance test against them.

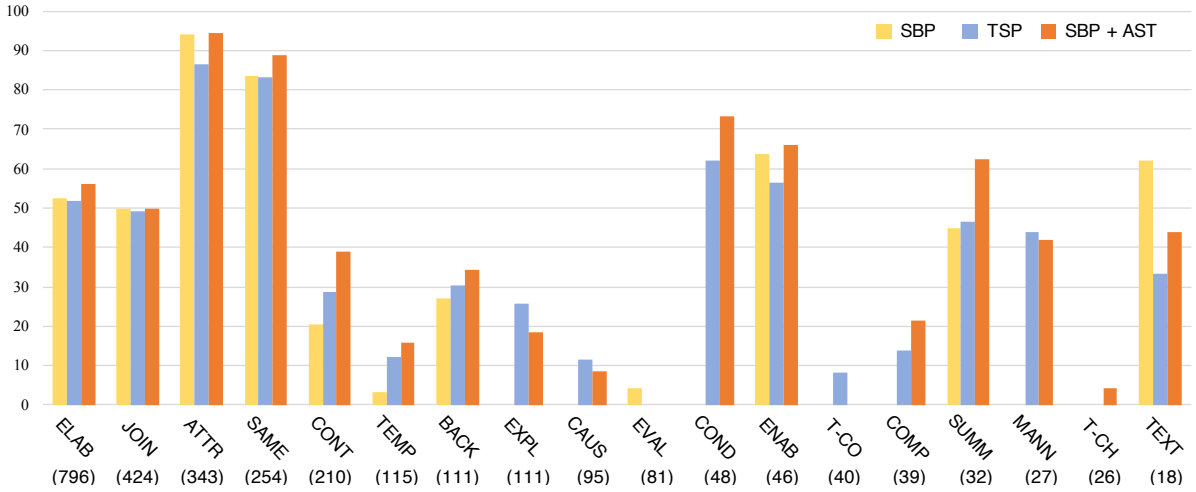

Figure 4: F_1 scores of SBP, SBP+AST, and the two-stage parser (TSP) for each relation label: **ELAB**ORATION, **JOIN**T, **ATTR**IBUTION, **SAME**-UNIT, **CONT**RAST, **TEMP**ORAL, **BACK**GROUND, **EXPL**ANATION, **CAUS**E, **EVAL**UATION, **COND**ITION, **ENAB**LEMENT, **T**OPIC-**COM**MENT, **COMP**ARISON, **SUMM**ARY, **MANN**ER-MEANS, **T**OPIC-**CH**ANGE, and **TEXT**UAL-ORGANIZATION. Relation labels are arranged in descending order of their frequency, shown in parenthesis under their label.

6 Conclusion

To solve the problem of the limited amount of training data available for neural RST parsing, we proposed a method of exploiting agreement subtrees as silver data: We pre-train a parser with the silver data and fine-tune it with the gold data. We also presented an algorithm that efficiently extracts overlapping subtrees as the agreement subtrees from multiple trees.

Experimental results on the RST-DT demonstrated that our method significantly improves the performance of relation-labeled and fully-labeled F_1 scores, which are strongly affected by data sparseness due to a small number of training data. Furthermore, the results showed that our method achieves the state-of-the-art nuclearity-labeled, relation-labeled, and fully-labeled F_1 scores.

References

Sonia Badene, Kate Thompson, Jean-Pierre Lorré, and Nicholas Asher. 2019. Data programming for learning discourse structure. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 640–645, Florence, Italy. Association for Computational Linguistics.

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.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1903–1913, Osaka, Japan. The COLING 2016 Organizing Committee.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Iria da Cunha, Juan-Manuel Torres-Moreno, and Gerardo Sierra. 2011. On the development of the RST Spanish treebank. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 1–10, Portland, Oregon, USA. Association for Computational Linguistics.

Akshay Deepak and David Fernández-Baca. 2014. Enumerating all maximal frequent subtrees in collections of phylogenetic trees. *Algorithms Mol. Biol.*, 9:16.

David duVerle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 665–673, Suntec, Singapore. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 60–68, Jeju Island, Korea. Association for Computational Linguistics.

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.

Michael Heilman and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Patrick Huber and Giuseppe Carenini. 2019. Predicting discourse structure using distant supervision from sentiment. 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 2306–2316, Hong Kong, China. 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.

Kailang Jiang, Giuseppe Carenini, and Raymond Ng. 2016. Training data enrichment for infrequent discourse relations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2603–2614, Osaka, Japan. The COLING 2016 Organizing Committee.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 486–496, Sofia, Bulgaria. 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 *Proceedings of the 2020 Conference on Artificial Intelligence for the American*, pages 8099–8106, New York, America.

Naoki Kobayashi, Tsutomu Hirao, Kengo Nakamura, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. 2019. Split or merge: Which is better for unsupervised RST parsing? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5797–5802, Hong Kong, China. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014a. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069, Doha, Qatar. 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.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014b. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 457–467, Baltimore, Maryland. 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.

Yang Liu and Mirella Lapata. 2017. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1298, Copenhagen, Denmark. Association for Computational Linguistics.

W.C. Mann and S.A Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC/ISI.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

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.

Xuan-Phi Nguyen, Shafiq Joty, Kui Wu, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. In *Advances in Neural Information Processing Systems*, volume 33, pages 10018–10029. Curran Associates, Inc.

Noriki Nishida and Hideki Nakayama. 2020. Unsupervised discourse constituency parsing using viterbi em. *Transactions of the Association for Computational Linguistics*, pages 215–230.

Viktor Pekar, Juntao Yu, Mohab El-karef, and Bernd Bohnet. 2014. Exploring options for fast domain adaptation of dependency parsers. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 54–65, Dublin, Ireland. Dublin City University.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Manfred Stede and Arne Neumann. 2014. Potsdam commentary corpus 2.0: Annotation for discourse research. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 925–929, Reykjavik, Iceland. European Language Resources Association (ELRA).

Xinhao Wang, James Bruno, Hillary Molloy, Keelan Evanini, and Klaus Zechner. 2017a. Discourse annotation of non-native spontaneous spoken responses using the Rhetorical Structure Theory framework. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 263–268, Vancouver, Canada. Association for Computational Linguistics.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017b. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188, Vancouver, Canada. Association for Computational Linguistics.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967, Brussels, Belgium. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China. Association for Computational Linguistics.

Juntao Yu, Mohab Elkaref, and Bernd Bohnet. 2015. Domain adaptation for dependency parsing via self-training. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 1–10, Bilbao, Spain. 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.

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.

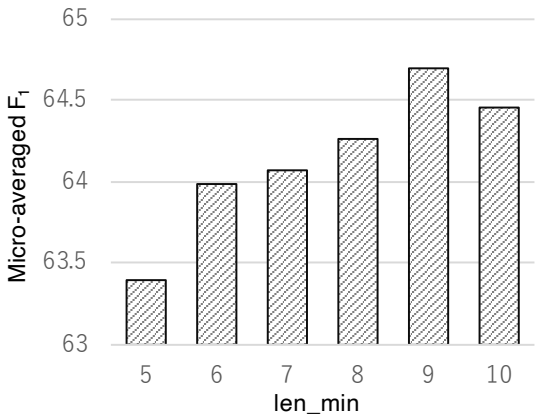

Figure 5: Fully-labeled F_1 scores with different l_{min} on the development dataset.

Model	S	N	R	F
SBP+DT (SBP)	87.1	74.5	60.3	59.7
SBP+DT (TSP)	87.4	74.7	62.7	61.7
SBP (Kobayashi et al., 2020)	87.1	74.6	60.0	59.6

Table 4: Comparison of teacher parsers.

A Effects of Parameter l_{min}

Figure 5 shows fully-labeled F_1 scores in changing l_{min} on the development dataset. From the figure, it is clear that the F_1 score was changed with l_{min}. The best F_1 score was achieved by $l_{min} = 9$. The results indicate that a large number of smaller subtrees prevents better pre-training of SBP. A small number of larger subtrees is also not useful for the pre-training.

B Performance of a case when the Span-based parser was used both as teacher and student parsers

We used different parsers for teacher and student parsers. In this section, we examine the setting of using the Span-based parser both as teacher and student parsers. We compare DT (SBP) and DT (TSP) as the silver datasets used for pre-training and show the results in Table 4. The results show that SBP+DT (SBP) does not obtain any gain of performance compared to SBP, that does not use any silver dataset. The better results with SBP+DT (TSP) demonstrate the effectiveness of using different types of parsers for teacher and student parsers.

C Performance with Original Parseval

In this paper, we used the gold EDU segmentation following conventional studies and evaluated the model performance for binarized trees with RST-Parseval (Marcu, 2000). Morey et al. (2017) reported that when evaluating the performance of binarized trees over manual EDU segmentation, the level of agreement between RST trees is artificially raised. To avoid this, they recommended using the original Parseval for the trees of label-attachment decisions. Following them, we evaluated our models with the original Parseval[9] and show the results in Table 5.

The results show the same tendency as that when employing RST-Parseval as the evaluation metrics. That is, SBP+AST obtained the best results for Nuclearity, Relation, and Full.

Model	S	N	R	F
(Feng and Hirst, 2014)*	68.6	55.9	45.8	44.6
(Ji and Eisenstein, 2014)*	64.1	54.2	46.8	46.3
(Wang et al., 2017b)**	72.0	60.5	50.4	48.2
(Yu et al., 2018)**	71.8	60.3	49.4	48.4
(Zhang et al., 2020)	67.2	55.5	45.3	44.3
(Kobayashi et al., 2020)**	**74.1**	63.7	48.8	47.9
SBP+AST	**74.1**	**64.7**	**54.1**	**52.7**

Table 5: Micro-averaged F_1 scores with the original Parseval (Morey et al., 2017). * indicates the reported scores in (Morey et al., 2017). ** indicates the scores for the re-produced models.

D Performance of Parser with Pre-training Alone

In our method, we applied both pre-training and fine-tuning to the target neural parser because it is the conventional way to improve neural network-based models. However, this might be different from the usual way of re-training models based on traditional supervised learning in a semi-supervised fashion. To investigate whether the approach with both pre-training and fine-tuning is effective, we compared other training methods, specifically, pre-training alone with the CNN and with both the CNN and the RST-DT, and the comparison results are shown in Table 6. The scores of Nucleus, Relation, and Full were not statistically significantly different from each other, which indicates that the difference between the two methods is minimal.

[9]We utilized Morey's code, available at https://github.com/irit-melodi/educe/.

Model	S	N	R	F
Pretrain w/ RST-DT	85.9	72.4	59.3	58.8
Pretrain w/o RST-DT	86.3	72.3	59.1	58.5
SBP+AST (Ave.)	86.8	74.7	62.5	61.8
SBP+AST (Ens.)	87.1	75.0	63.2	62.6
Two-stage parser	86.0	72.4	59.7	58.8

Table 6: Comparison of the training methods (pre-training alone v.s. pre-training and fine-tuning)

.

Furthermore, compared with the performance of the two-stage parser, i.e., our *teacher* parser, it is confirmed that our silver data provides adequate quality. Compared with the fine-tuned models, it is also confirmed that fine-tuning improves performance.

E Hyperparameters

Table 7 shows the hyperparameters of SBP+AST.

Computing Infrastructure	Nvidia TITAN RTX
Training duration (Pre-train) with $l_{\min}$=**9**	20 hours
Training duration (Fine-tune)	30 minutes
Hyperparameters	
number of epochs (Pre-train)	5
number of epochs (Fine-tune)	10
batch size (# of documents)	10
embedding	GloVe and ELMo
hidden size	[250, **500**]
dropout	0.4
learning rate scheduler	Exponential decay
scheduler reduction factor	0.99
optimizer	Adam
learning rate	0.001
gradient clipping	5.0
validation criteria	Micro-averaged F_1 of Relation
$l_{\min}$	[5, 6, 7, 8, **9**, 10]
$l_{\max}$	240

Table 7: SBP+AST search space. Values in **bold** indicate best assignments.