Top-Down RST Parsing Utilizing Granularity Levels in Documents

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

Abstract

Some downstream NLP tasks exploit discourse dependency trees converted from RST trees. To obtain better discourse dependency trees, we need to improve the accuracy of RST trees at the upper parts of the structures. Thus, we propose a novel neural top-down RST parsing method. Then, we exploit three levels of granularity in a document, paragraphs, sentences and Elementary Discourse Units (EDUs), to parse a document accurately and efficiently. The parsing is done in a top-down manner for each granularity level, by recursively splitting a larger text span into two smaller ones while predicting nuclearity and relation labels for the divided spans. The results on the RST-DT corpus show that our method achieved the state-of-the-art results, 87.0 unlabeled span score, 74.6 nuclearity labeled span score, and the comparable result with the state-of-the-art, 60.0 relation labeled span score. Furthermore, discourse dependency trees converted from our RST trees also achieved the state-of-the-art results, 64.9 unlabeled attachment score and 48.5 labeled attachment score.

Introduction

A discourse structure of a document can be represented as a tree, like a syntactic structure of a sentence being represented as a tree. Rhetorical structure theory (RST) (Mann and Thompson 1987) is well-known and has been widely studied for representing a document as a tree. An RST tree is a type of constituent tree, whose terminal nodes (leaves) are elementary discourse units (EDUs), clause-like units, and whose non-terminal nodes represent the nuclearity status, *nucleus* or *satellite*, for the span that consists of a sequence of EDUs or a single EDU. The span dominated by a nucleus is more essential than the one dominated by a satellite. That is, the satellite has a role of supporting the nucleus. Furthermore, rhetorical relations are defined between two adjacent spans. A mono-nuclear relation, such as "Elaboration" or "Condition", is assigned between a nucleus and its satellite, and a multi-nuclear relation, such as "Same-unit" or "Topic-change", is assigned between two nuclei.

RST trees have important roles in natural language processing (NLP) tasks, such as summarization (Marcu 1998; Hirao et al. 2013), question-answering (Jansen, Surdeanu, and Clark 2014), sentiment analysis (Bhatia, Ji, and Eisenstein 2015), and text categorization (Ji and Smith 2017). Most of them rely on discourse dependency trees, that explicitly express parent-child relationships between EDUs, converted from RST trees. When determining the parent of an EDU, the following procedure is usually applied: (1) Find the nearest satellite from the ancestors of the EDU. (2) From the sibling nucleus of the nearest satellite, follow only nuclei downward in the tree and find the leftmost descendant EDU. (3) Then assign the EDU as its parent. Fig. 1 shows an example. Here, consider a case where a parser built an RST tree with swapped N/S labels for the children of the root node. Since the number of incorrect labeling is only two, the nuclearity labeled span score is .85(=11/13). However, parent-child relationships between EDUs are drastically changed by the swap. The unlabeled attachment score (UAS) of the discourse dependency tree is .43(=3/7). On the other hand, in a case of swapping N/S labels for the parents of e_4 and e_5, the nuclearity labeled span score is still .85(=11/13) but we obtain better UAS, .71(=5/7). Thus, inaccuracies in the upper parts of an RST tree cause a critical problem for the dependency conversion. That is, we need to develop an RST parser that can build RST trees with accurate upper parts to obtain good discourse dependency trees for improving the

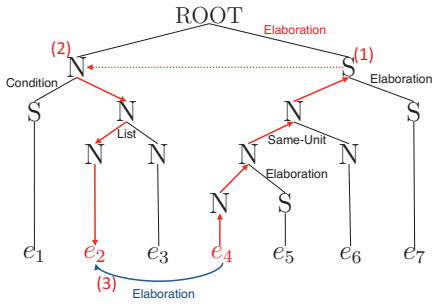


Figure 1: The procedure of determining the parent of e_4. N, S and e indicate nucleus, satellite and EDU, respectively.

performance of the downstream NLP tasks[1].

Since RST trees are a kind of constituent tree, any algorithms proposed for syntactic parsing can also be applied for RST parsing. However, because the number of EDUs in a document is much larger than that of words in a sentence, CKY-based parsing algorithms that incur heavy computational costs are not preferable for RST parsing even though these can obtain optimal trees. Therefore, most RST parsers use transition-based algorithms with linear time complexity. In fact, the best results on the standard benchmark dataset, RST discourse treebank (RST-DT) (Carlson, Marcu, and Okurowski 2001), are obtained using a transition-based method (Wang, Li, and Wang 2017). However, as Hong and Huang (2018) described, transition-based parsers tend to be easily affected by local errors during parsing, due to the small search space. In transition-based parsers, with either left-to-right or right-to-left search direction, local errors near leaf nodes may propagate to the parsing results at the upper parts, even though the nodes at the upper parts are more important than those near leaf nodes for deciding global tree structures.

To parse a document without suffering from such error propagation, we propose a top-down RST parser, which is inspired by the span-based neural syntactic parser (Stern, Andreas, and Klein 2017). Then, we exploit three levels of granularity in a document to parse a document efficiently and accurately. We build RST trees at each granularity level by recursively splitting a larger text span into smaller ones while predicting their nuclearity status and the relation. The parser builds an RST tree whose leaves are paragraphs for a document, RST trees whose leaves are sentences for each paragraph, and RST trees whose leaves are EDUs for each sentence. We then obtain an RST tree for the whole document by merging the trees together, namely, replacing leaves of upper-level RST trees with lower-level RST trees that were already constructed.

The experimental results obtained on RST-DT demonstrated that our method improved the upper parts of the RST tree structures, as we expected. The improvements derive the state-of-the-art unlabeled span score and nuclearity labeled span score, F_1 of 87.0 and F_1 of 74.6, and the comparable relation labeled span score, F_1 of 60.0, with the state-of-the-art. Moreover, discourse dependency trees converted from our RST trees also achieved the state-of-the-art unlabeled attachment score (UAS) and labeled attachment score (LAS), 64.9 and 48.5, respectively.

Related Work

Early studies on RST parsing employed traditional statistical models with handcrafted features. DuVerle and Prendinger (2009) proposed a bottom-up greedy parser that recursively merges adjacent spans based on SVMs. Feng and Hirst (2012) employed two levels of granularity in a document, i.e., intra- and inter- multi-sentence parsing models. Joty et al. (2013) introduced the CKY algorithm to obtain optimal trees with two levels of granularity in a document. Since the CKY algorithm requires $O(n^3)$ time complexity, Feng and Hirst (2014) proposed a bottom-up greedy parser with CRFs as a local classifier. The method achieved the former best results (Morey, Muller, and Asher 2017). Recently, Wang et al. (2017) proposed a shift-reduce parser based on SVMs and achieved the current best results. The method first builds naked RST trees, trees holding only nuclearity status, then assigns rhetorical relation labels between two adjacent spans.

Inspired by the success of neural networks in NLP tasks, several neural network-based RST parsers have been proposed. Ji and Eisenstein (2014) proposed a shift-reduce parser that uses a feedforward neural network. Li et al. (2014) proposed a CKY-based parser that uses recurrent neural networks to learn the representation of EDUs. Li et al. (2016) proposed another CKY-based parser that utilizes hierarchical BiLSTMs to learn the representation of text spans, and tensor-based transformation functions to compose adjacent text spans. These methods are sophisticated but the performances are not so good (Morey, Muller, and Asher 2017). Liu and Lapata (2017) proposed a transition-based neural parser as an extension of Fang and Hirst's (2014). The parser uses LSTMs to learn the representation of text spans. However, the performances are outperformed by the state-of-the-art statistical methods.

Recently, a top-down RST parsing method based on pointer networks (Vinyals, Fortunato, and Jaitly 2015) which are variants of attention-based encoder-decoder has been proposed for the sentence level (Lin et al. 2019). Their method is similar to ours in that both utilize the top-down parsing but the focus of parsing is quite different from ours. Their focus is only on sentence-level RST parsing while our focus is on the whole document-level RST parsing. Since their method is specific to parsing a sentence, it would be difficult to extend it so as to parse a whole document. To parse a document in a top-down manner, a parser needs to split longer spans consisting of about 50 EDUs, the average number of EDUs in a document in RST-DT. However, it is well known that the performance of the attention-based encoder-decoder seriously degrades when applied to long sequences (Koehn and Knowles 2017).

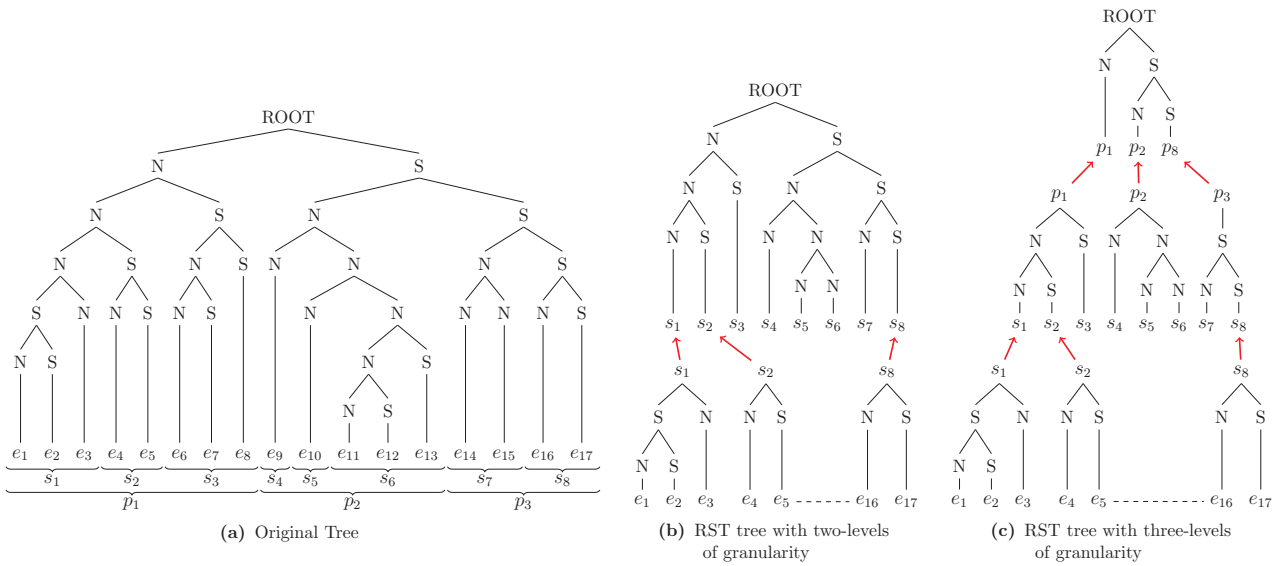
Neural Top-down RST Parsing

In general, RST trees are converted into right-branching binary trees by applying the right-heavy binarization procedure (Morey, Muller, and Asher 2017), as in constituent parsing. Note that the transformation is the de-facto standard binarization utilized by recent state-of-the-art document-level RST parsing researches, while there may be a few exceptions. Thus, we use top-down parsing (Stern, Andreas, and Klein 2017) that recursively splits a text span into smaller two ones for each granularity level with neural models.

Three Levels of Granularity in Document

Conventional RST parsers build RST trees whose leaves are EDUs for each sentence in a document and build an RST

[1]Of course, we can directly build discourse dependency trees by applying syntactic dependency parsing technologies. However, it is known that the performance is not so good (Hayashi, Hirao, and Nagata 2016).



(a) Original Tree (b) RST tree with two-levels of granularity (c) RST tree with three-levels of granularity

Figure 2: Original RST tree and RST trees with different number of granularity levels. p, s, and e denote paragraph, sentence, and EDU, respectively.

tree whose leaves are sentences for a document. They then obtain an RST tree by replacing sentences as leaves with the already constructed parse trees for the sentences. Thus, these methods disregard paragraphs that are likely to be about the same topic as another level of granularity in a document.

In this paper, we build an RST tree with the following procedure:

1. Build a document tree whose leaves correspond to paragraphs for a document.
2. Build paragraph trees whose leaves correspond to sentences for each paragraph.
3. Build sentence trees whose leaves correspond to EDUs for each sentence.
4. Replace all leaves of the document tree with the paragraph trees and all leaves of the paragraph trees with the sentence trees.

We give an example of RST trees with different number of granularity levels. Fig. 2 (a) shows an original RST tree, a document directly represented as an RST tree whose leaves are EDUs. Fig. 2 (b) shows an RST tree with two levels of granularity, a document represented as an RST tree whose leaves are sentences and a sentence represented as an RST tree whose leaves are EDUs. This representation has been widely utilized in conventional RST parsing models. Fig. 2 (c) shows an RST tree with three levels of granularity, a document represented as an RST tree whose leaves are paragraphs, a paragraph represented as an RST tree whose leaves are sentences, and a sentence represented as an RST tree whose leaves are EDUs. This is our representation for an RST tree of a document.

Since the number of leaves in paragraph and sentence trees is smaller than that in the original RST tree, and we do not need to build subtrees that cross paragraph or sentence boundaries, we can reduce the search space of the parsing with the multiple levels of granularity. Assuming the number of possible span splits as the size of the search space, our investigation shows that considering two-levels and three-levels of granularity makes the search space 7.6% and 0.8% respectively for RST-DT, compared with the case with a single granularity level. Due to some gold RST trees in RST-DT cross paragraph or sentence boundaries, the upper bound of unlabeled span score of a parser with two levels of granularity is .963 and that with three levels of granularity is .955 on the test data of RST-DT.

Representations for Text Spans

To parse a document in a top-down manner, we need to represent an atomic unit, either a paragraph, a sentence or an EDU, as a feature vector. Thus, we utilize BiLSTMs and the selective gate mechanism (Zhou et al. 2017), which are widely used in text summarization tasks to obtain the vector representaion. By using a forward-LSTM function $\overrightarrow{\text{LSTM}}$ and backward-LSTM function $\overleftarrow{\text{LSTM}}$, where the forward and backward hidden states of j-th word in the atomic unit is represented as follows:

$$\begin{aligned}\overrightarrow{\mathbf{h}}_j &= \overrightarrow{\text{LSTM}}_{\text{word}}(\overrightarrow{\mathbf{h}}_{j-1}, \mathbf{emb}_j),\\ \overleftarrow{\mathbf{h}}_j &= \overleftarrow{\text{LSTM}}_{\text{word}}(\overrightarrow{\mathbf{h}}_{j+1}, \mathbf{emb}_j),\\ \mathbf{h}_j &= [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j],\end{aligned} \quad (1)$$

where $\mathbf{emb}_j$ is the embedding of j-th word. To obtain the representation for $\mathbf{emb}_j$, we use a concatenation of three layers in ELMo (Peters et al. 2018) and a GloVe vector (Pennington, Socher, and Manning 2014) for j-th word, by following successful studies for EDU segmentation (Wang, Li, and Yang 2018) and semantic role labeling (Strubell et al. 2018; Ouchi, Shindo, and Matsumoto 2018).

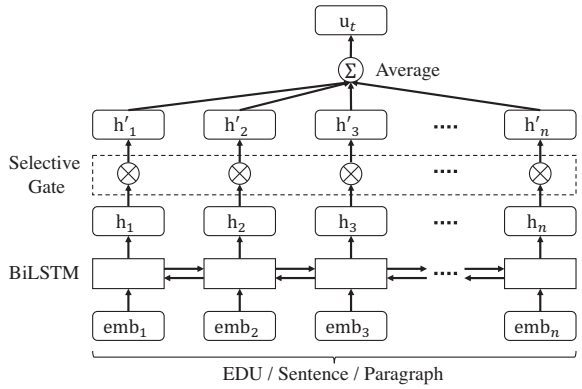


Figure 3: Overview of representation for the atomic unit.

We then introduce selective gates to control the importance of each word in the unit. The selective gates receive hidden state $\mathbf{h}_j$ and context vector $\mathbf{s} = [\mathbf{h}_n; \mathbf{h}_1]$ and compute new hidden state $\mathbf{h}'_j$ as follows:

$$\begin{aligned} \mathbf{sGate}_j &= \sigma(\mathbf{W}_s \mathbf{h}_j + \mathbf{U}_s \mathbf{s} + \mathbf{b}_s), \\ \mathbf{h}'_j &= \mathbf{h}_j \odot \mathbf{sGate}_j, \end{aligned} \tag{2}$$

where $\mathbf{W}_s, \mathbf{U}_s$ are weight matrices and $\mathbf{b}_s$ is a bias vector. σ is the sigmoid activation function, and $\odot$ is elementwise multiplication. Finally, we calculate $\mathbf{u}_t$ that is the vector representation of either a paragraph, sentence or an EDU, as follows:

$$\mathbf{u}_t = \frac{1}{n} \sum_{j \in \{1, \cdots, n\}} \mathbf{h}'_j. \tag{3}$$

Then, we represent a text span, which consists of either a paragraph, sentence, EDU or its sequence, based on BiLSTMs and the representation of the textual unit as follows:

$$\begin{aligned} \mathbf{f}_i &= \overrightarrow{\mathrm{LSTM}}_{\mathrm{unit}}(\mathbf{f}_{i-1}, \mathbf{u}_i), \\ \mathbf{b}_i &= \overleftarrow{\mathrm{LSTM}}_{\mathrm{unit}}(\mathbf{b}_{i+1}, \mathbf{u}_i), \end{aligned} \tag{4}$$

where $\mathbf{f}$ and $\mathbf{b}$ are hidden states of the forward and backward LSTMs, respectively. Fig. 3 shows the overview of our representation for the atomic unit.

Finally, we denote a span from the i-th unit to j-th unit as $\mathbf{u}_{i:j}$. The vector representation of $\mathbf{u}_{i:j}$ is defined as a concatenation of two vectors as follows:

$$\mathbf{u}_{i:j} = [\mathbf{f}_j - \mathbf{f}_{i-1}; \mathbf{b}_{i-1} - \mathbf{b}_j]. \tag{5}$$

To investigate which is effective between explicitly dividing the process into multiple levels in a document or implicitly representing them as features, we embed the encoding of the information obtained from the granularity levels, in cases where they are not explicitly utilized in the process division. For example, we embed the encoding of paragraph boundary information into the feature vectors when we divide the process into only one or two levels. That is, we concatenate the boundary information for a sentence and a paragraph to $\mathbf{u}_{i:j}$. We use the following four types of boundary information: whether the span is the start of a sentence, the start of a paragraph, crosses a sentence boundary, and crosses a paragraph boundary. We express 16 ($= 2^4$) possibilities, based on the above 4 types of information, with an embedding of 10 dimensions.

Parsing Model

Recently, top-down parsing has been successfully applied to syntactic parsing (Stern, Andreas, and Klein 2017; Shen et al. 2018). These methods build a constituent tree by recursively splitting a text span that consists of words. Since an RST tree can be regarded as a constituent tree, we follow the top-down parsing approach.

Splitting Span To split a span $\mathbf{u}_{i:j}$ that consists of either a paragraph, sentence or EDU at each position k, we define a deep biaffine scoring function (Dozat and Manning 2016) $s_{\mathrm{split}}(i, j, k)$ as follows:

$$s_{\mathrm{split}}(i, j, k) = \mathbf{h}_{i:k}^\top \mathbf{W}_u \mathbf{h}_{k+1:j} + \mathbf{v}_\ell^\top \mathbf{h}_{i:k} + \mathbf{v}_r^\top \mathbf{h}_{k+1:j}, \tag{6}$$

where $\mathbf{W}_u$ is a weight matrix, $\mathbf{v}_\ell$ and $\mathbf{v}_r$ are weight vectors corresponding to the left and right spans, respectively. The $\mathbf{h}_{i:k}$ and $\mathbf{h}_{k+1:j}$ are defined as follows:

$$\mathbf{h}_{i:k} = \mathrm{MLP}_{\mathrm{left}}(\mathbf{u}_{i:k}), \tag{7}$$

$$\mathbf{h}_{k+1:j} = \mathrm{MLP}_{\mathrm{right}}(\mathbf{u}_{k+1:j}), \tag{8}$$

where MLP_* is the multi-layer perceptron. We use a single feedforward network and the ReLU function as the activation function and learn parameters for the left and right span vectors.

We split a span with position k that maximizes Eq. (6):

$$\hat{k} = \underset{k \in \{i, \ldots, j-1\}}{\arg\max} \, [s_{\mathrm{split}}(i, j, k)]. \tag{9}$$

Labeling Spans To determine the nuclearity and rhetorical relation labels of two adjacent spans that are children of $u_{i:j}$, we define the following same scoring function $s_{\mathrm{label}}(i, j, k, \ell)$ for splitting position k for both nuclearity and relation labels:

$$s_{\mathrm{label}}(i, j, k, \ell) = \mathbf{W}_\ell \mathrm{MLP}([\mathbf{u}_{i:k}; \mathbf{u}_{k+1:j}; \mathbf{u}_{1:i}; \mathbf{u}_{j:n}]), \tag{10}$$

where $\mathbf{W}_\ell$ is the projection layer for the nuclearity or rhetorical relation labels. $\mathbf{u}_{1:i}; \mathbf{u}_{j:n}$ are left and right spans that appear outside the current focus.

We select the label for the spans that maximizes Eq. (10):

$$\hat{\ell} = \underset{\ell \in \mathcal{L}}{\mathrm{argmax}} [s_{\mathrm{label}}(i, j, k, \ell)], \tag{11}$$

where $\mathcal{L}$ denotes a set of valid nuclearity label combinations, {N-S,S-N,N-N}, in predicting the nuclearity, and a set of rhetorical relation labels, {Elaboration, Condition,. . .},[2] in predicting the rhetorical relation. Note that we separately learn the weight parameters $\mathbf{W}_\ell$ and MLP for nuclearity labeling and rhetorical relation labeling.

[2]The number of rhetorical relations is 18.

Algorithm 1 Top-down parsing

1: SPLITSPAN($u^p_{1:n}$)
2: **for** $m = 1$ to n **do**
3: $b_m \leftarrow$ start(u^p_m), $e_m \leftarrow$ end(u^p_m)
4: SPLITSPAN($u^s_{b_m:e_m}$)
5: **for** $t = b_m$ to e_m **do**
6: $b_t \leftarrow$ start(u^s_t), $e_t \leftarrow$ end(u^s_t)
7: SPLITSPAN($u^e_{b_t:e_t}$)
8: **procedure** SPLITSPAN(SPAN)
9: $i \leftarrow$ start(SPAN),$j \leftarrow$ end(SPAN)
10: **if** $j - i > 0$ **then**
11: $\hat{k} \leftarrow \underset{k \in \{i,\ldots,j-1\}}{\arg\max} [s_{split}(i, j, k)]$
12: $\hat{\ell} \leftarrow \underset{\ell \in \mathcal{L}}{\arg\max} [s_{\text{label}}(i, j, \hat{k}, \ell)]$
13: Left_child(SPAN) $\leftarrow u_{i:\hat{k}}$
14: Right_child(SPAN) $\leftarrow u_{\hat{k}+1:j}$
15: Left_label($u_{i:\hat{k}}$) $\leftarrow \hat{\ell}_{\text{left}}$
16: Right_label($u_{\hat{k}+1:j}$) $\leftarrow \hat{\ell}_{\text{right}}$
17: SPLITSPAN($u_{i:\hat{k}}$)
18: SPLITSPAN($u_{\hat{k}+1:j}$)

Learning To optimize the parameters of our method, we employ margin-based learning. When the correct splitting position k^* and label ℓ^* are given for a span $u_{i:j}$, losses for splitting, nuclearity and rhetorical-relation labeling are defined as follows:

$$\max(0, 1 + s_{\text{split}}(i, j, k^*) - s_{\text{split}}(i, j, \hat{k})), \quad (12)$$

$$\max(0, 1 + s_{\text{label}}(i, j, \hat{k}, \ell^*) - s_{\text{label}}(i, j, \hat{k}, \hat{\ell})). \quad (13)$$

We can obtain optimal parameters by minimizing the sum of the losses in each splitting point, using the optimization algorithm based on the gradient method.

Parsing Algorithm Algorithm 1 shows a top-down parsing algorithm. A superscript for u indicates what it consists of, either a paragraph, sentence or EDU. $u^p_{1:n}$ denotes a text span consisting of n paragraphs, $u^s_{b_m:e_m}$ denotes a text span consisting of sentences in the m-th paragraph, and $u^e_{b_t:e_t}$ denotes a text span consisting of EDUs in the t-th sentence. Functions start() and end() return the start and end indexes of the unit in the input span, respectively. For example, start(u^p_i) returns the index of the first sentence in the i-th paragraph and end(u^p_i) returns the index of the last sentence. Left_child(SPAN) and Right_child(SPAN) denote the left and right children of the span, respectively, and Left_label() and Right_label() denote the labels of the child spans, respectively.

First, the algorithm builds a document tree whose leaves are paragraphs (line 1). Second, it builds paragraph trees whose leaves are sentences for each paragraph (lines 2-4). Third, it builds sentence trees whose leaves are EDUs for each sentence (lines 5-7).

Finally, we obtain an RST tree for the whole document by connecting all the trees, *i.e.*, we replace leaves of the document tree with the paragraph trees and then replace the leaves of the paragraph trees with the sentence trees.

Fig. 4 shows an example of building a subtree of the RST tree corresponding to span $e_{3:5}$. To compute the splitting score (Eq.(6)) for each candidate point, $k = 3, 4$, we obtain span representations $\mathbf{u}_{3:3}$, $\mathbf{u}_{4:5}$ and $\mathbf{u}_{3:4}$, $\mathbf{u}_{5:5}$ by exploiting BiLSTMs. Then, we split the span with k which maximizes the split score. In this example, we split the span $e_{3:5}$ into left span($e_{3:3}$) and right span($e_{4:5}$). After the splitting, we assign nuclearity and relation labels for the spans by computing labeling scores (Eq. (10)). In the example, nucleus is assigned to $e_{3:3}$, satellite is assigned to $e_{4:5}$, and "Elaboration" is assigned as the relation label between the two spans.

Experiments

Settings

Dataset We evaluated our method by using the standard benchmark dataset RST-DT (Carlson, Marcu, and Okurowski 2001). RST-DT was officially divided into 347 documents as the training dataset and 38 documents as the test dataset, which indicates that there is no development dataset available. Thus, we used 40 documents in the training dataset as the development data by following the study by Heilman and Sagae (2015). We conducted all experiments on gold EDU segmentation by following the previous studies.

Model parameters The dimension size of the hidden layers was set to 250, dropout layers were incorporated with the ratio 0.4 at the training step, and the maximum training epoch was set to 50 for each model. All weight parameters were updated using Adam (Kingma and Ba 2014) with an initial learning rate being 0.001. The learning rate was decayed for each epoch with the ratio of 0.99. The gradient-clipping threshold was set to 5.0 and the weight-decay was set to $1e^{-4}$. In testing, trained parameters with the highest fully-labeled span score on the development data were used for each model.

Evaluation Measures To evaluate the performance of our parser, we used micro-averaged F_1 scores of unlabeled spans, those of nuclearity labeled spans, those of rhetorical relation labeled spans based on RST-Parseval (Marcu 2000). Moreover, to evaluate discourse dependency trees converted from RST trees, we used unlabeled attachment score (UAS) and labeled attachment score (LAS).

Compared Methods To demonstrate that our parser is practical, we compared it with the following high-performance parsers described in a survey paper (Morey, Muller, and Asher 2017), and two state-of-the-art parsers :
HILDA (duVerle and Prendinger 2009) is a reimplementation (Hayashi, Hirao, and Nagata 2016) of the classical bottom-up greedy parser with SVMs. The parser does not exploit different levels of granularity in a document.
FH14gCRF (Feng and Hirst 2014) is the previously best traditional statistical model, a bottom-up greedy parser with linear-chain CRF models that exploits two levels of granularity in a document.
JE14DPLP (Ji and Eisenstein 2014) is a shift-reduce parser

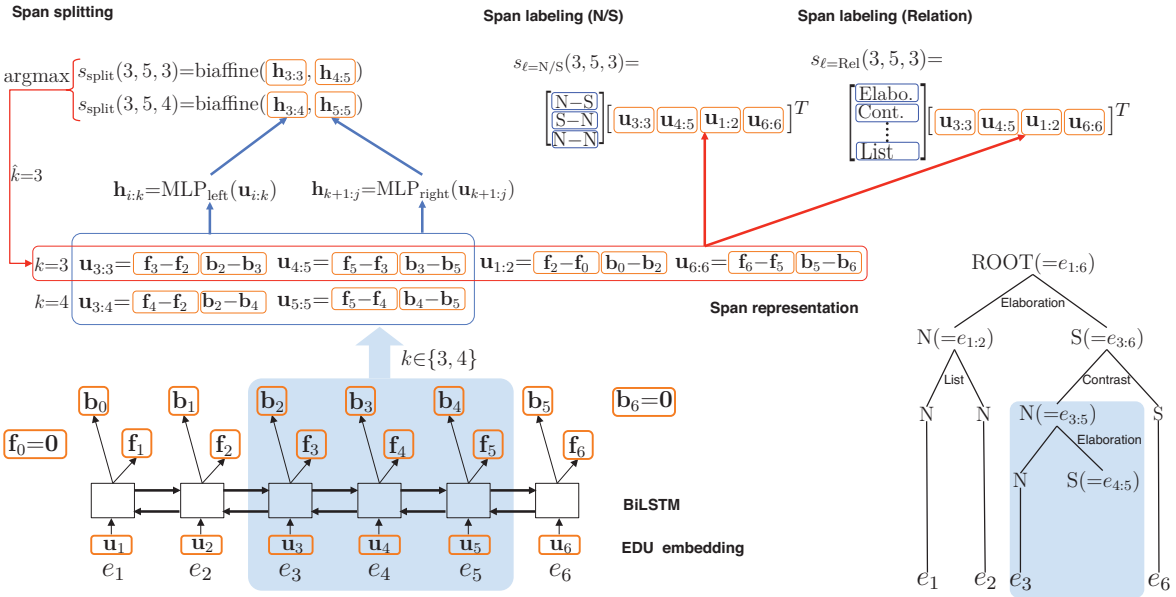


Figure 4: The process of building an RST tree.

with SVM that is trained by features obtained from representation learning. The parser does not exploit different levels of granularity in a document.

LLC16 (Li, Li, and Chang 2016) is a neural network-based CKY parser based on hierarchical BiLSTMs. The first layer is used to encode EDUs as sequences of words, and the second layer is used to encode a document as a sequence of EDUs. It also does not exploit different levels of granularity in a document.

WLW17 (Wang, Li, and Wang 2017) is the currently best method based on the traditional statistical model with SVM. They predict relation labels in three levels of granularity, but the structure was predicted all at once from EDUs.

YZF18 (Yu, Zhang, and Fu 2018) is a transition-based neural parser that uses vectors from a pretrained dependency parser as features. This parser does not exploit different levels of granularity. However, it achieved the state-of-the-art on relation labeled span F_1 scores.

To demonstrate the effectiveness of considering different levels of granularity in a document, we evaluated our top-down parser with the following settings:

D2E for directly building an RST tree whose leaves are EDUs for a document. The information of sentence boundaries and paragraph boundaries is embedded as features.

D2S2E for exploiting two levels of granularity in a document. The information of paragraph boundaries is embedded as features. **D2P2S2E** for exploiting three levels of granularity in a document.

Results and Discussion

Table 1 shows the results in terms of micro-averaged unlabeled span, nuclearity labeled span and relation labeled span F_1 scores. The scores corresponding to HILDA, FH14gCRF, JE14DPLP and LLC16 were obtained from a survey paper

Model	Span	Nuc	Rel
D2E	86.1	73.1	58.9
D2S2E	86.4	73.4	59.4
D2P2S2E	**87.0**	**74.6**	60.0
WLW17	86.0	72.4	59.6
YZF18	85.5	73.1	**60.2**
YZF18*	85.9	72.5	59.4
HILDA	82.6	66.6	54.6
FH14gCRF	84.3	69.4	56.9
JE14 DPLP	82.0	68.2	57.8
LLC16	82.2	66.5	51.4
Human	88.3	77.3	65.4

Table 1: Micro averaged F_1 scores based on RST-Parseval. Span denotes unlabeled span F_1 scores, Nuc denotes nuclearity labeled span F_1 scores, and Rel denotes relation labeled span F_1 scores.

(Morey, Muller, and Asher 2017) and those corresponding to YZF18 were obtained from their publication. The scores of YZF18* and WLW17 were obtained from the results by running their codes. The scores of WLW17 agreed with those reported in their publication. However, the scores of YZF18* were slightly different from YZF18. The scores of all of our models were obtained from 5 model ensemble with different random seeds.

From the results, as the number of granularity levels for dividing a document increased, the performance was improved. D2P2S2E that exploits three levels of granularity in a document obtained the best results. These results imply that pre-defined granularity levels of a document, such

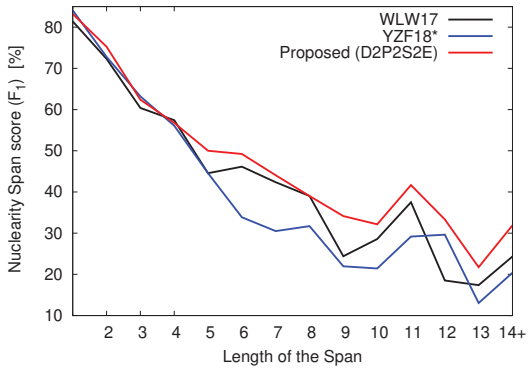


Figure 5: The relation between nuclearity span score and the length of a span

as sentences and paragraphs, are important features in RST parsing, and our division by three levels of granularity in a document is more suitable for RST parsing than previous division by two levels of granularity.

Then, our methods outperformed both neural and statistical previous methods in most cases. In particular, D2P2S2E achieved remarkable scores: the best Span (87.0), Nuc (74.6) and the second best Rel (60.0).

To clearly show the differences between our method D2P2S2E and previous methods, WLW17 and YZF18*, we performed significance tests, using paired bootstrap resampling (Koehn 2004) with Bonferroni correction at significance level=0.05. The results showed that there were significant differences between our method and the previous methods for Span and Nuc, while there were no significant differences for Rel. That is, our method completely outperformed WLW17 and YZF18* in Span and Nuc and obtained comparable performance in Rel.

To reveal the effectiveness of our top-down parsing in details, we investigated the relation between nuclearity labeled span score and the length of a text span, and compared it with those for WLW17 and YZF18*. Fig. 5 shows the results. Our method outperformed both WLW17 and TZF18* in most span lengths. In particular, as the spans became longer, the performance difference between our method and both WLW17 and YZF18* became larger. Thus, as we expected, our parser tends to build more accurate structures at the upper parts of RST trees than WLW17 and YZF18*. The tendency improves the Span and Nuc scores.

Since the improvements of upper parts of RST tree structures should derive better discourse dependency trees, we then evaluated discourse dependency trees converted from RST trees built by our methods, WLW17, YZF18* and HILDA, and those obtained from a simple MST parser. RST trees were converted into discourse dependency trees by applying transformation rules proposed by (Hirao et al. 2013), and the discourse dependency trees were evaluated by unlabeled attachment score (UAS) and labeled attachment score (LAS), that are widely utilized in syntactic dependency parsing researches.

Table 2 shows the results. Scores corresponding to

Model	UAS	LAS
D2E	63.9	45.1
D2S2E	64.0	46.3
D2P2S2E	**64.9**	**48.5**
WLW17	61.5	47.8
YZF18*	61.9	48.4
HILDA	57.1	46.2
MST	55.0	43.1

Table 2: Evaluation results for discourse dependency trees.

HILDA and MST were obtained from the paper (Hayashi, Hirao, and Nagata 2016). As we mentioned before, the performances of the MST parser that directly predicts discourse dependency trees were worse than those obtained from RST parsers. Our methods outperformed the previous methods on UAS, and D2P2S2E achieved the state-of-the-art results on both UAS and LAS, as we expected. Compared to the results in Table 1, the differences between D2P2S2E and the previous methods are more remarkable. To test whether the differences between D2P2S2E and the previous methods are significant, we performed significance tests using the same method as before. The results showed that there were significant differences between D2P2S2E and WLW17, YZF18* on UAS while there were no significant differences on LAS. Thus, the results demonstrate the effectiveness of the improvements of upper parts of RST tree structures. However, the results also show that we still have room for further improvement in relation labeling of our method.

Conclusion

To obtain better discourse dependency trees, we proposed a neural top-down RST parsing method exploit three levels of granularity in a document. By employing top-down parsing, we can improve the accuracy of RST trees at the upper parts of the structures that are required to obtain good discourse dependency trees. To the best of our knowledge, this was the first study on top-down document-level RST parsing that explored the multiple levels of granularity in a document. Experimental results on the RST-DT corpus showed that our method achieved the state-of-the-art unlabeled span and nuclearity labeled span scores, and obtained comparable relation labeled span score with the state-of-the-art. Discourse dependency trees converted from the RST trees obtained the state-of-the-art UAS and LAS scores. These results demonstrated the effectiveness of top-down parsing and introducing multiple levels of granularity in a document.

References

Bhatia, P.; Ji, Y.; and Eisenstein, J. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2212–2218.

Carlson, L.; Marcu, D.; and Okurowski, M. E. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Dozat, T., and Manning, C. D. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734.

duVerle, D., and Prendinger, H. 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*, 665–673.

Feng, V. W., and Hirst, G. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 60–68.

Feng, V. W., and Hirst, G. 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*, 511–521.

Hayashi, K.; Hirao, T.; and Nagata, M. 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*, 128–136.

Heilman, M., and Sagae, K. 2015. Fast rhetorical structure theory discourse parsing. *CoRR* abs/1505.02425.

Hirao, T.; Yoshida, Y.; Nishino, M.; Yasuda, N.; and Nagata, M. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1515–1520.

Hong, J., and Huang, L. 2018. Linear-time constituency parsing with RNNs and dynamic programming. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 477–483.

Jansen, P.; Surdeanu, M.; and Clark, P. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 977–986.

Ji, Y., and Eisenstein, J. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 13–24.

Ji, Y., and Smith, N. A. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 996–1005.

Joty, S.; Carenini, G.; Ng, R.; and Mehdad, Y. 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*, 486–496.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Koehn, P., and Knowles, R. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, 28–39.

Koehn, P. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 388–395.

Li, Q.; Li, T.; and Chang, B. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 362–371.

Li, J.; Li, R.; and Hovy, E. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2061–2069.

Lin, X.; Joty, S. R.; Jwalapuram, P.; and Bari, M. S. 2019. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4190–4200.

Liu, Y., and Lapata, M. 2017. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1289–1298.

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

Marcu, D. 1998. Improving summarization through rhetorical parsing tuning. In *Processing of the Sixth Workshop on Very Large Corpora*, 206–215.

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

Morey, M.; Muller, P.; and Asher, N. 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*, 1319–1324.

Ouchi, H.; Shindo, H.; and Matsumoto, Y. 2018. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1630–1642.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2227–2237.

Shen, Y.; Lin, Z.; Jacob, A. P.; Sordoni, A.; Courville, A.; and Bengio, Y. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 1171–1180.

Stern, M.; Andreas, J.; and Klein, D. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 818–827.

Strubell, E.; Verga, P.; Andor, D.; Weiss, D.; and McCallum, A. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 5027–5038.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, 2692–2700.

Wang, Y.; Li, S.; and Wang, H. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 184–188.

Wang, Y.; Li, S.; and Yang, J. 2018. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 962–967.

Yu, N.; Zhang, M.; and Fu, G. 2018. Transition-based neural rst parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, 559–570.

Zhou, Q.; Yang, N.; Wei, F.; and Zhou, M. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1095–1104.