Split or Merge: Which is Better for Unsupervised RST Parsing?

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

Abstract

Rhetorical Structure Theory (RST) parsing is crucial for many downstream NLP tasks that require a discourse structure for a text. Most of the previous RST parsers have been based on supervised learning approaches. That is, they require an annotated corpus of sufficient size and quality, and heavily rely on the language and domain dependent corpus. In this paper, we present two language-independent unsupervised RST parsing methods based on dynamic programming. The first one builds the optimal tree in terms of a dissimilarity score function that is defined for splitting a text span into smaller ones. The second builds the optimal tree in terms of a similarity score function that is defined for merging two adjacent spans into a large one. Experimental results on English and German RST treebanks showed that our parser based on span merging achieved the best score, around 0.8 F_1 score, which is close to the scores of the previous supervised parsers.

1 Introduction

Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is one of the theories that are most widely utilized for representing a discourse structure of a text in downstream NLP applications, such as automatic summarization (Marcu, 1998; Hirao et al., 2013), sentiment analysis (Bhatia et al., 2015), and text categorization (Ji and Smith, 2017). RST represents a text as a kind of constituent tree, whose leaves are Elementary Discourse Units (EDUs), clause-like units, and whose non-terminal nodes cover text spans consisting of a sequence of EDUs or a singleton EDU. The label of a non-terminal node represents the attribution of a text span, *nucleus* or *satellite*. A discourse relation is also assigned between two adjacent non-terminal nodes.

Since the RST tree can be regarded as a standard constituent (phrase structure) tree, syntactic parsing models for constituency parsing can also be successfully applied to RST parsing. In most cases, RST parsers have been developed on the basis of supervised learning algorithms, which require a high quality annotated corpus of sufficient size. 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). These supervised RST parsing methods might not be applied to languages with only a small-size corpus.

This paper presents two types of language independent unsupervised RST parsing methods based on the CKY-like dynamic programming-based approach. One method builds the most likely parse tree in terms of a dissimilarity score defined for splitting a text span into two smaller ones. The other builds the optimal tree in terms of a similarity score defined for merging two adjacent text spans into a larger one. The similarity and dissimilarity scores between text spans are calculated on the basis of their distributional representations. Note that since our method is fully unsupervised, the parser predicts only the skeleton of an RST tree.

Moreover, we exploit multiple granularity levels of a document: (1) we first independently build three types of trees, a tree whose leaves correspond to a paragraph, a tree whose leaves correspond to a sentence, and a tree whose leaves correspond to an EDU, and then (2) merge them to obtain the whole RST tree.

We conducted experimental evaluation on English and German datasets, RST-DT and the Potsdam Commentary Corpus (PCC) (Stede and Neumann, 2014), respectively. The results demonstrated that our method with span merging outperformed our method with span splitting and obtained .811 and .784 span scores for RST-DT and PCC, respectively. The scores are close to the scores of early supervised RST parsers.

2 Unsupervised RST Parsing

2.1 Motivation

Generally, RST trees with multinulcear relations are transformed into binary trees by applying right-heavy binarization as in syntactic parsing. As the result, we can easily imagine the following two simple parsing methods. First, when a split score for splitting a text span into two smaller spans is given for each candidate splitting point in the text span, we can build an RST tree by recursively splitting a text span with the best candidate points. Second, when a merge score for merging two adjacent spans into a larger one is given for each candidate pair of the spans, we can build an RST tree by recursively merging the best candidate pairs. Here, we can regard dissimilarity between two text spans as the split score and similarity between them as the merge score. Note that the dissimilarity score can be defined as $1 -$ similarity score. We propose the similarity score that is based on distributional representations of text spans in Section 2.2.

In general, while a parse tree can be obtained either by splitting or merging with a greedy parsing method, as mentioned above, it cannot be always optimal in terms of the total split or merge score. Thus, we propose our dynamic programming-based approach to obtain the optimal tree in Section 2.3.

Of course, it would be possible to utilize both similarity and dissimilarity scores. However, in this paper, we investigate which one is suitable for unsupervised RST parsing as the first step.

2.2 Similarity and Dissimilarity Scores between Text Spans

When two adjacent text spans, the left span ℓ from i-th to k-th atomic text unit[1] and the right span r from $k{+}1$ to j-th atomic text unit are given, we define the similarity score between them as follows:

$$\mathrm{sim}(\overrightarrow{\ell_{i:k}}, \overrightarrow{r_{k+1:j}}) = \frac{1}{2}\left\{ \frac{\overrightarrow{\ell_{i:k}} \cdot \overrightarrow{r_{k+1:j}}}{\|\overrightarrow{\ell_{i:k}}\| \|\overrightarrow{r_{k+1:j}}\|} + 1 \right\}. \tag{1}$$

[1]A paragraph, a sentence or an Elementary Discourse Unit (EDU).

Here, $\overrightarrow{\ell_{i:k}}$ and $\overrightarrow{r_{k+1:j}}$ indicate the vector representations of the left and right spans, which are defined as a concatenation of two vectors for the left most atomic unit and the right most atomic unit as follows:

$$\begin{aligned} \overrightarrow{\ell_{i:k}} &= [\overrightarrow{u_i}; \overrightarrow{u_k}], \\ \overrightarrow{r_{k+1:j}} &= [\overrightarrow{u_{k+1}}; \overrightarrow{u_j}]. \end{aligned} \tag{2}$$

The definition is inspired by that of (Ji and Eisenstein, 2014), which employs words at the beginning and end of a text span as important features to build an RST tree.

$\overrightarrow{u_t}$ is the vector representation of a t-th atomic unit u_t, which is defined on the basis of SIF (Arora et al., 2017)[2] as follows:

$$\overrightarrow{u_t} = \sum_{w \in W_t} \frac{a}{p(w) + a} \overrightarrow{w}. \tag{3}$$

$p(w)$ is the occurrence probability for word w, $\overrightarrow{w}$ is the vector representation of the word and W_t is a set of words in u_t. We use a concatenation of two word vectors obtained from ELMo (Peters et al., 2018) and Glove (Pennington et al., 2014) as a vector representation of a word. a is a parameter to decay the score of frequent words and is defined as $a = (1-\alpha)/(\alpha Z)$, where α is a hyper parameter and Z is the total number of words.

2.3 Dynamic Programming-Based Approach for Building Optimal Trees

We propose a dynamic programming-based approach to obtain the optimal tree in terms of either the total split or merge score from all the possible trees.

We first illustrate the algorithm with a merge (similarity) score. We define $V[b][e]$, which stores the maximum merge score for a span $u_{b:e}$ consisting from b-th unit to e-th unit, as follows:

$$V[b][e] = \begin{cases} 1, & b = e \\ \max\limits_{b \le k < e} V[b][k] \times \mathrm{sim}(\overrightarrow{\ell_{b:k}}, \overrightarrow{r_{k+1,e}}) \times V[k+1][e], & \text{otherwise} \end{cases} \tag{4}$$

where $V[b][k]$ stores the maximum merge score for a sub-span $u_{b:k}$, and $V[k{+}1][e]$ stores the maximum merge score for a sub-span $u_{k+1:e}$.

[2] As for the representation for sentences and paragraphs, (Arora et al., 2017) compared SIF with the following three: avg-GloVe, tfidf-GloVe, skip-thought vectors (Kiros et al., 2015).

Algorithm 1 Dynamic Programming-Based RST Parsing

```
 1: $V^p[1][d] \leftarrow$DP($u^p_{1:d}$)
 2: BACKTRACE($V^p[1][d]$)
 3: for $i = 1$ to $m$ do
 4:     $b_i \leftarrow$ start($u^p_i$), $e_i \leftarrow$ end($u^p_i$)
 5:     $V^s_i[b_i][e_i] \leftarrow$DP($u^s_{b_i:e_i}$)
 6:     BACKTRACE($V^s_i[b_i][e_i]$)
 7:     for $j = b_s$ to $e_s$ do
 8:         $b_j \leftarrow$ start($u^s_j$), $e_j \leftarrow$ end($u^s_j$)
 9:         $V^e_j[b_j][e_j] \leftarrow$DP($u^e_{b_j:e_j}$)
10:         BACKTRACE($V^e_j[b_j][e_j]$)
11: function DP(SPAN)
12:     $V \leftarrow \phi$
13:     $B \leftarrow$ start(SPAN),$E \leftarrow$ end(SPAN)
14:     for $i = 0$ to $E - B$ do
15:         for $j = 0$ to $E - B - i$ do
16:             $b \leftarrow B + j, e \leftarrow B + j + i$
17:             if $b = e$ then
18:                 $V[b][e] = 1$
19:             else
20:                 $V[b][e] = \max_{b \leq k < e} V[b][k] \times$
21:                 $\mathrm{sim}(\overrightarrow{\ell_{b:k}}, \overrightarrow{r_{k+1,e}}) \times V[k+1][e]$
    return $V[B][E]$
22: procedure BACKTRACE($V[b][e]$)
23:     $\hat{k} = \operatorname{arg\,max}_{k \in \{b,\ldots,e-1\}} V[b][k] \times \mathrm{sim}(\overrightarrow{\ell_{b:k}}, \overrightarrow{r_{k+1,e}}) \times$
24:                 $V[k+1][e]$
25:     $C_\ell(u_{b:e}) \leftarrow u_{b:\hat{k}}$
26:     $C_r(u_{b:e}) \leftarrow u_{\hat{k}+1:e}$
27:     BACKTRACE($V[b][\hat{k}]$)
28:     BACKTRACE($V[\hat{k}+1][e]$)
```

Algorithm 1 shows a dynamic programming-based RST parsing algorithm to build the optimal RST tree in terms of the total merge score. In the algorithm, DP is a function that fills out the triangular matrix V as in the CKY algorithm, and BACKTRACE is a procedure to build a tree by recursively determining the left and right children for a span. Figure 1 shows an example of table V. The parse tree is obtained by recursively tracing back the path with the maximum score from $V[1][6]$. For example, the left child of span $u_{1:6}$ is span $u_{1:4}$ and the right child is span $u_{5:6}$.

Furthermore, we regard a document as a text span consisting of three different granularity levels: (1) a span from 1st to d-th paragraphs, assuming that the document consists of d paragraphs, denoted as $u^p_{1:d}$, (2) a span from b_i-th to e_i-th sentences for the i-th paragraph u^p_i, which we denote as $u^s_{b_i:e_i}$, and (3) a span from b_j-th to e_j-th EDUs for the j-th sentence u^s_j, which we denote as $u^e_{b_j:e_j}$.

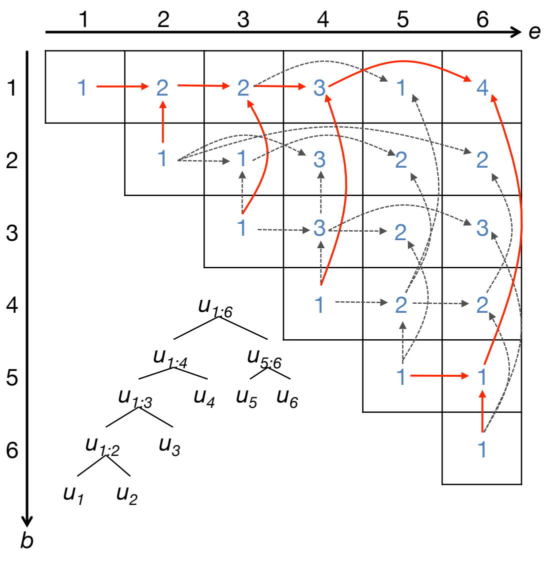

Figure 1: DP table and the RST tree-building.

	RST-DT	PCC
# of Docs.	385	174
# of EDUs/Doc.	56.6	16.1
# of Sents./Doc.	22.5	11.4
# of Para./Doc.	9.99	2.10
# of EDUs/Sent.	2.51	1.41
# of EDUs/Para.	5.67	7.66

Table 1: Statistics of the datasets.

First, the algorithm builds a document tree (lines 1-2). Then, it builds paragraph trees for each paragraph (lines 3-6) and builds sentence trees for each sentence (lines 7-10). Finally, we obtain an RST tree for the whole document by connecting the trees.

By replacing the similarity score, sim(,) with the dissimilarity score, $1 - \mathrm{sim}(,)$, we can obtain the optimal tree in terms of the split score. When we use the split score, the span consisting of only two atomic units can be divided into two uniquely, without calculating the split score for the span. Therefore, we compute $V[b][e] = 1$ when $e - b < 2$, instead of when $b = e$, in Equation (4). In addition, Algorithm 1, from lines 17 to 18, must be modified.

3 Experiments

3.1 Experimental Setting

We evaluated our methods on two datasets, the English RST treebank, RST-DT and the German RST treebank, PCC. RST-DT has 347 training docu-

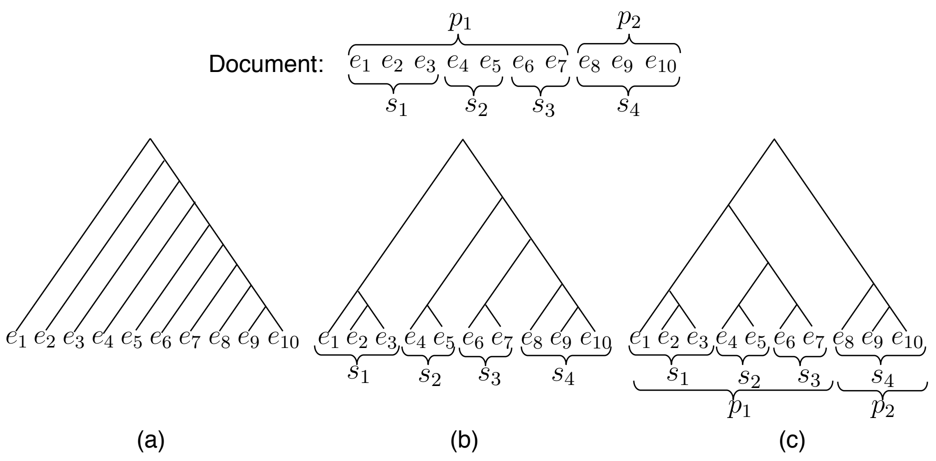

Figure 2: Right branching-trees of a document consists of ten EDUs, four sentences and two paragraphs. A right branching-tree with D2E setting (a), that with D2S2E setting (b) and that with D2P2S2E setting.

ments and 38 test documents. PCC[3] is smaller than RST-DT, and the number of documents is 174. We used the whole dataset as the test set on both datasets. Table 1 shows the statistics of the datasets. The numbers of paragraphs, sentences and EDUs per document for PCC are quite smaller than those for RST-DT.

We evaluated three variants of our methods, D2P2S2E, which is described in Algorithm 1, D2S2E, which employs two different granularity levels, *i.e.*, it regards a document as a span consisting of sentences and a sentence as a span consisting of EDUs, and D2E, which employs a single granularity level, *i.e.*, it regards a document as a span consisting of EDUs. To the best of our knowledge, there are no recent researches on unsupervised RST parsing, while we could find many on supervised RST parsing. As the result, we employed right-branching trees by merging the right most two spans, at each granularity level as baselines and compare those with ours. Figure 2 shows right branching trees at each granularity level. From the figure, we can see that the right-branching tree with two or three granularity, (b) and (c), have more complex structure than that with single granularity (a).

Since our unsupervised RST parsers can only construct unlabeled RST parse trees, we used only micro F_1 span score (Marcu, 2000) as an evaluation metric. Following previous studies, we conducted all the experiments on manually segmented EDUs.

[3]Since PCC contains two invalid RST trees, maz-12666 and maz-8838, we excluded them.

Dataset	Gran.	Split	Merge	RB
RST-DT	D2E	.602	**.656**	.545
	D2S2E	.755	**.788**	.751
	D2P2S2E	.793	**.811**	.803
PCC	D2E	.656	**.669**	.626
	D2S2E	.757	**.760**	.749
	D2P2S2E	.787	.784	**.789**

Table 2: Micro Span F_1 scores for RST-DT and PCC.

3.2 Results

Table 2 shows the evaluation results. Merge outperformed Split in most cases, and D2P2S2E achieved the best scores among our variants on both RST-DT and PCC. To clearly show the differences between our proposed method and RB, we performed significance tests, using paired bootstrap resampling (Koehn, 2004) at significance level=0.05. The results showed that there were significant differences between our method and RB at all the settings (D2E, D2S2E, D2P2S2E) for English, and at D2E and D2S2E for German, while there were no significant differences at D2P2S2E for German. The results imply that merging two adjacent spans and dividing with three granularity levels is suitable for unsupervised RST parsing. Comparing our methods with the baseline, right-branching (RB), we could find larger differences without considering the granularity levels of a document, whereas the differences became smaller by considering three granularity levels. In particular, right-branching

Method	Span F_1 score
Merge	.808
HHN16	.826
WLW17	.856

Table 3: Micro Span F_1 scores on the test set of RST-DT.

was slightly better than our method on PCC. We think the result was related to the right-heavy binary conversion of RST trees: RST trees with multi-nuclear relations were transformed into binary trees beforehand by applying the right-heavy branching procedure. As a result, RB obtains better scores when the number of documents whose RST trees contain the multi-nuclear relations is larger. In fact, the ratio of the documents whose RST trees contain multi-nuclear relations in PCC is 0.712, while that in RST-DT is 0.464. We can obtain the information of sentences boundaries in most cases, however, sometimes we cannot obtain the information of paragraph boundaries. Thus, it is significant that our method outperformed baselines on D2S2E settings.

Moreover, we compared our method with some supervised RST parsers on the test set of RST-DT. Table 3 shows the results. In the table, HHN16 denotes a simple transition-based RST parser (Hayashi et al., 2016), and WLW17 denotes a current state-of-the-art transition-based RST parser (Wang et al., 2017). Although the score of our method is lower than the scores of the supervised parsers, the score is close to that of HHN16. The results demonstrated the effectiveness of our unsupervised RST parsing method. For reference, (Braud et al., 2017) reported that their supervised RST parser obtained .802 span F_1 score on PCC. However, we cannot compare the score with our score because the test set differs from ours.

4 Conclusion

This paper proposed two kinds of unsupervised RST parsing methods based on dynamic programming that can build the optimal RST tree in terms of either a span splitting score or a span merging score. We regarded a document as a text span consisting of three different granularity levels and built trees at each level, a document tree, paragraph trees for each paragraph, and sentence trees for each sentence. Then, we obtained an RST tree by connecting all trees together. To the best of our knowledge, this is the first study on unsupervised RST parsing. The evaluation results on RST-DT and PCC showed the effectiveness of our proposal; the dynamic programming-based approach with the span merging score, exploiting three granularity levels in a document, achieved .811 and .784 span F_1 scores on RST-DT and PCC, respectively. The results are close to the traditional transition-based supervised parser.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the 5th International Conference on Learning Representations*.

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

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*, pages 292–304.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

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

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520.

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*, pages 13–24.

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

Jamie Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

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

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

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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*, pages 2227–2237.

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*, pages 925–929.

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