Empirical comparison of dependency conversions for RST discourse trees

†**Katsuhiko Hayashi, Tsutomu Hirao** and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
†hayashi.katsuhiko@lab.ntt.co.jp

Abstract

Two heuristic rules that transform Rhetorical Structure Theory discourse trees into discourse dependency trees (DDTs) have recently been proposed (Hirao et al., 2013; Li et al., 2014), but these rules derive significantly different DDTs because their conversion schemes on *multinuclear* relations are not identical. This paper reveals the difference among DDT formats with respect to the following questions: (1) How complex are the formats from a dependency graph theoretic point of view? (2) Which formats are analyzed more accurately by automatic parsers? (3) Which are more suitable for text summarization task? Experimental results showed that Hirao's conversion rule produces DDTs that are more useful for text summarization, even though it derives more complex dependency structures.

1 Introduction

Recent years have seen an increase in the use of dependency representations throughout various NLP applications. For the discourse analysis of texts, dependency graph representations have also been studied by many researchers (Prasad et al., 2008; Muller et al., 2012; Hirao et al., 2013; Li et al., 2014). In particular, Hirao et al. (2013) proposed a current state-of-the-art text summarization method based on trimming discourse dependency trees (DDTs). Dependency tree representation is the key to the formulation of the tree trimming method (Filippova and Strube, 2008), and dependency-based discourse syntax has further potential to improve the modeling of a wide range of text-based applications.

However, no large-scale corpus exists that is annotated with DDTs since it is expensive to manually construct such a corpus from scratch. Therefore, Hirao et al. (2013) and Li et al. (2014) proposed heuristic rules that automatically transform RST discourse trees (RST-DTs)[1] into DDTs. However, even researchers, who cited these two works in their papers, have ignored their differences, probably because the authors described only abstracts of their conversion methods. To clarify their algorithmic differences, this paper provides pseudocodes where the two different methods can be described in a unified form, showing that they analyze *multinuclear* relations differently on RST-DTs. As we show by example in Section 4, such a slight difference can derive significantly different DDTs.

The main purpose of this paper is to experimentally reveal the differences between dependency formats. By investigating the complexity of their structures from the dependency graph theoretic point of view (Kuhlmann and Nivre, 2006), we prove that the Hirao13 method, which keeps the semantic equivalence of multinuclear discourse units in the dependency structures, introduces much more complex DDTs than Li14, while a simple post-editing method greatly reduces the complexity of DDTs.

This paper also compares the methods with both intrinsic and extrinsic evaluations: (1) Which dependency structures are analyzed more accurately by automatic parsers? and (2) Which structures

[1] Mann and Thompson (1988)'s Rhetorical Structure Theory (RST), which is one of the most influential text organization frameworks, represents discourse as a (constituent-style) tree structure. RST was developed as the basis of annotated corpora for the automatic analysis of text syntax, most notably the RST Discourse Treebank (RST-DTB) (Carlson et al., 2003).

Proceedings of the SIGDIAL 2016 Conference, pages 128–136,
Los Angeles, USA, 13-15 September 2016.

are more suitable to text summarization? We show from experimental results that even though the Hirao13 DDT format reduces performance, as measured by intrinsic evaluations, it is more useful for text summarization. While researchers developing discourse syntactic parsing (Soricut and Marcu, 2003; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Li et al., 2014) have focused excessively on improving accuracies, our experimental results emphasize the importance of extrinsic evaluations since the more accurate parser does not always lead to better performance of text-based applications.

2 Related Work

Mann and Thompson (1988)'s Rhetorical Structure Theory (RST), which is one of the most influential text organization frameworks, represents a discourse structure as a constituent tree. The RST Discourse Treebank (RST-DTB) (Carlson et al., 2003) has played a critical role in automatic discourse analysis (Soricut and Marcu, 2003; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013), mainly because trees are both easy to formalize and computationally tractable. RST discourse trees (RST-DTs) are also used for modeling many text-based applications, such as text summarization (Marcu, 2000) and anaphora resolution (Cristea et al., 1998).

Hirao et al. (2013) and Li et al. (2014) introduced dependency conversion methods from RST-DTs into DDTs in which a full discourse structure is represented by head-dependent binary relations between elementary discourse units. Hirao et al. (2013) also showed that a text summarization method, based on trimming DDTs, achieves significant improvements against Marcu (2000)'s method using RST-DTs.

On the other hand, some researchers argue that trees are inadequate to account for a full discourse structure (Wolf and Gibson, 2005; Lee et al., 2006; Danlos and others, 2008; Venant et al., 2013). Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003) represents discourse structures as logical form, and relations function like logical operators on the meaning of their arguments. The annotation in the ANNODIS corpus was conducted based on SDRT (Afantenos et al., 2012). For automatic discourse analysis using the corpus, Muller et al. (2012) adopted dependency tree representation to simplify discourse parsing. They also presented a method to automatically derive DDTs from SDR structures.

Wolf and Gibson (2005) used a chain-graph for representing discourse structures and annotated 135 articles from the AP Newswire and the Wall Street Journal. The annotated corpus is called the Discourse Graphbank. The graph represents crossed dependency and multiple parentship discourse phenomena, which cannot be represented by tree structures, but whose graph structures become very complex (Egg and Redeker, 2010).

The Penn Discourse Treebank (PDTB) (Prasad et al., 2008) is a large-scale corpus of annotated discourse connectives and their arguments. Its connective-argument structure can also represent complex discourse phenomena like multiple parentship, but its objective is to annotate the discourse relations between individual discourse units, not full discourse structures. Unfortunately, to the best of our knowledge, neither the Discourse Graphbank nor the PDTB has been used for any specific NLP applications.

3 RST Discourse Tree

RST represents a discourse as a tree structure. The leaves of an RST discourse tree (RST-DT) correspond to *Elementary Discourse Units* (EDUs). Adjacent EDUs are linked by rhetorical relations, forming larger discourse units that are also subject to this relation linking. Figure 1 shows part of an RST-DT (wsj-0623), taken from RST-DTB, for this text fragment:

$$\Big\{[\text{The fiscal 1989 budget deficit figure came out Friday .}]_{\text{e-1}}\Big\}_1,\ \Big\{[\text{It was down a little .}]_{\text{e-2}}\Big\}_2,\ \Big\{[\text{The next time you hear a Member of Congress moan about the deficit ,}]_{\text{e-3}}, [\text{consider what Congress did Friday .}]_{\text{e-4}}\Big\}_3,\ \Big\{[\text{The Senate , 84-6 , voted to increase to \$ 124,000 the ceiling on insured mortgages from the FHA ,}]_{\text{e-5}}, [\text{which lost \$ 4.2 billion in loan defaults last year .}]_{\text{e-6}}\Big\}_4,\ \Big\{[\text{Then , by voice vote , the Senate voted a pork-barrel bill ,}]_{\text{e-7}}, [\text{approved Thursday by the House ,}]_{\text{e-8}}, [\text{for domestic military construction .}]_{\text{e-9}}\Big\}_5,\ \Big\{[\text{the Bush request to what the Senators gave themselves :}]_{\text{e-10}}\Big\}_6, \ldots$$

where each subscript at the end of square brackets [] corresponds to a leaf unit (EDU) in the tree. EDUs grouped by {} consist of a sentence that is labeled with its index in the text.

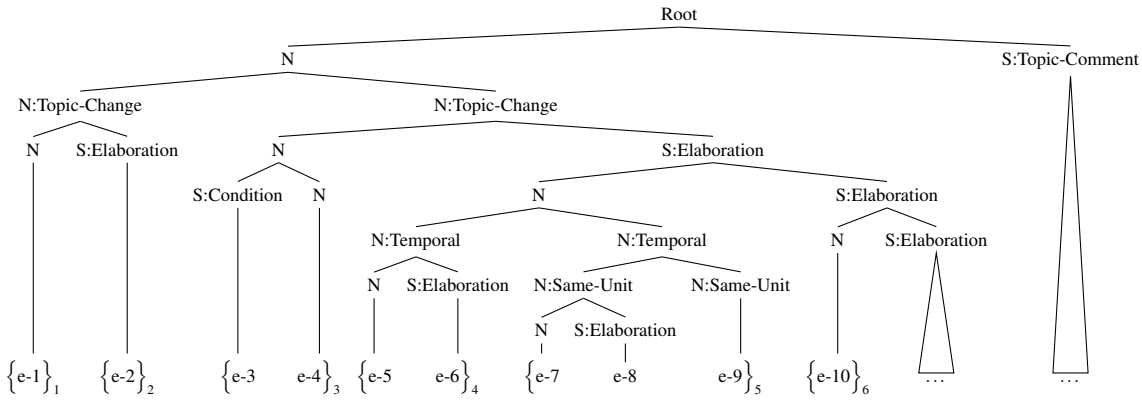

Figure 1: Part of discourse tree (wsj-0623) in RST-DTB: 'S', 'N' and 'e' stand for Satellite, Nucleus and EDU. Each EDU is labeled with its index in the text, and EDUs grouped with {} brackets are in the same sentence.

Each discourse unit in the tree that forms a rhetorical relation is characterized by a rhetorical status: *Nucleus* (N), which represents the most essential piece of information in the relation, or *Satellite* (S), which indicates the supporting information. Rhetorical relations must be either *mononuclear* or *multinuclear*. Mononuclear relations hold between two units with Nucleus and Satellite, whereas multinuclear relations hold among two or more units with Nucleus. Each unit in a multinuclear relation has similar semantic information as the other units. Rhetorical relations can be grouped into classes that share such rhetorical meaning as "Elaboration" and "Condition". In Figure 1, the Satellite unit (covering e-3) and its sibling Nucleus unit (covering e-4) are linked by a mononuclear relation with rhetorical label "Condition", and two Nucleus units (covering e-5, e-6 and e-7, e-8, e-9) are linked by a multinuclear relation with rhetorical label "Temporal".

4 Conversions from RST-DTs to DDTs

Next, this paper discusses text-level dependency syntax, which represents grammatical structure by head-dependent binary relations between EDUs. This section introduces two existing automatic conversion methods from RST-DTs to DDTs: the methods of Li et al. (2014) and Hirao et al. (2013). Additionally, this paper presents a simple post-editing method to reduce the complexity of DDTs. The heart of these conversions closely resembles that of constituent-to-dependency conversions for English sentences (Yamada and Matsumoto, 2003; Johansson and Nugues, 2007; De Marneffe and Manning, 2008), since RST-DTs can be regarded as Penn Treebank-style constituent trees because EDUs and discourse units respectively correspond to terminal and non-terminal nodes, and a rhetorical relation, like a CFG-rule, forms an edge in the tree.

Algorithm 1 convert-rst-into-dep

Require: RST discourse tree: *rst-dt*
Ensure: discourse dependency tree: *ddt*
$ddt \leftarrow \emptyset$
for all EDU e-j in *rst-dt* **do**
  $P \leftarrow \begin{cases} \text{find-My-Top-Node(e-}j\text{) // Li14} \\ \text{find-Nearest-S-Ancestor(e-}j\text{) // Hirao13} \end{cases}$
  if isRoot(P) = TRUE **then**
    $\ell \leftarrow$ Root
    $i \leftarrow 0$
  else
    $\ell \leftarrow$ Label(P)
    $P' \leftarrow$ Parent(P)
    $i \leftarrow$ find-Head-EDU(P')
  end if
  $j \leftarrow$ Index(e-j)
  $ddt \leftarrow ddt \cup (i, \ell, j)$
end for
Return *ddt*

Algorithm 2 find-My-Top-Node(e)

Require: EDU: e
Ensure: C
$C \leftarrow e$
$P \leftarrow$ Parent(e)
while LeftmostNucleusChild(P) = C and
    isRoot(P) = FALSE **do**
  $C \leftarrow P$
  $P \leftarrow$ Parent(P)
end while
if isRoot(P) = TRUE **then**
  $C \leftarrow P$
end if
Return C

4.1 Li et al. (2014)'s Method

Algorithm 3 find-Head-EDU(P)

Require: non-terminal node: P
Ensure: i
1: **while** isLeaf(P) = FALSE **do**
2: $P \leftarrow$ LeftmostNucleusChild(P)
3: **end while**
4: $i \leftarrow$ Index(P)
5: Return i

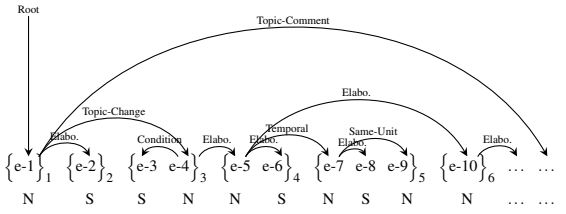

Figure 2: Discourse dependency tree produced by Li's method for RST discourse tree in Figure 1: "Elabo." is short for "Elaboration".

Li et al. (2014)'s dependency conversion method is based on the idea of assigning each discourse unit in an RST-DT a unique head selected among the unit's children. Traversing each non-terminal node in a bottom-up manner, the head-assignment procedure determines the head from its children in the following manner: the head of the leftmost child node with the Nucleus is the head; if no child node is the Nucleus, the head of the leftmost child node is the head.

The procedure was originally introduced by Sagae (2009), and its core idea is identical as the head-assignment rules for Penn Treebank-style constituent trees (Magerman, 1994; Collins, 1999). Li's conversion method uses the procedure to assign a head to each non-terminal node of a right-branching binarized RST-DT (Hernault et al., 2010) and transforms the head-annotated binary tree into a DDT.

Algorithms 1-3 show the dependency conversion method. For brevity, we describe it in a different form from Li's original conversion process[2] cited above. In Algorithm 1, the main routine iteratively processes every EDU in given RST-DT t to directly find its single head rather than transforming head-annotated trees into DDTs. The main process is largely separated into three steps:

1. Algorithm 1 calls Algorithm 2 at line 3, which finds the highest non-terminal node in t to which current processed EDU e-j must be assigned as the head in Sagae's lexicalization manner. Parent(P) and LeftmostNucleusChild(P) are respectively operations that return the parent node of node P and the leftmost child node with the Nucleus of node P[3].

2. After obtaining node P from Algorithm 2, Algorithm 1 seeks the head EDU that is assigned to the parent node of P. If P is the root node of t, we set ℓ to rhetorical label "Root" and i to a special index 0 of virtual EDU e-0 (lines 5-6 in Algorithm 1). Otherwise, we set $\ell \leftarrow$ Label(P) and $P' \leftarrow$ Parent(P) (lines 8-9 in Algorithm 1), where Label(P) returns the rhetorical label attached to node P[4]. Then Algorithm 1 at line 10 calls Algorithm 3, which iteratively seeks the leftmost child node with the Nucleus in a top-down manner, starting from P', until it reaches terminal node e-i. Operation Index(P) returns the index of EDU P.

3. We attach e-j to head e-i and assign rhetorical label ℓ to the dependency edge. We write (i,ℓ,j) to denote that a dependency edge exists with rhetorical label ℓ from head e-i to modifier e-j.

Assuming that e-j is the e-7 of the RST-DT in Figure 1, Algorithm 2 returns the 'N:Temporal' node (covering e-7, e-8, e-9) since its parent node 'N' has the other 'N:Temporal' node (covering e-5, e-6) as its leftmost Nucleus child. Starting from the parent node 'N', Algorithm 3 iteratively seeks the leftmost Nucleus child in the top-down manner until it reaches the terminal node e-5. Finally, we obtain a dependency edge $(5, \text{Temporal}, 7)$.

The DDT in Figure 2 is produced by this method for the RST-DT in Figure 1. To each EDU, we also assign 'N' or 'S' rhetorical status of its parent node. Li's dependency format is always *projective*, i.e., when all the edges are drawn in the half-plane above the text, no two edges *cross* (Kübler et al., 2009).

4.2 Hirao et al. (2013)'s Method

[2]Unlike Li's procedure, our algorithm can take not only binary but also n-ary RST-DTs as inputs. To derive the same DDTs as those produced by Li's original method, experiments were performed on right-branching binary RST-DTs.

[3]If P has no Nucleus children, LeftmostNucleusChild(P) returns the leftmost child node.

[4]If P does not have any rhetorical labels, Label(P) returns a special non-rhetorical label: "Span".

Algorithm 4 find-Nearest-S-Ancestor(e)

Require: EDU: e
Ensure: P
1: $P \leftarrow$ Parent(e)
2: **while** isNucleus(P) = TRUE and
isRoot(P) = FALSE **do**
3: $P \leftarrow$ Parent(P)
4: **end while**
5: Return P

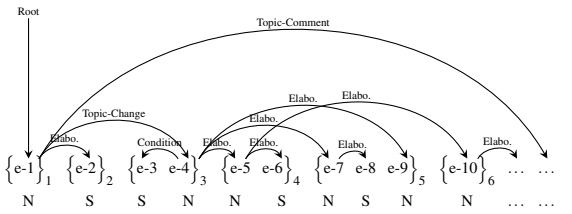

Figure 3: Discourse dependency tree produced by Hirao's method for RST discourse tree in Figure 1.

Hirao et al. (2013) also proposed a dependency conversion method for RST-DTs. The only difference between Li's and Hirao's methods is the process that finds the highest non-terminal node to which each EDU must be assigned as the head. At line 3 of Algorithm 1, Hirao's method calls Algorithm 4, which seeks the nearest Satellite to each EDU on the path from it to the root node of t. Note that this head-assignment manner was originally presented in the Veins Theory (Cristea et al., 1998).

Assuming that e-j is the e-7 in Figure 1, Algorithm 4 returns the 'S:Elaboration' node (covering e-5, e-6, e-7, e-8, e-9, e-10, …), which is the nearest Satellite on the path from e-7 to the root node. Then, as well as in Li's method, Algorithm 3 iteratively seeks the leftmost child node with the Nucleus, starting from the parent node of the Satellite, until it reaches terminal node e-4. Finally, we obtain a dependency edge $(4, \text{Elaboration}, 7)$.

Figure 3 represents the DDT produced by Hirao's method for the RST-DT in Figure 1. Note that unlike Li's method, Hirao's dependency format is not always projective. The dependency edges made from the mononuclear relations are the same as those in Figure 2, but the difference comes from the treatment of the multinuclear relations. We take as an example the "Temporal" multinuclear relation in Figure 1 that links sentences 4 (e-5 and e-6) and 5 (e-7, e-8, and e-9). The Li14 DDT format links them with a "parent-child" relation, while in the Hirao13 DDT format, they have a "sibling" relation.

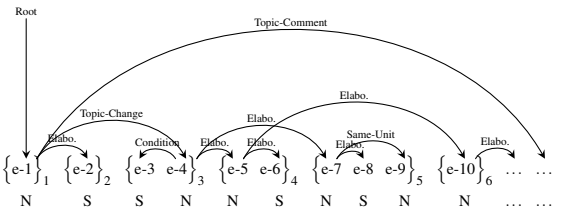

Figure 4: Discourse dependency tree (DDT) obtained by post-editing the DDT in Figure 3.

4.3 Post-editing Algorithm for Multi-rooted Sentence Tree Structures

Unlike Li's method, the dependency structures produced by Hirao's method often lose the single-rooted tree structure of a sentence since Algorithm 4 has no constraints that restrict the EDUs covered by multinuclear relations to find its head outside the sentence. For example, in Figure 3, both EDUs e-7 and e-9 in sentence 5 have the same head e-4 outside the sentence.

Most sentences form a single-rooted subtree in a full-text RST-DT (Joty et al., 2013), and previous studies on sentence-level discourse parsing were based on this insight (Soricut and Marcu, 2003; Sagae, 2009). To reduce the complexity of DDTs, it is reasonable to restrict the tree structure of a sentence to be single-rooted in a full-text DDT.

To revise a multi-rooted dependency tree structure of a sentence to a single-rooted one, we propose a simple post-editing method. Let $\mathscr{L} = \langle \text{e-}x_1, \ldots, \text{e-}x_n \rangle$ be a *multi-root list* consisting of more than two EDUs ($n \geq 2$ and $x_1 < \cdots < x_n$) in identical sentence s, each of which has a head outside s. Next we define the *post-editing* process of multi-root list $\mathscr{L}$; for each EDU e-x_j ($2 \leq j \leq n$), let its head be e-y_j with rhetorical label ℓ_j. Then the post-editing method replaces the dependency edge (y_j, ℓ_j, x_j) by $(x_1, \text{Label}(P), x_j)$, where P is a child node, which covers e-x_j, of the highest node among those that cover only sentence s in the RST-DT.

For the DDT in Figure 3, the post-editing process for multi-root list $\mathscr{L} = \langle \text{e-}7, \text{e-}9 \rangle$ replaces the edge $(4, \text{Temporal}, 9)$ by $(7, \text{Same-Unit}, 9)$. This process makes the tree structure of sentence 5 single-rooted (Figure 4). Note that if an input dependency graph structure is a tree, even after post-editing all the multi-root lists of the input tree, the result remains a tree structure. This post-editing reduces the number of non-projective dependency

Label	Li14	Hirao13	M-Hirao13
Attribution	3070	3182	3176
Background	937	1176	1064
Cause	692	731	729
Comparison	300	200	246
Condition	328	344	338
Contrast	1130	838	892
Elaboration	7902	10358	9242
Enablement	568	609	603
Evaluation	419	596	501
Explanation	986	1527	1255
Joint	1990	42	593
Manner-Means	226	272	266
Root	385	385	385
Same-Unit	1404	62	1092
Span	1	0	1
Summary	223	332	289
Temporal	530	271	355
TextualOrganization	157	137	121
Topic-Change	205	401	344
Topic-Comment	336	326	297

Table 1: Rhetorical label frequencies in automatically created discourse dependency corpora.

edges, even though the structure might continue to be non-projective.

5 Experiments

5.1 Analysis of Dependency Structures

5.1.1 Dependency Label Distributions

Our experiments are based on data from the RST Discourse Treebank (RST-DTB) (Carlson et al., 2003)[5], which consists of 385 Wall Street Journal articles. Following previous studies on RST-DTB, we used 18 coarse rhetorical labels. We converted all 385 RST-DTs to DDTs using the methods introduced in Section 4. Table 1 compares three distributions of 18 rhetorical labels and 2 special non-rhetorical labels: "Span"[6] and "Root". M-Hirao13 denotes a modified version of the Hirao13 dependency format by *post-editing*.

Here, we focus on the three underlined labels. Even though the DDTs produced by the Hirao13 method contain more edges labeled as "Elaboration", the number of "Joint" and "Same-Unit" labels, which are assigned to some multinuclear relations, decreases considerably. This is because for each EDU, Algorithm 4 in the Hirao13 method finds a Satellite covering the EDU through multinuclear relations and most Satellites have the "Elaboration" label.

In practice, we should refine such "Elaboration" labels by encoding in them the information of multinuclear relations that appear on the path from the EDU to the Satellite. However, this encoding scheme has a trade-off; increasing the amount of information encoded in an edge label reduces the accuracy of the label prediction by automatic parsers. In future work, we will investigate what label encoding scheme strikes the best balance in the trade-off.

Property	Li14	Hirao13	M-Hirao13
max path len.	10.2	8.4	8.6
nodes (depth 2)	6.5	9.6	8.6
nodes (depth 3)	14.3	22.1	20.3
nodes (depth 4)	23.3	35.0	33.3
gap degree 0	385	113	247
gap degree 1	0	260	137
gap degree 2	0	12	1
projective	385	113	247
well-nested	385	385	385

Table 2: Experimental results on average maximum path length, number of nodes within depth x, and number of dependency structures that satisfy the property described in Kuhlmann and Nivre (2006).

5.1.2 Complexity of Dependency Structures

This section investigates the complexity of the dependency structures produced by each conversion method. Table 2 shows the average maximum path length from an artificial root to a leaf EDU and the number of nodes where depth $x \in \mathbb{N}$. The results clearly show that Hirao13 produces more broad and shallow dependency tree structures than Li14.

Table 2 also displays how large a portion of the dependency structures is allowed under *projectivity*, *gap degree*, and *well-nestedness* constraints. In the dependency parsing community, it is well-known that these three constraints create a good balance between expressivity and complexity in dependency analysis. These constraints were formally defined (Kuhlmann and Nivre, 2006)[7], and refer to that work for details.

All of the DDTs produced by the Li14 method are projective. Projectivity is the most popular constraint for sentence-level dependency pars-

[5] https://catalog.ldc.upenn.edu/LDC2002T07

[6] In RST theory, a "Span" label may not be assigned to any dependency edges. We suspect that the illegal "Span" label in Table 1 might have been caused by an annotation error in a subtree from e-7 to e-9 of the wsj-1189 file.

[7] Unlike Kuhlmann and Nivre (2006), when calculating the statistics in Table 2, we add an edge $(0, \text{Root}, i)$ for every real root EDU e-i ($i \geq 1$) of the DDT.

		UAS	LAS
MST (Dep)	Li14	66.6	48.3
	Hirao13	55.0	43.1
	M-Hirao13	60.5	42.8
HILDA (RST)	Li14	64.7	49.0
	Hirao13	57.1	46.2
	M-Hirao13	62.4	49.2

Table 3: Dependency unlabeled and labeled attachment scores (UAS and LAS) for MST dependency and HILDA RST parsers.

ing since it offers cubic-time dynamic programming algorithms for dependency parsing (Eisner, 1996; Eisner and Satta, 1999; Gómez-Rodrıguez et al., 2008). A higher gap degree means that the dependency trees have more complex non-projective structures. Both the Hirao13 and M-Hirao13 methods produce many non-projective dependency edges, but most of the DDTs have at most 1 gap degree and all are well-nested. The well-nested dependency structures of the low gap degree also allow efficient dynamic programming solutions with polynominal time complexity to dependency parsing (Gómez-Rodrıguez et al., 2009).

5.2 Impact on Automatic Parsing Accuracy

The conversion methods introduce different complexities in DDTs. This section investigates which formats are more accurately analyzed by automatic discourse parsers. For evaluation, we implemented a maximum spanning tree algorithm for discourse dependency parsing, which was recently proposed (Muller et al., 2012; Li et al., 2014; Yoshida et al., 2014). To compare discourse dependency parsing with standard RST parsing, we also implemented the HILDA RST parser (Hernault et al., 2010), which achieved 82.6/66.6/54.2 points for a standard set of RST-style evaluation measures, i.e., Span, Nuclearity and Relation (Marcu, 2000).

We used a standard split of DDTs automatically converted from RST-DTB: 347 DDTs as the training set and 38 as the test set.

Table 3 shows the evaluation results of dependency parsing. The lower the complexity of the DDT format, the higher is the dependency unlabeled attachment score. Post-editing the Hirao13 DDTs improves the dependency attachment scores because the intra-sentential discourse analysis is more accurate than the inter-sentential one. In all the DDT formats, the labeled attachment scores are considerably worse that the unlabeled scores.

Compared with the HILDA parser, the Hirao13 and M-Hirao13 DDTs by the MST parser are less accurate than those by the RST parser, probably because unlike word dependency parsing, the features defined over the EDUs are too sparse to describe complex non-projective dependency relations.

5.3 Impact on Text Summarization

Hirao et al. (2013) proposed a state-of-the-art single text summarization method based on trimming unlabeled DDTs. That can be formulated by the *Tree Knapsack Problem* (TKP), which they solved with integer linear programming. To examine which dependency structures produced by the three conversion schemes are more suitable to the task, we performed text summarization experiments with the TKP method.

The 30 Wall Street Journal articles have a human-made reference summary, which we used for our evaluations. Table 4 shows the ROUGE scores for the 30 gold-standard and auto-parse DDTs. The auto-parse DDTs were obtained by the MST and HILDA parsers, which were trained with 325 articles and whose hyper parameters were tuned with 30 articles.

Hirao13 achieved the best results when we employed the gold DDTs, although the differences between Hirao13 and the other methods were not large. On the other hand, Hirao13 and M-Hirao13 obtained good results when we employed automatic parse trees. The gains against Li14 are large. It is remarkable that the performance with MST's DDTs closely approached that of the gold DDTs. These results imply that the auto parse trees obtained from Hirao13 have broad and shallow hierarchies because important EDUs, which must be included in a summary, can be easily extracted by TKP. Thus, the DDTs converted by the Hirao13 rule have better tree structures for a single document summarization even though the structures are complex and difficult to parse. This is a significant advantage over Li's conversion rule.

6 Summary

We evaluated two different RST-DT-to-DDT conversion schemes from various perspectives. Experimental results show that even though the Hirao13 DDT format produces more complex dependency structures, it is more useful for text summa-

	Conv.	R-1 w/s.	R-1 wo/s.	R-2 w/s.	R-2 wo/s.
Gold	Li14	.347	.321	.096	.098
	Hirao13	**.349**	**.333**	**.109**	**.117**
	M-Hirao13	.344	.322	.106	.098
MST (Dep)	Li14	.328	.292	.096	.086
	Hirao13	**.341**	**.307**	.106	**.111**
	M-Hirao13	**.341**	.303	**.107**	**.111**
HILDA (RST)	Li14	.315	.281	.083	.086
	Hirao13	**.326**	**.294**	**.087**	**.093**
	M-Hirao13	.315	.285	.084	.089

Table 4: ROUGE-N scores for text summarization on gold and auto-parse DDTs ($N = 1, 2$).

rization. While studies developing discourse parsing have focused on improving parser accuracies, our experimental results identified the importance of extrinsic evaluations over intrinsic evaluations. In future work, we will further compare the methods by extrinsic evaluation metrics using discourse relation labels.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was supported in part by JSPS KAKENHI Grant Numbers JP26730126 and JP26280079.

References

Stergos Afantenos et al. 2012. An empirical resource for discovering cognitive principles of discourse organisation: the annodis corpus. In *Proceedings of the 12th International Conference on Language Resources and Evaluation*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. *Building a discourse-tagged corpus in the framework of rhetorical structure theory*. Springer.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, Stanford University.

Dan Cristea, Nancy Ide, and Laurent Romary. 1998. Veins theory: A model of global discourse cohesion and coherence. In *Proceedings of the 17th International Conference on Computational linguistics-Volume 1*, pages 281–285.

Laurence Danlos et al. 2008. Strong generative capacity of rst, sdrt and discourse dependency dags. pages 69–95.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Markus Egg and Gisela Redeker. 2010. How complex is discourse structure? In *Proceedings of the 7th International Conference on Language Resources and Evaluation*.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 457–464. Association for Computational Linguistics.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th international conference on Computational linguistics*, pages 340–345.

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: Long Papers-Volume 1*, pages 60–68.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32.

Carlos Gómez-Rodrıguez, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL08: HLT)*, pages 968–976.

Carlos Gómez-Rodrıguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 291–299.

Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3).

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

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *16th Nordic Conference of Computational Linguistics*, pages 105–112.

Shafiq R Joty, Giuseppe Carenini, Raymond T 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, pages 486–496.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 507–514.

Alan Lee, Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, and Bonnie Webber. 2006. Complexity of dependencies in discourse: Are dependencies in discourse more complex than in syntax. In *Proceedings of the 5th International Workshop on Treebanks and Linguistic Theories, Prague, Czech Republic, December*.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 25–35.

David M Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, University of Pennsylvania.

William Mann and Sandra Thompson. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. pages 1883–1899. Proceedings of the 24th International Conference on Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*.

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 81–84. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156.

Antonine Venant, Nicholas Asher, Philippe Muller, Pascal Denis, and Stergos Afantenos. 2013. Expressivity and comparison of models of discourse structure. In *Proceedings of the SIGDIAL*, pages 2–11.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–287.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, volume 3.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1834–1839.