

図 2: 図 1 の RST-DT に対応する DEP-DT

2 談話構造木

2.1 修辞構造型論に基づく談話構造木

RST では、文書中の有意味な最小のテキストスパン、Elementary Discourse Unit (EDU:節に相当する) は、「核 (Nucleus)」,「衛星 (Satellite)」のいずれかの属性を持っており、隣接する EDU をまとめあげていったテキストスパンもまた核か衛星の属性を持つ。さらに、核と衛星、核と核の間にはその修飾関係の方向とその意味を表す関係子が定義されている。これらの関係を非終端記号が核か衛星、終端記号が EDU である木として表現したものが RST-DT である。図 1 に RST-DT の例を示す。図中、 e は EDU を表し、 N は核、 S は衛星を表す。

2.2 依存構造に基づく談話構造木

本節では、要約生成における RST-DT の問題点を指摘し、RST-DT をより要約生成に適した構造である DEP-DT への変換法について説明する。

RST-DT の構造を制約として取り入れるための最も素朴な方法は、木構造を破壊することなく長さ制約を満たす EDU を選択することであろう。つまり、RST-DT の枝を刈り込むことで要約を生成する問題として定式化すれば良い。

しかし、RST-DT では、隣接していない EDU 間の依存関係が明示されていないため、ある EDU を削除する際、直接依存関係にある EDU を削除すべきかどうかの判断は容易であるが、それ以外の EDU を削除すべきかどうかを判断することが難しい。たとえば、 e_8 を削除する場合、それを直接修飾する e_7 を削除しなければならないことまでは分かるが、 e_8 に対する

e_9, e_{10} の依存関係が明示されていないため、これらを削除しなければならないかどうかの判断が難しい。文書を EDU 間の依存構造木として表現できたなら、ある EDU を削除する時、その子孫もすべて削除しなければならないことは容易に分かる。このように、木の刈り込みにより要約を生成するという目的を考えた場合、EDU 間の依存関係が直接分かる構造の方が好ましい。

本稿では、EDU 間の依存関係を直接表した談話構造木である DEP-DT として文書を表現することとする。以下に提案する RST-DT から DEP-DT への変換手続きを用いることで RST-DT を一意に DEP-DT へと変換できる。

1. RST-DT のすべてのノードに対しヘッドを設定する。EDU の親であるノードのヘッドは未定義とし、それ以外については、子孫の中で最左の N の子供である EDU をヘッドとする。
2. S 属性を持つ EDU を直近のヘッドが定義されている先祖ノードのヘッドにかける。
3. N 属性を持つ EDU はヘッドが定義されている直近の先祖の S を探した後、その S に対してヘッドが定義されている直近の先祖ノードのヘッドにかける。
4. 先祖に S がいない N 属性を持つ EDU を根のヘッドにかける。

上記手続きを図 1 の RST-DT に適用すると図 2 の DEP-DT を得る。DEP-DT では、RST-DT で定義されていたノード間の関係詞の情報が失われることに注意されたい¹。図 2 より、 e_8 を削除するときにはその子孫である e_7, e_9, e_{10} も同時に削除しなければならないことが分かる。

3 依存構造木の刈り込みによる要約

3.1 定式化

図 2 のような EDU をノードとする依存構造木が与えられた時、すべての根付き部分木の集合を T とし、その要素である木 $t \in T$ のスコアを $F(t)$ とする。依存構造木の刈り込みによる要約生成は、長さ制約 L のもとで $F(t)$ を最大化する部分木 t^* を求める問題、つまり、木制約のもとで

¹ただし、RST-DT において直接依存関係がある EDU 間、すなわち、衛星属性の EDU から核属性の EDU への依存関係の関係詞は保存することができる。

Algorithm 1 Dynamic programming for summary generation

```

1:  $C \leftarrow \text{GetChildren}(n)$ 
2: Initialize  $c_{\max}$ 
3: if  $C \neq \phi$  then
4:    $c_{\max} \leftarrow \text{MaxVariation}(C)$ 
5:    $C \leftarrow C \setminus \{c_{\max}\}$ 
6:   for each array  $c$  in  $C$  do
7:      $S \leftarrow \text{GenSubTree}(c)$ 
8:      $j \leftarrow 0$ 
9:     for each subtree  $s$  in  $S$  do
10:      for  $k = 0$  to  $L - \sum_{r \in \text{anc}(n) \cup \{n\}} \ell(r)$  do
11:         $a_j[k] = \max(c_{\max}[k], c_{\max}[k - \ell(s)] + w(s))$ 
12:      end for
13:       $j \leftarrow j + 1$ 
14:    end for
15:     $c_{\max} \leftarrow \text{Update\_by\_MaxElm}(a_0, \dots, a_j)$ 
16:  end for
17: end if
18: for  $m = 0$  to  $L - \sum_{r \in \text{anc}(n)} \ell(r)$  do
19:   if  $m < \ell(n)$  then
20:      $v_n[m] = 0$ 
21:   else
22:      $v_n[m] = c_{\max}[m - \ell(n)] + w(n)$ 
23:   end if
24: end for

```

のナップサック問題として以下の通り定式化できる。なお、この問題は木の分割問題 [Lukes, S. A. 74, Jhonson, D. S. and Niemi, K.A. 83] の変種として捉えることができる。

$$\begin{aligned}
 t^* &= \arg \max_{t \in T} F(t) \\
 \text{s.t. } &\text{Length}(t) \leq L
 \end{aligned}$$

$F(t)$ は、 t に含まれる EDU のスコアの和として以下の式で定義する。

$$F(t) = \sum_{e \in E_t} \frac{W(e)}{\text{depth}(e)}$$

ここで、 E_t は木 t に含まれる EDU の集合、 $\text{depth}(e)$ は、根を 1 とした時の T における e の深さ、 $W(e)$ は以下に定義する e のスコアである。

$$W(e) = \sum_{w \in W_e} S(w)$$

W_e は e に含まれる単語の集合を表し、 $S(w)$ は単語 w の重要度である²。

3.2 アルゴリズム

t^* を求めるため、長さ制約を満たす T の根付き部分木を最右拡張 [Abe 02] を用いて数え上げること

²本稿では $\text{tf} \cdot \text{idf}$ に基づき単語重要度を決定した。

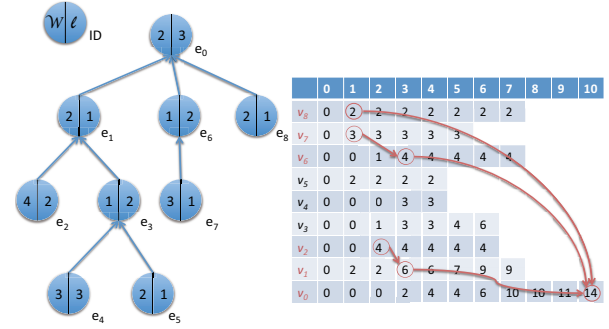


図 3: 要約生成の動作過程

能ではあるが、木の数え上げは NP 困難であるため、現実的な時間で終わるとは限らない。本稿では、ナップサックアルゴリズムにヒントを得て、動的計画法に基づく効率的なアルゴリズムを提案する。

長さの上限 L が与えられたとき、 T の任意のノード e_n に対して許される要約 (e_n の子孫からなる部分木) の長さの上限は L から、 e_n の親から根までの EDU の長さの和を引いたもの、つまり、 $L - \sum_{r \in \text{anc}(e_n)} \ell(r)$ となる。 $\text{anc}(e_n)$ は、 e_n から根に至るまでのパス上のノード集合を表す。ただし、 e_n 自身はこれに含まめない。ここで、 e_n に対して許容される全ての長さ、つまり、 0 から $L - \sum_{r \in \text{anc}(e_n)} \ell(r)$ までの長さの要約スコアの最大値は、 e_n の子ノードにおける各長さまでの要約スコアの最大値の組合せによって決定される。よって、すべてのノードに長さ $L - \sum_{r \in \text{anc}(e_n)} \ell(r) + 1$ の配列を用意し、葉ノードから根に至るまで順に許容される可能な長さの要約の最大値を順に求めていくことで、 t^* を求めることができる³。Algorithm 1 に e_n に対する最大要約スコアを求める手順を示す。

まず、 e_n の子ノードの配列の集合を C とする (1 行目)。 C が空でない場合、配列中に格納されたスコアの異なり数が最も多い配列を c_{\max} として、 C から取り出す (4,5 行目)。次に C から 1 つの配列を取り出し、格納されたスコアの異なり数に応じて、部分木集合⁴ S を得る。 S から取り出した部分木と c_{\max} を用いてナップサック問題を解き、その結果を配列 a へ保存する (10~12 行目)。 S のすべての要素に対して処理を終えた後、得られた j 個の配列 a_0, \dots, a_j の各添字に対し、最大値のみを保持した配列で c_{\max} を上書きする (15 行目)。最後に c_{\max} を右に e_n の長さだけシフトさせ、 e_n のスコアを足すことにより、配列 v_n を得る (18~24 行目)。なお、11 行目、22 行目で配列の

³ T を S 式で表現し、右側ノードから順に処理を行えばよい。

⁴配列に含まれるスコアの異なり数だけ部分木が存在する。

表 1: 評価結果

	ROUGE-1
提案手法	0.302
[Marcu 98]	0.292

値を更新する時に利用した部分木を記憶させておく。すると、根の配列を処理し終えた段階で、最大スコアを持つ要素に t^* が記憶される。

図 3 に提案したアルゴリズムを用いて $T = (e_0(e_1(e_2)(e_3(e_4)(e_5)))(e_6(e_7))(e_8))$ 対し、長さ $L = 10$ の要約を生成する過程を示す。図より、 $F(t^*) = 14$ 、 $t^* = (e_0(e_1(e_2))(e_6(e_7))(e_8))$ となる。

4 評価実験

提案手法の有効性を示すため、[Marcu 98] で提案された RST-DT を利用した要約生成手法との比較評価を行った。以下に [Marcu 98] の要約生成手順を説明する。

1. RST-DT において、核属性を持つ EDU を直近の衛星属性を持つ先祖まで昇格させる。先祖に衛星属性を持つノードがない EDU は、根まで昇格させる。
2. 根から順に RST-DT の高さに応じて EDU の優先順序を決定する。ただし、同じ高さに複数の EDU が割り当てられている場合、それらの間の順序関係は考えない。
3. 長さ制約を満たすまで優先順位の高い EDU を要約として選択する。

図 1 の例では、EDU 間の優先順序関係は以下の通りとなる。

$$e_2 > e_8 > e_3, e_{10}, > e_1, e_4, e_5, e_7, e_9, > e_6$$

実験には RST Discourse Treebank コーパスを用いた [Carlson 01]。これは、Penn Treebank の一部の記事に対し、RST に基づく談話構造木を手で与えたコーパスであり、そのうち、29 記事に人間が作成した要約が与えられている。提案手法、Marcu の方法ともこの 29 文書に付与された正解の RST-DT に基づき要約を生成した。なお、長さ制約 L は人間が作成した正解要約と同じ単語数とし、評価指標には、ROUGE-1 を用いた [Lin 04]。29 文書での ROUGE-1 スコアの平均値を表 1 に示す。提案手法が Marcu の手法よりも 1 ポイント ROUGE スコアが高い。2 手法の間に統計

的に有意な差はみられないが、全 29 文書のうち 19 文書で提案手法が Marcu の手法以上の ROUGE スコアを獲得しており、有効性が確認できる。

5 おわりに

本稿では、要約生成を与えられた長さ制約のもとで談話構造木の根付き部分木を選択する問題として定式化し、動的計画法を用いて効率的に最適解を得る手法を提案した。また、談話構造木として従来から用いられている RST-DT を要約生成に適した DEP-DT へ変化する手続きも提案した。提案手法を RST-DT に基づく要約生成手法である [Marcu 98] と比較したところ、提案手法がより良い ROUGE スコアを獲得した。

参考文献

- [Abe 02] Abe, K., Kawazoe, S., Asai, T., Arimura, H., and Arikawa, S.: Optimized Substructure Discovery for Semi-structured Data, in *Proc. of the 6th PKDD*, pp. 1–14 (2002)
- [Carlson 01] Carlson, L., Marcu, D., and Okurowski, M.: Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory, in *Proc. of the SIG-DIAL01*, pp. 1–10 (2001)
- [Filatova 04] Filatova, E. and Hatzivassiloglou, V.: A Formal Model for Information Selection in Multi-document Sentence Extraction, in *Proc. of the 20th COLING*, pp. 397–403 (2004)
- [Jhonson, D. S. and Niemi, K.A. 83] Jhonson, D. S. and Niemi, K.A.: On Knapsacks, Partitions, and a New Dynamic Programming Technique for Trees, *Mathematics of Operations Research*, Vol. 8, No. 1, pp. 1–14 (1983)
- [Lin 04] Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of Summaries, in *Proc. of Workshop on Text Summarization Branches Out*, pp. 74–81 (2004)
- [Lukes, S. A. 74] Lukes, S. A.: Efficient algorithm for the partitioning of trees, *IBM Journal of Research and Development*, Vol. 18, No. 3, pp. 217–224 (1974)
- [Mann, W. C. and Thompson, S. A. 88] Mann, W. C. and Thompson, S. A.: Rhetorical Structure Theory: Toward a functional theory of text organization, *Text*, Vol. 8, No. 3, pp. 243–281 (1988)
- [Marcu 98] Marcu, D.: Improving summarization through rhetorical parsing tuning, in *Proc. of the 6th Workshop on Very Large Corpora*, pp. 206–215 (1998)
- [McDonald 07] McDonald, R.: A Study of Global Inference Algorithm in Multi-document Summarization, in *Proc. of the 29th ECIR*, pp. 557–564 (2007)
- [Takamura 09] Takamura, H. and Okumura, M.: Text Summarization Model based on Maximum Coverage Problem and its Variant, in *Proc. of the 12th EACL*, pp. 781–789 (2009)