オラクル要約の列挙

平尾努　　西野正彬　　鈴木潤　　永田昌明
日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
{hirao.tsutomu,nishino.masaaki,suzuki.jun,nagata.masaaki}@lab.ntt.co.jp

1　はじめに

オラクル要約とは，人間が生成した参照要約との間の評価関数の値が最大，かつ，要約システムが出力可能な要約を指す．以降，本稿では，評価関数として，ROUGE$_n$[Lin 04] を採用し，システムは文抽出により要約を生成することを仮定するので，オラクル要約とは，システムが獲得可能な ROUGE$_n$ の上限値を持つ文の集合を指す．なお，与えられた文集合中に似た文がある場合，1 つの参照要約に対し複数のオラクル要約が存在することに注意されたい．

オラクル要約を得ることで，我々は要約システムの性能を詳細に分析できるようになる．たとえば，ROUGE$_n$ は [0,1] の値をとるが，文の抽出しかしない要約システムが人間が「生成」した要約に対し，上限値である 1 を獲得することはほぼあり得ないため，スコアを直感的に解釈することが難しい．たとえば，あるシステムの ROUGE$_n$ が 0.5 だったとしよう．この時，そもそもシステムは 1 という上限値を獲得できないにもかかわらず，それに対し，50%までしか到達できなかったととらえるべきだろうか．ここで，オラクル要約がわかっていれば，それに対してどこまで近づけたかでシステムを評価できる．オラクル要約の ROUGE$_n$ が 0.7 であれば，先のシステムは上限値に対し，約 70%を達成したことがわかるので ROUGE$_n$ の解釈が容易になる．さらに，オラクル要約は文の集合であるため，要約システムが抽出に成功，失敗した文がわかる．よって，文を基準とした再現率も計算できる．システムが出力した文集合を X，オラクル要約を O とすると，再現率 $=|O \cap X|/|O|$ となる．ただし，オラクル要約が複数あると任意のオラクル要約のみに対して計算した再現率だけでは不当に低い評価になる可能性がある．よって，可能なオラクル要約をすべて列挙しておき，それらとの間で計算した再現率のうち最も高いスコアを採用する必要がある．しかし，与えられた文集合からオラクル要約を列挙するためには文の数に対して指数オーダの計算量が要求される．このため，従来では，貪欲法などを用いて近似オラクル要約を得ていた．たとえば，Sipos らは，ROUGE$_n$ が単調増加な劣モジュラ関数であることに着目し，精度保証付き貪欲法 [Khuller 99] で効率的に近似オラクルを求めた [Sipos 12]．しかし，こうした手法では，近似オラクルしか得られない上，それらを列挙しようとすると，結局，文の数に対し指数オーダの計算量が要求されてしまう．そこで本稿では，分枝限定法 [Land 60] に基づき，ROUGE$_n$ スコアの下限値と上限値を利用して探索候補を絞り込むことで効率的にオラクル要約を列挙するアルゴリズムを提案する．

提案手法を用いて，Document Understanding Conference (DUC) 2003, 2004 のコーパスからオラクル要約を列挙し，分析した結果，提案手法で得たオラクル要約の ROUGE$_n$ スコアは，貪欲法で得た近似オラクル要約の ROUGE$_n$ スコアよりも高いことを確認した．また，80 から 90% の参照要約が複数のオラクル要約を持つこと，単なる長さの制約のみで探索する場合と比較して，提案手法は探索空間を大幅に削減できたことを明らかにした．

2　分枝限定法によるオラクル要約の探索

2.1　探索戦略

本稿では，任意のノードから根に至るまでの経路がオラクル要約の候補 (つまり，任意の文集合) を表す探索木を利用する．文の数を k とするならば，探索木のノード数は 2^k となる．図 1 に $k = 5$ の場合の探索木を示す．

この探索木を根から順に深さ優先で長さ制約を満たすノードまでを探索し，ROUGE$_n$ スコアを記録していけばオラクル要約を列挙することができる．だが，長さの制約があるとは言え，2^k 通りの組合せを探索することは現実的ではない．

一方，根から任意のノードまでたどった段階で，ROUGE$_n$ の下限値とそのノード以下の探索を続けた際に得ることができる上限値がわかれば，それ以下のノードの探索を続けるか，探索を破棄し，新たな文の組合せを探索するかの判断ができるので，効率的に探索できる．図 1 の例では，root $\rightarrow s_1 \rightarrow s_2$ とたどっ

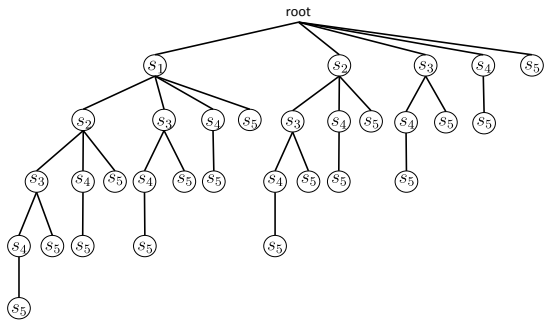

図 1: 探索木の一例

た段階で，それ以下のノードを探索した際に得られる ROUGE$_n$ の上限値がこれまでに記録した ROUGE$_n$ の下限値を超えるのであれば探索を続け，超えないのであればそれ以下のノードの探索は行わず，root $\rightarrow s_1 \rightarrow s_3$ という新たな文の組合せから探索を続行すれば良い．以降，ROUGE$_n$ の上限値，下限値の求め方，それに基づくオラクル要約の探索アルゴリズムを説明する．

2.2 2 つの排他的な文集合の和集合に対する ROUGE$_n$ の定義

ROUGE$_n$ の上限値を求めるための準備として，V, W という排他的な集合の和集合 $V \cup W$ に対する ROUGE$_n$ を定義する．

$\mathcal{R}$ を参照要約に出現する N グラムの多重集合，D を任意の文集合とした時，$\mathcal{R}$ と D との間の ROUGE$_n$ スコア (以降，式の中では R_n と略記する) は，以下の式で定義される．

$$\mathrm{R}_n(\mathcal{R}, D) = \frac{\sum_{t_n \in \mathcal{U}(\mathcal{R})} \min(\mathrm{cnt}(t_n, \mathcal{R}), \mathrm{cnt}(t_n, \mathcal{B}(D)))}{\sum_{t_n \in \mathcal{U}(\mathcal{R})} \mathrm{cnt}(t_n, \mathcal{R})} \quad (1)$$

t_n は，長さ n の N グラムを表し，関数 cnt は，t_n の多重集合内での頻度を返す関数であり，$\mathcal{U}$ は多重集合を集合へと変換する関数である．さらに，文集合をそこに出現する N グラムの多重集合へと変換する関数を $\mathcal{B}$ とする．ここで，排他的な 2 つの文集合 V, W に対し以下の定理が成立する (証明は割愛する)．

定理 1 $V \cup W$ に対する ROUGE$_n$，V に対する ROUGE$_n$，V, W に対する ROUGE$_n$′ (R'_n) の間に以下の関係が成立する．

$$\mathrm{R}_n(\mathcal{R}, V \cup W) = \mathrm{R}_n(\mathcal{R}, V) + \mathrm{R}'_n(\mathcal{R}, V, W) \quad (2)$$

ただし，ROUGE$_n$′ は以下の式で定義する．

$$\mathrm{R}'_n(\mathcal{R}, V, W) = \frac{\sum_{t_n \in \mathcal{U}(\mathcal{R} \setminus \mathcal{B}(V))} \min(\mathrm{cnt}(t_n, \mathcal{R} \setminus \mathcal{B}(V)), \mathrm{cnt}(t_n, \mathcal{B}(W)))}{\sum_{t_n \in \mathcal{U}(\mathcal{R})} \mathrm{cnt}(t_n, \mathcal{R})} \quad (3)$$

2.3 ROUGE$_n$ の上限値

探索木において，任意のノードから根までの経路に対応する文集合を V，任意のノードの子ノードとなり得る文の集合を Ω，参照要約の長さを $L_{\mathcal{R}}$ とすると，任意のノード以下を探索した際に得られる ROUGE$_n$ スコアの上限値 $\widehat{\text{ROUGE}}_n$(以降，$\widehat{\mathrm{R}_n}$) は以下の式で与えられる．

$$\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega) = \max_{W \subseteq \Omega} \{\mathrm{R}_n(\mathcal{R}, V \cup W) : \ell(V \cup W) \leq L_{\mathcal{R}}\} \quad (4)$$

Algorithm 1 FINDUPPERBOUND

```
Function: UPPER(V, Ω, L)
  for each w ∈ Ω do
    append(W, R'_n(R, V, {w}) / ℓ({w}))
  end for
  sort(W, 'decsend')
  U ← 0
  for each w ∈ W do
    if L − ℓ({w}) ≥ 0 then
      U ← U + R'_n(R, V, {w})
      L ← L − ℓ({w})
    else
      U ← U + (R'_n(R, V, {w}) / ℓ({w})) × L
      break the loop
    end if
  end for
  return U
end
```

ℓ は文集合の長さを返す関数である．ここで，式 (2) の関係を用いて式 (4) を変形すると以下の式を得る．

$$\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega) = \mathrm{R}_n(\mathcal{R}, V) + \max_{W \subseteq \Omega} \{\mathrm{R}'_n(\mathcal{R}, V, W) : \ell(W) \leq L_{\mathcal{R}} - \ell(V)\} \quad (5)$$

式 (5) の右辺第 2 項の最適解を求めることは NP 困難であるが，次の不等式が成り立つことに注意する．

$$\mathrm{R}'_n(\mathcal{R}, V, W) \leq \sum_{w \in W} \mathrm{R}'_n(\mathcal{R}, V, \{w\}) \quad (6)$$

この関係を用いると，式 (5) 右辺第 2 項に対し以下の不等式が成り立つ．

$$\max_{W \subseteq \Omega} \{\mathrm{R}'_n((\mathcal{R}, V, W) : \ell(W) \leq L_{\mathcal{R}} - \ell(V)\} \leq \max_{\boldsymbol{x}} \left\{ \sum_{i=1}^{|\Omega|} \mathrm{R}'_n(\mathcal{R}, V, \{w_i\}) x_i : \sum_{i=1}^{|\Omega|} \ell(\{w_i\}) x_i \leq L_{\mathcal{R}} - \ell(V) \right\} \quad (7)$$

ただし，$x_i \in \{0, 1\}, w_i \in \Omega$．式 (7) 右辺は整数計画問題，0-1 ナップサック問題である．この厳密解は ILP ソルバ，動的計画法によるアルゴリズムを用いて得ることができるが，Ω，$L_{\mathcal{R}} - \ell(V)$ の値によっては計算コストが高い．そこで，本稿ではこの整数計画問題を線形計画問題に緩和し，Algorithm 1 に示す貪欲法にて解を求める．Algorithm 1 では，まず，Ω の各要素 (文) を単位長さあたりのアイテムとしてとらえ，その ROUGE' スコアでアイテムを降順にソートしておく．そして，先頭からちょうど長さ L までアイテムを選択した際の ROUGE' スコアの和を返す．0-1 ナップサック問題の緩和問題の解はもとの問題の厳密解よりも大きくなることが保証されており，貪欲法で厳密解が得られるため計算量が少ないという利点がある．

よって，任意のノードまでたどった際の ROUGE$_n$ の上限値 (式 (4)) は以下で再定義される．

$$\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega) = \mathrm{R}_n(\mathcal{R}, V) + \mathrm{UPPER}(V, \Omega, L_{\mathcal{R}} - \ell(V)) \quad (8)$$

Algorithm 2 FINDLOWERBOUND

Function: GREEDY($\mathcal{R}, D, L_{\mathcal{R}}$)
  $L \leftarrow 0, C \leftarrow \phi, E \leftarrow D$
  while $E \neq \phi$ **do**
    $s^* \leftarrow \arg\max_{s \in E} \left\{ \frac{\mathrm{R}_n(\mathcal{R}, C \cup \{s\}) - \mathrm{R}_n(\mathcal{R}, C)}{\ell(\{s\})} \right\}$
    $L \leftarrow L + \ell(\{s^*\})$
    if $L \leq L_{\mathcal{R}}$ **then**
      $C \leftarrow C \cup \{s^*\}$
    end if
    $E \leftarrow E \setminus \{s^*\}$
  end while
  $i^* \leftarrow \arg\max_{i \in D, \ell(\{i\}) \leq L_{\mathcal{R}}} \mathrm{R}_n(\mathcal{R}, \{i\})$
  $S \leftarrow \arg\max_{K \in \{\{i^*\}, C\}} \mathrm{R}_n(\mathcal{R}, K)$
  return $\mathrm{R}_n(\mathcal{R}, S)$
end

Algorithm 3 FINDORACLE

Read $\mathcal{R}$,D,$L_{\mathcal{R}}$
$\tau \leftarrow$ GREEDY($\mathcal{R}, D, L_{\mathcal{R}}$),$O_\tau \leftarrow \phi$, $C \leftarrow \phi$
for each $s \in D$ **do**
  append(S,$< \mathrm{R}_n(\mathcal{R}, \{s\}), s >$)
end for
sort(S,'decsend')
call FINDORACLE(S, C)
Procedure: FINDORACLE(Q, V)
  while $Q \neq \phi$ **do**
    $s \leftarrow$shift(Q)
    append(V, s)
    $\Omega \leftarrow$suffix(S,last(V)))
    $L_{\mathrm{rem}} \leftarrow L_{\mathcal{R}} - \mathrm{len}(V)$
    if $L_{\mathrm{rem}} \geq 0$ **then**
      if $\mathrm{R}_n(\mathcal{R}, V) \geq \tau$ **then**
        $\tau \leftarrow \mathrm{R}_n(\mathcal{R}, V)$
        append(O_τ, V)
        call FINDORACLE(Q, V)
      else if $\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega) \geq \tau$ **then**
        call FINDORACLE(Q, V)
      end if
    end if
    pop(V)
  end while
end

2.4 ROUGE$_n$ の下限値

ROUGE$_n$ の上限値と下限値の差が小さければ小さいほど，探索候補の絞り込みが効果的になる．よって，良い下限値を得ることが必要とされる．本稿では，ROUGE$_n$ が劣モジュラ関数 [Lin 11] であることに着目し，精度保証付き貪欲法で得た近似オラクルの ROUGE$_n$ スコアを初期値として利用する．アルゴリズムを Algorithm 2 に示す．Algorithm 2 では，ROUGE$_n$ スコアの利得が最大となる文を長さ制約を満たす間，逐次的に C へ追加していき，最後に単体で ROUGE$_n$ が最大の文とスコアを比較し，大きい方のスコアを返す．なお，このアルゴリズムで得た近似オラクルの ROUGE$_n$ スコアはオラクルの ROUGE$_n$ スコアの $\frac{1}{2}(1\text{-}\frac{1}{e})$ 以上であることが保証される．こうして得たスコアで下限値 τ を初期値しておき，探索を続けて行く間にこれよりも大きな ROUGE$_n$ スコアを記録すればその値で τ を更新しておく．

2.5 オラクル探索アルゴリズム

根から任意ノードまでの経路に対応する文集合を V，そのノードの子ノードとなり得る文集合を Ω とした時，それ以下のノードの探索は次の 3 通りの条件により決定する．

1. $\mathrm{R}_n(\mathcal{R}, V) \geq \tau$
2. $\mathrm{R}_n(\mathcal{R}, V) < \tau$, かつ，$\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega) < \tau$
3. $\mathrm{R}_n(\mathcal{R}, V) < \tau$, かつ，$\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega) \geq \tau$

1 の場合，V をオラクル要約として更新し，τ を $\mathrm{R}_n(\mathcal{R}, V)$ で上書きした後，更にそれ以下のノードの探索を続ける．2 の場合，それ以下のノードの探索を打ち切る．3 の場合，$\mathrm{R}_n(\mathcal{R}, V)$ は τ を超えないが，$\widehat{\mathrm{R}_n}(\mathcal{R}, V, \Omega)$ が τ を超えるため，それ以下のノードを探索した際に現在のオラクル要約の ROUGE$_n$ スコアを超える文集合が存在する可能性がある．よって，オラクル要約の更新は行わず探索のみを続ける．これらの探索戦略に基づくオラクル要約探索のアルゴリズムを Algorithm 3 に示す．

Algorithm 3 では，参照要要約の多重集合，テキストユニット集合，長さ制約 $L_{\mathcal{R}}$ を読み込み，スコアが τ であるオラクルを格納する O_τ，探索木をたどった履歴を保持するプライオリティキュー C を初期化し，Algorithm 2 で求めた下限値を τ にセットする．次に，各文の ROUGE スコアを計算し，それらを降順にソートし，S に格納する．そして，手続き FINDORACLE に引数として，S と C を与え，手続きの再帰呼び出しにて深さ優先探索を行う．FINDORACLE では，第 1 引数のプライオリティキュー Q から先頭の文を取り出し，第 2 引数のプライオリティキュー V に追加する (10,11 行目)．そして，S の先頭から V の最後の要素である文までを削除したものを Ω とする (12 行目)．長さ制約は，全体の長さ制約 $L_{\mathcal{R}}$ から V に格納されているテキストユニットの長さの和を引いたものとし，これがゼロ以上の場合に先に示した 3 通りの条件でオラクル要約の更新，探索を続けるか否かの判断をする．最後に，V の最後の要素を削除し (23 行目) 再帰手続きの先頭に戻る．なお，同じ下限値を持つオラクル要約を記録しておくことですべてのオラクル要約を列挙できる (17 行目)．

表 1: オラクル要約の ROUGE スコア (ストップワードなし)

	DUC-03		DUC-04	
	$n=1$	$n=2$	$n=1$	$n=2$
Exact	.424*	.183*	.404*	.154*
Greedy	.418	.182	.393	.152

表 2: 1 つの参照要約あたりのオラクル要約の数

	DUC-03		DUC-04	
	$n=1$	$n=2$	$n=1$	$n=2$
中央値	7.5	10	8	11
最大値	1082	792	6894	4601
最小値	1 (0.15)	1 (0.67)	1 (0.14)	1 (0.10)

3 実験結果と考察

3.1 コーパス

DUC-03,04 の複数文書要約タスクのデータを用いてオラクル要約を列挙した．DUC-03 は 30 文書セッ

表 3: 探索したノード数

	DUC-03			DUC-04		
	中央値	最大値	最小値	中央値	最大値	最小値
実行可能解の数	208×10^{9}	205×10^{13}	513×10^{5}	158×10^{9}	462×10^{17}	176×10^{7}
ノード数 ($n=1$)	373×10	261×10^{3}	386	495×10	115×10^{4}	488
ノード数 ($n=2$)	209	653×10	180×10^{-1}	267	911×10^{2}	230×10^{-1}

ト，1 文書セットあたりの平均文書数は約 10，平均文数は 258 である．DUC-04 は，50 文書セット，平均文書数は 10，平均文数は 261 である．1 文書セットあたり 4 つの参照要約 (100 単語) が用意されている．

3.2 オラクル要約の ROUGE_n スコア

提案手法 (Algorithm 3):Exact と貪欲法 (Algorithm 2):Greedy で求めたオラクル要約の $\text{ROUGE}(n=1,2)$ スコアを比較した．表 1 に平均値を示す．先に説明したとおり，貪欲法が保証する最適値に対する理論的下限は $\frac{1}{2}(1-\frac{1}{e})\simeq 0.32$ 倍であるが，この実験結果ではほぼ最適値に近いスコアを記録している．ただし，双方の差をウィルコクソンの符号順位検定 [Wilcoxon 45] を用いて検定したところ，その差は有意であった．つまり，差は小さくとも提案手法で得たオラクル要約の ROUGE スコアは貪欲法で求めたそれよりも高くなる傾向が強いことを示唆している．

3.3 参照要約あたりのオラクル要約の数

提案法は貪欲法とは異なり，すべてのオラクル要約を列挙する．2 つのデータセットに対し，1 つの参照要約が持つオラクル要約数の中央値，最大値，最小値を表 2 に示す．最小値の行のカッコ内の数値は総参照要約数に対する 1 つのオラクル要約しか持たない参照要約の割合を示す．表から明らかなとおり，8 から 9 割程度の参照要約が複数のオラクル要約を持ち，その数の中央値は 7〜10 程度である．

3.4 探索空間

提案法によりどの程度探索空間を削減できたかを調べるため，部分和問題の動的計画アルゴリズム [Cormen 09] に基づき長さ制約を満たす全ての組合せ，つまり実行可能解の数[1]を数え，提案法が探索途中に訪れたノード数と比較した．その結果を表 3 に示す．表より，実行可能解の数に対し，提案法のが訪れたノード数の中央値は 10^9 分の 1，最大値は 10^{12} から 10^{17} 分の 1，最小値は 10^5 から 10^8 分の 1 程度であり，探索空間を大幅に狭めることができた．これは，貪欲法による下限値が実際にはオラクル要約の ROUGE スコアと大差ないことが大きな要因であろう．また，$n=1$ と $n=2$ を比較した場合，$n=2$ の場合の探索効率が良い．これは，n を大きくすると式 (6) の上限値を見積もる不等式の右辺と左辺の差が小さくなるからであろう．

[1]探索木において長さ制約を満たすノード数に対応する．

4 おわりに

本稿では，与えられた文集合から ROUGE_n スコアを最大にするオラクル要約を列挙するアルゴリズムを分枝限定法に基づき提案した．DUC-03,04 のコーパスをもちいてオラクル要約を列挙し，提案手法によるオラクル要約と貪欲法による近似オラクル要約の ROUGE_n スコアを比較したところ，提案法によるオラクル要約の ROUGE_n スコアの方が統計的有意に高いことを確認した．また，80 から 90%の参照要約に対し，2 つ以上のオラクル要約が存在することを明らかにした．さらに，提案法の探索効率が非常に良いことを長さ制約を満たす文の集合の数と比較し，明らかにした．

参考文献

[Cormen 09] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E.: *Introduction to Algorithms*, The MIT Press, 3rd edition (2009)

[Khuller 99] Khuller, S., Moss, A., and Naor, J. S.: The Budgeted Maximum Coverage Problem, *Inf. Process. Lett.*, Vol. 70, No. 1, pp. 39–45 (1999)

[Land 60] Land, A. H. and Doig, A. G.: An Automatic Method of Solving Discrete Programming Problems, *Econometrica*, Vol. 28, No. 3, pp. 497–520 (1960)

[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)

[Lin 11] Lin, H. and Bilmes, J.: A Class of Submodular Functions for Document Summarization, in *Proc. of the 49th ACL/HLT*, pp. 510–520 (2011)

[Sipos 12] Sipos, R., Shivaswamy, P., and Joachims, T.: Large-Margin Learning of Submodular Summarization Models, in *Proc. of the 13th EACL*, pp. 224–233 (2012)

[Wilcoxon 45] Wilcoxon, F.: Individual Comparisons by Ranking Methods, *Biometrics Bulletin*, Vol. 1, No. 6, pp. 80–83 (1945)